

Diseño de Software en el Ámbito de las Actividades I+D

Fundamentos y Recursos

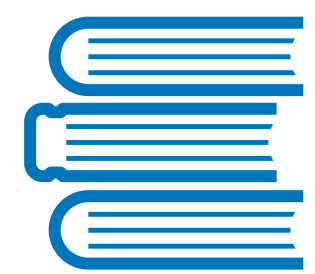
"A journey of a thousand miles begins with a single step"

Lao Tzu

Objeto de la Presentación



Presentar los **fundamentos** de la actividad ***Desarrollo de Software Científico - Técnico***.



Analizar algunos **recursos** básicos en relación con la generación de software.

PARTE I: FUNDAMENTOS DE LA ACTIVIDAD

Contenido

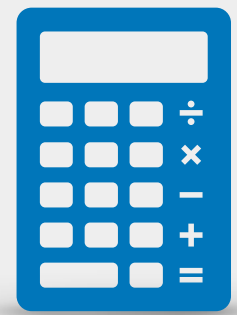
Parte I - Fundamentos de la actividad

1. Objetivos
2. Entorno
3. Antecedentes
4. Análisis
5. Escenario
6. Paradigma

(I - 1/6) Objetivos Generales

Desarrollo de Software Científico - Técnico

Objetivos



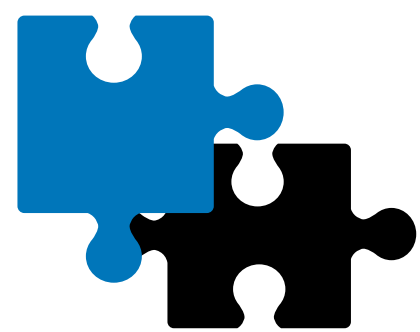
1. Desarrollar y poner a disposición del laboratorio una **capacidad tecnológica** que contribuya de un modo tangible a la mejora de sus **Procedimientos Técnicos** y **Proyectos I+D**.



2. Iniciar una **Línea de Investigación** en *exploración, desarrollo y aplicación* de herramientas avanzadas de software en el ámbito de la actividad realizada.

Desarrollo de Software Científico - Técnico

Objetivo 1: Desarrollo de Capacidad Tecnológica



- a) Poner a punto una **metodología general** para abordar la generación de software propio, cumpliendo los requisitos formales aplicables.

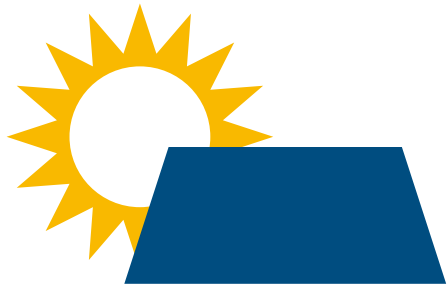


- b) Crear **herramientas de acceso abierto** en las que se apoyen, entre otros, los programas privados que operan en el laboratorio.

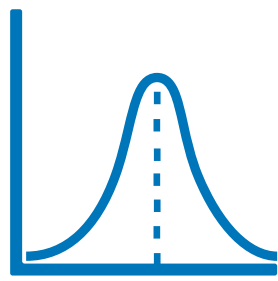
(I - 2/6) Análisis del Entorno de Trabajo

Desarrollo de Software Científico - Técnico

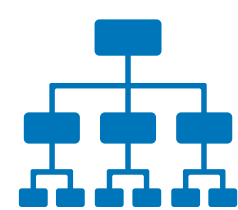
Entorno



Sectores objetivo (fotovoltaico, ...) en pleno crecimiento, desarrollo industrial y progreso tecnológico.



Actividad en Investigación y Desarrollo, donde es esperable cierto grado de innovación (posición activa) en la gestión de la información C-T.



Grupo de trabajo en el centro, que aconseja adecuarse a las NNTT y obtener retorno de la actividad realizada.



Promoción profesional, necesidad creciente de acreditar el trabajo realizado y producción científica.

Desarrollo de Software Científico - Técnico

Entorno

PERSPECTIVA...una diminuta muestra de retos presentes y futuros:

EVOLUCIÓN TECNOLÓGICA PTs:

SENSORES CON SALIDA DIGITAL
CALIBR. OTROS DISPOSITIVOS, SENSORES
EVOLUCIÓN SISTEMAS DE MEDIDA

NUEVOS PROCEDIMIENTOS:

CALIBRACIÓN PIRANÓMETROS EN INTERIOR
CALIBRACIÓN CURVAS I-V MÓDULOS FV
Etc.

VALIDACIÓN, ESTANDARIZACIÓN, ACCESO ABIERTO:

HERRAMIENTAS SOFTWARE ESTÁNDAR.
EVALUACIÓN DE CRITERIOS, ALGORITMOS, ETC.
DOCUMENTACIÓN AMPLIA, CÓDIGO Y APLICACIONES.

OTROS ASPECTOS RELACIONADOS CON NNTT:

WEB GRUPO (INFORMACIÓN INCRUSTADA)
ELABORACIÓN MATERIAL, INFORMACIÓN
PLATAFORMAS FORMACIÓN, DIVULGACIÓN, Etc.

BASES DE DATOS:

ACCESO Y MANEJO ESTÁNDAR
CONFIDENCIALIDAD, CONTROL ACCESO
ESTADÍSTICAS, EVALUACIÓN
ANÁLISIS CIENTÍFICO
FORMULACIÓN NECESIDADES
Etc.

DISPONIBILIDAD DE DATOS:

PERFIL DIARIO IRRADIANCIA SOLAR
OTRAS VARIABLES INTERÉS FV, IRRAD.SOLAR

DATA SCIENCE:

HERRAMIENTAS DE GRAN POTENCIA (**BIG DATA**)
ALGORITMOS COMPLEJOS: REGULARIZACIÓN, OPTIMIZACIÓN, CONVOLUTIONAL
NETWORKS AND SEQUENCE MODELING (**DEEP LEARNING**)
OTROS ALGORITMOS ADAPTATIVOS (**MACHINE LEARNING**)
REPRESENTACIÓN Y MOD. DATOS (**REPRESENTATION LEARNING**)
OTRAS HERRAMIENTAS DE **INTELIGENCIA ARTIFICIAL**

NECESIDADES INTERNAS
EXIGENCIAS EXTERNAS

(I - 3/6) Antecedentes de la Actividad

Actividad de J.P. Silva en el ámbito del Laboratorio de Componentes y Sistemas Fotovoltaicos (U.E.S.F. - CIEMAT)

Desarrollo de Software Científico - Técnico

Antecedentes

AÑO	ÁMBITO	APLICACIÓN	RETORNO
2007	TESIS DOCTORAL	MEDIDAS	<i>“Experimental Measurement of Working Temperature of PV Modules Considering Electrical Performance and Proposal of a New Indoor NOCT Determination”.</i> Sep-2007. Silva J.P., Chenlo F., Balenzategui J.L., Procs. of the 22th PVSEC. pp. 2740-2742. <i>“On the Relationship between Spectral Reflectance and Working Temperature of PV Modules”.</i> Sep-2008. Silva J.P., Balenzategui J.L., Procs. of the 23rd PVSEC. pp. 2861-2864.
2008		STEADY	
2009		DYNAMIC	
2011	INNDISOL / EVADIFOT S.E.C. Módulos FV	PVPOWER	<i>“Spectral Reflectance Patterns of Photovoltaic Modules and Their Thermal Effects”.</i> Silva J.P., Nofuentes G., Muñoz J.V., Journal of Solar Energy Engineering. Noviembre 2010, Vol. 132. 041016-1. DOI: 10.1115/1.4002246. <i>“Temperatura de Operación de Módulos Fotovoltaicos”.</i> Silva J.P., Colección Documentos Ciemat, 2019 (90 pags.). Ed. Ciemat. ISBN: 978-84-7834-826-8.
2012	CALENER	CALENER 2011	
2013...	S.E.C. Sensores RS (varias versiones)	SENSORS	

Desarrollo de Software Científico - Técnico

Antecedentes

AÑO	ÁMBITO	APLICACIÓN	RETORNO
2007	TESIS DOCTORAL	MEDIDAS	<i>“Aplicación para el Manejo de Curvas Características de Módulos Fotovoltaicos – PVPower 2011”</i>. Silva J.P., Vela N., Informe Proy. Inndisol - Sub.INNPACTO. Nov-2011. <i>“Caracterización de Módulos de Diversas Tecnologías en el Marco del Proyecto Inndisol”</i>. Silva J.P., Asenjo B., Vela N., Chenlo F. Actas del XV Congr.Ibérico y X C.Iberoamericano de E.S. pp. 427-432. Junio 2012. <i>“Ensayos de caracterización y evaluación de módulos Fotovoltaicos (PV rating)”</i>. Silva J.P., Inndisol Project Meeting. Febrero 2012. <i>“Determinación de Parámetros Eléctricos y Métodos de Corrección de Curvas Características de Módulos Fotovoltaicos”</i>. Silva J.P., Fabero F., Chenlo F. Informe 5.8. Informe de Proyecto (23 pags.). Proyecto EVADIFOT. <i>“Caracterización de Módulos Fotovoltaicos Representativos de las Centrales de Fotowatio”</i>. Silva J.P., Fabero F., Chenlo F. Informes 5.1-5.7, 5.9-5.11 (112 pags. c/u). Proyecto EVADIFOT.
2008		STEADY	
2009		DYNAMIC	
2011	INNDISOL / EVADIFOT S.E.C. Módulos FV	PVPOWER	
2012	CALENER	CALENER 2011	
2013...	S.E.C. Sensores RS (varias versiones)	SENSORS	

Desarrollo de Software Científico - Técnico

Antecedentes

AÑO	ÁMBITO	APLICACIÓN	RETORNO
2007	TESIS DOCTORAL	MEDIDAS	<i>“Temperature – Dependence of Small Grid-Connected Inverters: A Simplified Model for Dynamic Environments”</i>. Silva J.P., Alonso-García F., Chenlo F., Maderuelo E., Mejuto E. Proc. 27th PVSEC, pp. 3665-3668. <i>“A Comparative Analysis of PV Modules Efficiency from Five Different Cell-Technology Arrays: Experimental Results Within the Calener Project”</i>. Alonso-García M.C., Silva J.P. et al. Proc. 27th PVSEC, pp. 3275-3279. <i>“Aplicación para el manejo de Parámetros de Operación de Pequeñas Instalaciones Fotovoltaicas – Calener 2011”</i>. Silva J.P., Alonso-García M.C. Informe de Proyecto CALENER, 2011/05. Octubre 2012.
2008		STEADY	
2009		DYNAMIC	
2011	INNDISOL / EVADIFOT S.E.C. Módulos FV	PVPOWER	
2012	CALENER	CALENER 2011	
2013...	S.E.C. Sensores RS (varias versiones)	SENSORS	

Desarrollo de Software Científico - Técnico

Antecedentes

AÑO	ÁMBITO	APLICACIÓN	RETORNO
2007	TESIS DOCTORAL	MEDIDAS	<i>"Aplicación de Tratamiento de Datos para la Calibración de Dispositivos de Medida de la Irradiancia Solar". Silva J.P., Ariza – Brigidano M.A. Informe de la Aplicación. Octubre 2013.</i> <i>"PVLab DT-PVLab-02-02. Aplicación Excel de Tratamiento de Datos de Calibración de Piranómetros". Silva J.P. Febrero 2019.</i> <i>"PVLab DT-PVLab-02-01. Validación de Software para el Tratamiento de Datos de Calibración". Silva J.P. Abril 2019.</i> <i>"PVLab PT-PVLab-02-02 Ed.1: Tratamiento de Datos de Calibración de Piranómetros". Silva J.P. Diciembre 2020.</i>
2008		STEADY	
2009		DYNAMIC	
2011	INNDISOL / EVADIFOT S.E.C. Módulos FV	PVPOWER	
2012	CALENER	CALENER 2011	
2013	S.E.C. Sensores RS (varias versiones)	SENSORS	

Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación MEDIDAS

Ámbito de Trabajo: TESIS DOCTORAL.

Necesidades: VISUALIZACIÓN Y CORRELACIÓN DE VARIABLES CLIMÁTICAS, PROPIEDADES TÉRMICAS Y EFICIENCIA ELÉCTRICA DE MÓDULOS FV.

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño de Formularios: No
- Funciones Privadas (gestión de eventos): Si
- Funciones Públicas (módulos exportables ".bas"): Si (1)
- Implementación de Cálculos: celdas.

Capacidades: Generación automática de gráficos, conversión de magnitudes, composición de series numéricas experimentales, filtrado automático, cambio de escalas, unidades, etc.

Hoja datos procesados:		VER LOS DATOS ORIGINALES			Serie seleccionada
Fila Nombre	150	+ Aserie	+ Gserie	+ Tserie	Rango X / Rango Y
Fila inicial Valores	151				K&Z CM21 - 04150 - 11.18-10
Fila final Valores	5301				2V_TEMPBACK(C)
Rangos de la serie seleccionada		- Aserie	- Gserie	- Tserie	Seleccionar nuevos rangos
Name	=Procesadas!R150C14				Marcar como Rango X
XValues	=Procesadas!R151C7:R5301C7				Marcar como Rango Y
Values	=Procesadas!R151C14:R5301C14				
Rangos estándar para el eje X	102[Time stamp]	K&Z CM21 - 04150 - 11.18-10-6 V/Wm-2 (11.10-10-6 V/Wm-2)			
Series A = 0	Series T = 1	Series G = 1			
LISTADO DE CANALES VISUALIZADOS		ESTADO DEL CANAL			+ Datos
CANAL	COLUMNA N°				N° FILAS
102[Time stamp]	4	102[Time stamp]			IRRAD
102[Seconds]	5	102[Seconds]			
102[VDC]	6	102[VDC]			
K&Z CM21 - 04150 - 11.18-10	7	K&Z CM21 - 04150 - 11.18-10-6 V/Wm-2 (11.10-10-6 V/Wm-2)			GRÁFICO 1
21(C)	8	21(C)			
105[VDC]	9	105[VDC]			
A04[VDC]	10	A04[VDC]			GRÁFICO 2
-1(1-21(C)+0)*1*1(-9.52380352380352-Voc - Sanyo HIP230-HD	11	-1(1-21(C)+0)*1*1(-9.52380352380352-Voc - Sanyo HIP230-HD			
A01(C)	12	A01(C)			
A01(C)	13	A01(C)			GRÁFICO 3
2V_TEMPBACK(C)	14	2V_TEMPBACK(C)			
0(1-A01(C)+0)*1*1(1-Temperatura promedio, módulo en circuito s	15	0(1-A01(C)+0)*1*1(1-Temperatura promedio, módulo en circuito s			
A01(C)	16	A01(C)			GRÁFICO 4
A01(C)	17	A01(C)			
3V_TEMPBACK(C)	18	3V_TEMPBACK(C)			
0(1-A01(C)+0)*1*1(1-Temperatura promedio, módulo en circuito s	19	0(1-A01(C)+0)*1*1(1-Temperatura promedio, módulo en circuito s			GRÁFICO 5
A01(C)	20	A01(C)			
A01(C)	21	A01(C)			
4V_TEMPBACK(C)	22	4V_TEMPBACK(C)			GRÁFICO 6
0(1-A01(C)+0)*1*1(1-Temperatura promedio, módulo en circuito s	23	0(1-A01(C)+0)*1*1(1-Temperatura promedio, módulo en circuito s			
A01(C)	24	A01(C)			
A01(C)	25	A01(C)			GRÁFICO 7
5V_TEMPBACK(C)	26	5V_TEMPBACK(C)			
0(1-A01(C)+0)*1*1(1-Temperatura promedio, módulo en circuito s	27	0(1-A01(C)+0)*1*1(1-Temperatura promedio, módulo en circuito s			
203(C)	28	203(C)			GRÁFICO 8
203(C)	29	203(C)			
6V_TEMPBACK(C)	30	6V_TEMPBACK(C)			
0(1-203(C)+0)*1*1(1-Add. Tmp mediano aislado - POA87FA044	31	0(1-203(C)+0)*1*1(1-Add. Tmp mediano aislado - POA87FA044			GRÁFICO 9
203(C)	32	203(C)			
203(C)	33	203(C)			
7V_TEMPBACK(C)	34	7V_TEMPBACK(C)			GRÁFICO 10
0(1-203(C)+0)*1*1(1-Add. Tmp mediano aislado - POA87FA044	35	0(1-203(C)+0)*1*1(1-Add. Tmp mediano aislado - POA87FA044			
211(C)	36	211(C)			
211(C)	37	211(C)			GRÁFICO 11
203(C)	38	203(C)			
0(1-211(C)+0)*1*1(1-Add. Tmp pequeño aislado - POA87FA044	39	0(1-211(C)+0)*1*1(1-Add. Tmp pequeño aislado - POA87FA044			
211(C)	40	211(C)			GRÁFICO 12
203(C)	41	203(C)			
102[VDC]	42	102[VDC]			
-1(1-211(C)+0)*1*1(1-Tamb - interior cuadro+0)*1*1(2567.567567	43	-1(1-211(C)+0)*1*1(1-Tamb - interior cuadro+0)*1*1(2567.567567			GRÁFICO 13
203(C)	44	203(C)			
108[VDC]	45	108[VDC]			
102[VDC]	46	102[VDC]			GRÁFICO 14
1(-0.0035278-203(C)+0.1138)*1*1(-3.15-Señal (Vel. de viento) -	47	1(-0.0035278-203(C)+0.1138)*1*1(-3.15-Señal (Vel. de viento) -			
203(C)	48	203(C)			
203(C)	49	203(C)			

Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación MEDIDAS

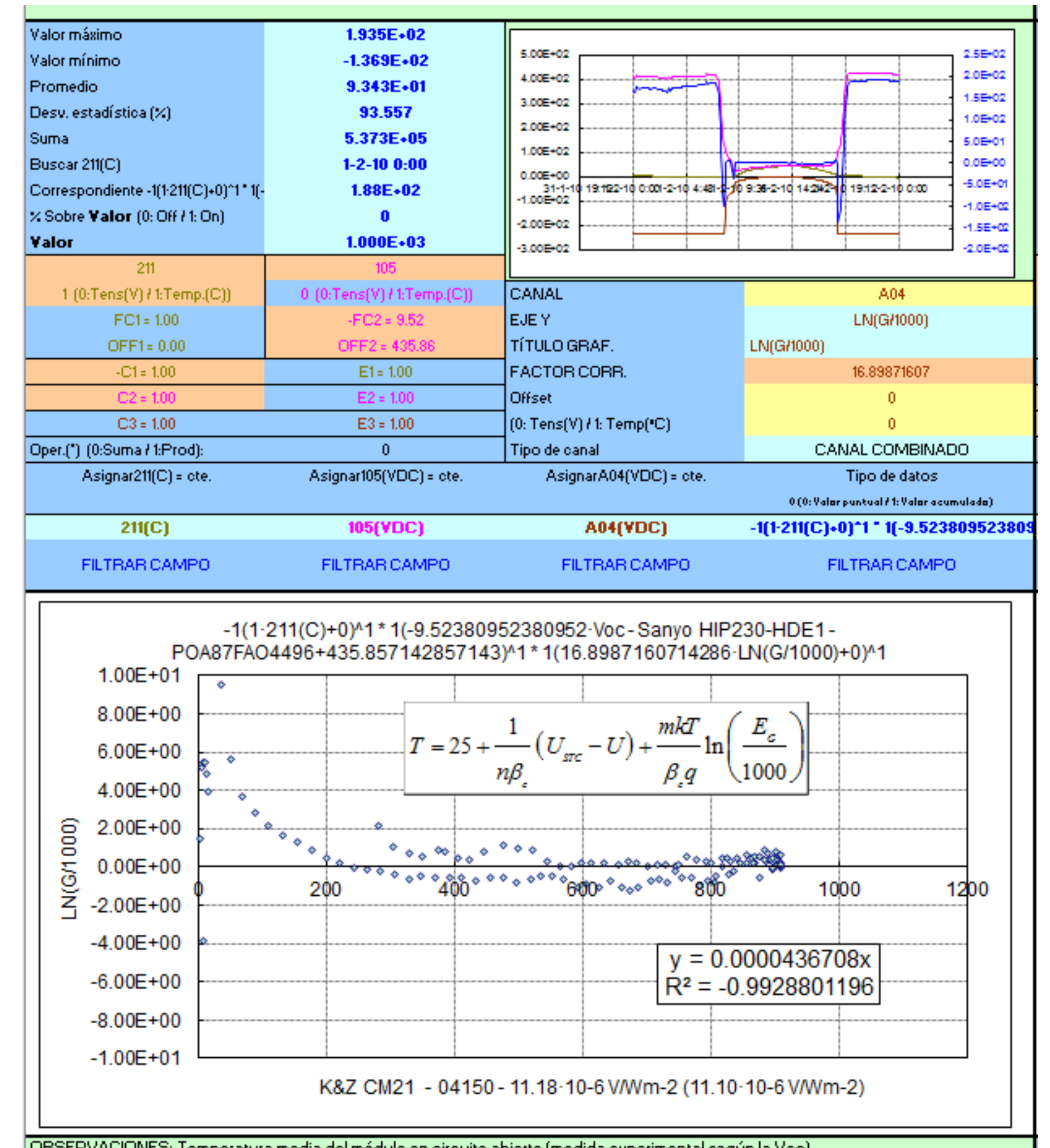
Ámbito de Trabajo: TESIS DOCTORAL.

Necesidades: VISUALIZACIÓN Y CORRELACIÓN DE VARIABLES CLIMÁTICAS, PROPIEDADES TÉRMICAS Y EFICIENCIA ELÉCTRICA DE MÓDULOS FV.

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño de Formularios: No
- Funciones Privadas (gestión de eventos): Si
- Funciones Públicas (módulos exportables ".bas"): Si (1)
- Implementación de Cálculos: celdas.

Capacidades: Generación automática de gráficos, conversión de magnitudes, composición de series numéricas experimentales, filtrado automático, cambio de escalas, unidades, etc.



Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación STEADY

Ámbito de Trabajo: TESIS DOCTORAL.

Necesidades: MODELIZACIÓN TÉRMICA DE MÓDULOS FV EN RÉGIMEN ESTACIONARIO POR BALANCE DE ENERGÍA, ANÁLISIS DE SENSIBILIDAD DE LA TEMPERATURA DE TRABAJO.

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: No
- Funciones Privadas (gestión de eventos): Si
- Funciones Públicas (módulos exportables ".bas"): Si (6)
- Implementación de cálculos: celdas.

Capacidades: Parametrización automática, resolución de funciones explícitas e implícitas, cálculo y ajustes iterativos, automatización de escenarios (cálculo recursivo), generación automática tablas de resultados y gráficos.

Module nº	1
Module Area (m ²)	1.277 m2
Eff @STC (%)	13.7%
Power Temp. Coeff.	-0.48% / °C
NOCT (°C)	48.0 °C
Tilt angle (°)	45.0 °
R (%)	8.28%
IR av. Refl.(4µm-25µm)	0.25
Emisividad (Superficie frontal)	0.880
Emisividad (Superficie posterior)	0.950
Emisividad (Suelo)	0.800
Emisividad (Cielo)	0.760
Glass Thickness (m)	0.0032 m
Frontal E.V.A. Thickness (m)	0.0005 m
Cell Thickness (m)	0.0003 m
Rear E.V.A. Thickness (m)	0.0005 m
Rear enc. Thickness (m)	0.0003 m
k (Glass)	1.000 W/m°C
k (Upper EVA)	0.150 W/m°C
k (Cell)	150 W/m°C
k (Lower EVA)	0.150 W/m°C
k (Rear Cover)	0.240 W/m°C
Contact Resistance1	1.0E-05 m2 °C/W
Contact Resistance2	1.0E-05 m2 °C/W
Ambient Temperature (°C)	20.00 °C
Ground Temperature (°C)	20.00 °C
Wind Temperature (°C)	20.00 °C
Frontal Windspeed (ms ⁻¹)	1.00 m/s
Rear Windspeed (ms ⁻¹)	0.75 m/s
Humidity(%)	35.0%
Brightness	80.0%
Global Irradiance, E _G (Wm ⁻²)	1000.00 W/m2
Voc/Pmax	MPP
G (4µm-44µm)	0.0 W/m2

Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación STEADY

Ámbito de Trabajo: TESIS DOCTORAL.

Necesidades: MODELIZACIÓN TÉRMICA DE MÓDULOS FV EN RÉGIMEN ESTACIONARIO POR BALANCE DE ENERGÍA, ANÁLISIS DE SENSIBILIDAD DE LA TEMPERATURA DE TRABAJO.

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: No
- Funciones Privadas (gestión de eventos): Si
- Funciones Públicas (módulos exportables ".bas"): Si (6)
- Implementación de cálculos: celdas.

Capacidades: Parametrización automática, resolución de funciones explícitas e implícitas, cálculo y ajustes iterativos, automatización de escenarios (cálculo recursivo), generación automática tablas de resultados y gráficos.

(#1) Actual (mono-Si) - vidrio templado 3.2 mm 72s x 1p - 5" (5.23 A); Glass - EVA - TPT	Cell Temperature	49.84 °C	
	Frontal Surface Temp.	46.85 °C	
	Rear Surface Temp.	47.99 °C	
Heat Fluxes	Frontal Side	Rear Side	Flujo de calor total
Radiación al suelo	20.1 W/m2	124.6 W/m2	144.6 W/m2
Radiación al cielo	153.2 W/m2	29.1 W/m2	182.3 W/m2
Convección forzada	141.2 W/m2	112.5 W/m2	253.6 W/m2
Convección natural	141.5 W/m2	74.5 W/m2	215.9 W/m2
	Total (Frontal)	Total (Rear)	$\Sigma q \cdot (1-R-\eta)G$
TOTALS	455.9 W/m2	340.6 W/m2	0.00 W/m2

Natural Convection Adjust

Set module

	Free parameters - set in "Set module"
	Free parameters - not set in "Set module"

Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación STEADY

Ámbito de Trabajo: TESIS DOCTORAL.

Necesidades: MODELIZACIÓN TÉRMICA DE MÓDULOS FV EN RÉGIMEN ESTACIONARIO POR BALANCE DE ENERGÍA, ANÁLISIS DE SENSIBILIDAD DE LA TEMPERATURA DE TRABAJO.

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: No
- Funciones Privadas (gestión de eventos): Si
- Funciones Públicas (módulos exportables ".bas"): Si (6)
- Implementación de cálculos: celdas.

Capacidades: Parametrización automática, resolución de funciones explícitas e implícitas, cálculo y ajustes iterativos, automatización de escenarios (cálculo recursivo), generación automática tablas de resultados y gráficos.

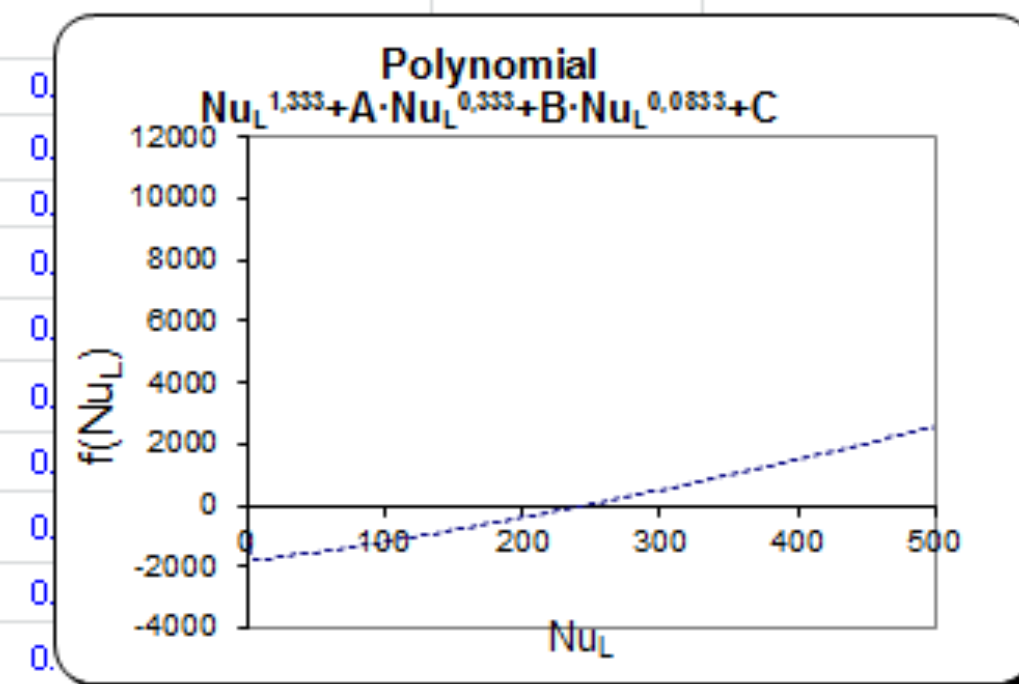
FORCED CONVECTION EXPRESSIONS		DESCRIPTION		
$h_{lam}=(k/L) \cdot 0,664 \cdot Re_L^{-1/2} \cdot Pr^{1/3}$	3.15 W/m2 *C	Laminar Flow	Flat Plates	
$h_{lam}=(k/L) \cdot 0,17 \cdot Re_L^{0,47}$	0.65 W/m2 *C	Laminar Flow	Collectors Pr=0.7	
$h_{lam}=(k/L) \cdot 0,6795 \cdot Re^{-1/2} \cdot Pr^{1/3}$	3.22 W/m2 *C	Laminar Flow	Exact Expr. - Known qw	
$h_{lam}=(k/L) \cdot 0,6955 \cdot Re^{1/2} \cdot Pr^{1/3} / ((1+(0,0207/Pr)^{2/3})^{1/4})$	3.23 W/m2 *C	Laminar Flow	Exact Expr. - Known qw	
$h_{tur}=(k/L) \cdot 0,036 \cdot Re^{0,8} \cdot Pr^{0,33}$	4.88 W/m2 *C	Turbulent Flow	Flat Plates - Frontal Conv.	
$h_{tur}=(k/L) \cdot 0,036 \cdot Re^{0,8} \cdot Pr^{0,33}$	3.85 W/m2 *C	Turbulent Flow	Flat Plates - Rear Conv.	
$h_{tur}=(k/L) \cdot 0,062 \cdot Re^{0,62}$	1.26 W/m2 *C	Turbulent Flow	Collectors Pr=0.7	
GLOBAL CONVECTION EXPRESSIONS		DESCRIPTION		
$h=2,8+3,0 \cdot v$	5.80 W/m2 *C	Turbulent Flow	Frontal Side	
$h=0,75 \cdot (2,8+3,0 \cdot v)$	4.35 W/m2 *C	Turbulent Flow	Rear Side	
$h=8,6 \cdot v / L^{0,4}$	8.19 W/m2 *C	Turbulent Flow	Panels in Roofs	
$h=5,7+3,8 \cdot v$	9.50 W/m2 *C	Turbulent Flow	Solar Panels	
$h=9,0+2,8 \cdot v$	11.80 W/m2 *C	Turbulent Flow	ASRAHE	
$h_{conv}=(h_{nat}^3+h_{for}^3)^{1/3}$	6.16 W/m2 *C	Turbulent Flow	Frontal (Unst. layer)	
$h_{conv}=(h_{nat}^3+h_{for}^3)^{1/3}$	4.18 W/m2 *C	Turbulent Flow	Rear (Est. layer)	
NATURAL CONVECTION EXPRESSIONS		DESCRIPTION		
$h_{nat}=0,95[(t_{placa}-t_{ambiente})\cdot \sin \phi]$	2.57 W/m2 *C	Laminar Flow	Estable layer- ΔT known	
$h_{nat}=(k/L) \cdot 0,56 \cdot (Ra_L \cdot \cos \theta)^{1/4}$	2.53 W/m2 *C	Turbulent Flow	Estable layer- ΔT known	
$h_{nat}=(k/L) \cdot 0,56^{4/5} \cdot (Ra_L \cdot \cos \theta)^{1/5}$	2.90 W/m2 *C	Turbulent Flow	Estable layer- qw known	
$h_{nat}=(k/L) \cdot 0,14[(Gr_L \cdot Pr)^{1/3}-\dots]$	4.88 W/m2 *C	Turbulent Flow	Unstable layer- ΔT known	
$Nu_L^{1,333}+A \cdot Nu_L^{0,333}+B \cdot Nu_L^{0,8833}+C=0$	4.89 W/m2 *C	Turbulent Flow	Unstable layer- qw known	
$Nu_L \cdot "f(Nu_L)+0"$	244.91	A =	109.1	
$Gr/Re^2=$	1.03E+00	B =	-533.0	
$\Delta T = q_u/h_{nat}=$	28.9 *C	C =	-1368.1	
ROOT VALUE		SIGN CHANGE	Nu_L	$Nu_L^{1,333}+A \cdot Nu_L^{0,333}+B \cdot Nu_L^{0,8833}+C$
				-1368
				-1783
				-1757
				-1730
				-1702
				-1673
				-1644
				-1614
				-1584
				-1553
				-1521

Polynomial

$Nu_L^{1,333}+A \cdot Nu_L^{0,333}+B \cdot Nu_L^{0,8833}+C$

$f(Nu_L)$

Nu_L



Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación STEADY

Ámbito de Trabajo: TESIS DOCTORAL.

Necesidades: MODELIZACIÓN TÉRMICA DE MÓDULOS FV EN RÉGIMEN ESTACIONARIO POR BALANCE DE ENERGÍA, ANÁLISIS DE SENSIBILIDAD DE LA TEMPERATURA DE TRABAJO.

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: No
- Funciones Privadas (gestión de eventos): Si
- Funciones Públicas (módulos exportables ".bas"): Si (6)
- Implementación de cálculos: celdas.

Capacidades: Parametrización automática, resolución de funciones explícitas e implícitas, cálculo y ajustes iterativos, automatización de escenarios (cálculo recursivo), generación automática tablas de resultados y gráficos.

SKY EQ. TEMP.		4.76 °C	NON-REFL. IRR.		796.5 W/m2
Hum(%)		35.0%	G		1000.0 W/m2
Clearness Index		80.0%	$p_m G$		82.8 W/m2
$(T_{sky temp} = 0.0552 \cdot T_{amb}^{1.5})$		7.1 °C	$(1-p_m)G$		917.2 W/m2
$es(t) [-65^{\circ}C..0.01^{\circ}C]$		2837.05 Pa	Electr. Yield at Tmodule		12.1%
$es(t) [-45^{\circ}C..60^{\circ}C]$		2332.60 Pa	Voc/Pmax		MPP
$e(t)$		816.41 Pa	DC Converted		120.7 W/m2
Tdp3		4.06 °C	$(1-p_m-\eta_m)G$		796.5 W/m2
Tsky		0.95 °C	G (4 μ m-44 μ m)		0.0 W/m2
			$\epsilon_{IRm} G$		0.0 W/m2
			$(1-p_{IRm})G$		0.0 W/m2
Radiación térmica		a cielo (W/m ⁻² ·°C)	a suelo (W/m ⁻² ·°C)		
h_F		3.48	0.68		Frontal Side
h_R		0.65	4.27		Rear Side
Convección		natural (W/m ⁻² ·°C)	forzada (W/m ⁻² ·°C)		
h_F		4.89	4.88		Frontal Side
h_R		2.53	3.85		Rear Side
Conducción		$R_{t,b}$ (m ² ·°C/W)			
$R_{TH,F}$		0.0065			Frontal Side
$R_{TH,R}$		0.0054			Rear Side
TEMP. FRONTAL SURF.		T_{CELL}		$Q_F R_{COND.F}$	
$T_{FRONTAL} = T_{CELL} - Q_F R_{FR}$		46.85 °C	49.84 °C		2.98 °C
TEMP. REAR SURF.		T_{CELL}		$Q_R R_{COND.R}$	
$T_{REAR} = T_{CELL} - Q_R R_{RE}$		47.99 °C	49.84 °C		1.85 °C
Heat Flux (Wm ⁻²)		$(\sigma_i / \sigma_{ij}) q_{ij}$ Frontal	$(\sigma_j / \sigma_{ij}) q_{ij}$ Back	Total	
Radiation - Ground		20.1	124.6		144.6
Radiation - Sky		153.2	29.1		182.3
Forced convection		141.2	112.5		253.6
Natural convection		141.5	74.5		215.9
Total - output		455.9	340.6		796.5
$R \cdot E_G$					82.8
$\eta \cdot E_G$					120.7
Total - input		$(1-R-\eta)E_G$			796.5

Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación STEADY

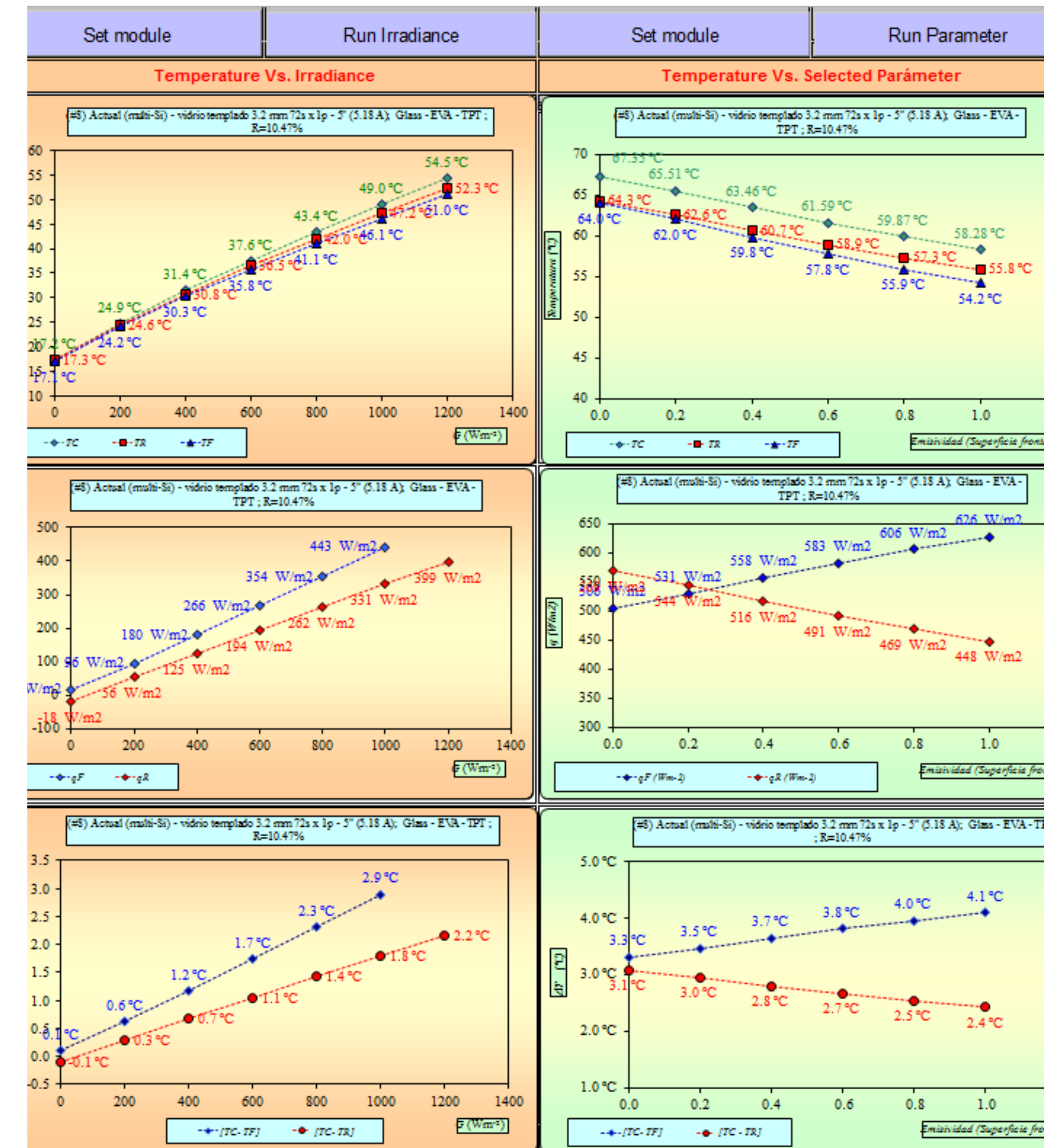
Ámbito de Trabajo: TESIS DOCTORAL.

Necesidades: MODELIZACIÓN TÉRMICA DE MÓDULOS FV EN RÉGIMEN ESTACIONARIO POR BALANCE DE ENERGÍA, ANÁLISIS DE SENSIBILIDAD DE LA TEMPERATURA DE TRABAJO.

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: No
- Funciones Privadas (gestión de eventos): Si
- Funciones Públicas (módulos exportables ".bas"): Si (6)
- Implementación de cálculos: celdas.

Capacidades: Parametrización automática, resolución de funciones explícitas e implícitas, cálculo y ajustes iterativos, automatización de escenarios (cálculo recursivo), generación automática tablas de resultados y gráficos.



Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación DYNAMIC

Ámbito de Trabajo: TESIS DOCTORAL.

Necesidades: MODELIZACIÓN TÉRMICA DE MÓDULOS FV EN RÉGIMEN DINÁMICO POR BALANCE DE ENERGÍA, EVOLUCIÓN DE TODAS LAS VARIABLES IMPLICADAS.

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: No
- Funciones Privadas (gestión de eventos): No
- Funciones Públicas (módulos exportables ".bas"): Si (1)
- Implementación de cálculos: código VB.

Capacidades: Separación total de cálculos (código VB) Vs. interface gráfica (Excel). Almacenamiento de cálculos en vectores (exportable para todas las variables).

Nº	symbol	description	Calculate	value
1	TA	Tilted angle (degrees, fixed system)		3.000E+01
2	AZIM	Azimuth (fixed system)		1.800E+02
3	CINDEX	Clearness index (sky temperature purposes)		0.000E+00
4	ALCOEFFICIENT	Angular losses coefficient (mod. N. Martín)		1.800E-01
5	CCP1	Calibration constant pyranometer #1		9.009E+04
6	CCP2	Calibration constant pyranometer #2		9.009E+04
7	TCPMAX	Thermal coefficient of Pmax (%/°C)		-5.000E-03
8	EFFSTC	Module efficiency at STC		1.371E-01
9	BSE	Back surface emissivity		9.500E-01
10	FSE	Frontal surface emissivity		8.800E-01
11	GSE	Ground/Floor surface emissivity		8.000E-01
12	SKYE	Sky emissivity		7.600E-01
13	BST	Back surface thickness		3.000E-04
14	CELLT	Cell thickness		3.000E-04
15	EYAT1	Frontal E.V.A. thickness		5.000E-04
16	EYAT2	Rear E.V.A. thickness		5.000E-04
17	GLST	Glass thickness		3.200E-03
18	R	Reflected G(ASTM173-03) 300nm-2500 nm		1.047E-01
19	T	Transmitted G(ASTM173-03) 300nm-2500 nm		9.050E-01
20	FIXED	Fixed (1) or two-axes tracking (2)		1
21	CF	Covering factor for glass-glass PV modules		8.800E-01
22	RH	Relative Humidity (%)		3.500E-01
23	LAT	Latitude of location		4.045E+01
24	CL	Characteristic length of the PV module (for determination of adimens)		1.130E+00
25	LONGITUDE	Longitude of location		-3.727E+00
26	NDAY	Day number		1.970E+02
27	NOCT	Nominal Operating Cell Temperature		4.800E+01
28	VP	Working point: open circuit (0) or maximum power point (1)		1
29	PSTC	Power at STC (W)		1.750E+02
30	AP	Atmospheric pressure (millibars)		1.013E+03
31	ET	Encapsulant type: glass-E.V.A.-T.P.T. (0) or glass-E.V.A.-glass (1)		1
32	GT	Glass transmittance		9.060E-01
Nº	symbol	measuring channels	Heading	
1	CMP1	Pyranometer1 channel	102(VDC)	
2	CMP2	Pyranometer2 channel	102(VDC)	
3	CMTA1	Ambient temperature1 channel	203(C)	
4	CMTA2	Ambient temperature2 channel	203(C)	
5	CMTG	Ground temperature channel	204(C)	
6	CMVY	Wind speed channel	108(VDC)	

Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación DYNAMIC

Ámbito de Trabajo: TESIS DOCTORAL.

Necesidades: MODELIZACIÓN TÉRMICA DE MÓDULOS FV EN RÉGIMEN DINÁMICO POR BALANCE DE ENERGÍA, EVOLUCIÓN DE TODAS LAS VARIABLES IMPLICADAS.

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: No
- Funciones Privadas (gestión de eventos): No
- Funciones Públicas (módulos exportables ".bas"): Si (1)
- Implementación de cálculos: código VB.

Capacidades: Separación total de cálculos (código VB) Vs. interface gráfica (Excel). Almacenamiento de cálculos en vectores (exportable para todas las variables).

Nº	symbol	description	Act.
1	V_AL(VDC)	Angular losses	1
2	V_AMBTEMP1(C)	Ambient temperature 1	
3	V_AMBTEMP2(C)	Ambient temperature 2	
4	V_AMBTEMPAVERAGED(C)	Averaged ambient temperature	
5	V_AOI(VDC)	Angle of incidence	1
6	V_CLEARNESS(VDC)	Clearness index	
7	V_CONDMODAMB(VDC)	Thermal conductance (PV module - ambient)	
8	V_CONDMODAMBB(VDC)	Rear thermal conductance (PV module - ambient)	
9	V_CONDMODAMBF(VDC)	Frontal thermal conductance (PV module - ambient)	
10	V_CONDMODGRND(VDC)	Thermal conductance (PV module - ground)	
11	V_CONDMODGRNDB(VDC)	Rear thermal conductance (PV module - ground)	
12	V_CONDMODGRNDF(VDC)	Frontal thermal conductance (PV module - ground)	
13	V_CONDMODSKY(VDC)	Thermal conductance (PV module - sky)	
14	V_CONDMODSKYB(VDC)	Rear thermal conductance (PV module - sky)	
15	V_CONDMODSKYF(VDC)	Frontal thermal conductance (PV module - sky)	
16	V_CONDMODWIND(VDC)	Thermal conductance (PV module - wind)	
17	V_CONDMODWINDB(VDC)	Rear thermal conductance (PV module - wind)	
18	V_CONDMODWINDF(VDC)	Frontal thermal conductance (PV module - wind)	
19	V_DELTATEMP(C)	Increment of temperature	
20	V_DELTATIME(VDC)	Time step	
21	V_DEWPOINTTEMP(C)	Dew point temperature	
22	V_EFFCURRTEMP(VDC)	PV module efficiency at current temperature	
23	V_FINALTEMP(C)	Final PV module temperature	1
24	V_GDCCONV(VDC)	Electricity - converted fraction of G	
25	V_GHEATCONV(VDC)	Heat - converted fraction of G	
26	V_GRASHOFFN(VDC)	Grashoff Adimensional Number	
27	V_GREFL(VDC)	Reflected fraction of G	
28	V_GRNDTEMP(C)	Ground temperature	
29	V_GTOTAL1(VDC)	Total G (pyranometer #1)	
30	V_GTOTAL2(VDC)	Total G (pyranometer #2)	
31	V_GTOTALAVERAGED(VDC)	Averaged G	1
32	V_GTRANSM(VDC)	Transmitted fraction of G	
33	V_HCFB(VDC)	Forced convection coefficient (back side)	
34	V_HCFF(VDC)	Forced convection coefficient (frontal side)	
35	V_HCNB(VDC)	Natural convection coefficient (back side)	
36	V_HCNF(VDC)	Natural convection coefficient (frontal side)	
37	V_HOUR(VDC)	Daily hour	
38	V_HRADBGRND(VDC)	Radiation coefficient (back side - ground)	
39	V_HRADBSKY(VDC)	Radiation coefficient (back side - sky)	

Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación DYNAMIC

Ámbito de Trabajo: TESIS DOCTORAL.

Necesidades: MODELIZACIÓN TÉRMICA DE MÓDULOS FV EN RÉGIMEN DINÁMICO POR BALANCE DE ENERGÍA, EVOLUCIÓN DE TODAS LAS VARIABLES IMPLICADAS.

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: No
- Funciones Privadas (gestión de eventos): No
- Funciones Públicas (módulos exportables ".bas"): Si (1)
- Implementación de cálculos: código VB.

Capacidades: Separación total de cálculos (código VB) Vs. interface gráfica (Excel). Almacenamiento de cálculos en vectores (exportable para todas las variables).

40	V_HRADFRND(VDC)	Radiation coefficient (frontal side - ground)	
41	V_HRADFSKY(VDC)	Radiation coefficient (frontal side - sky)	
42	V_INITEMP(C)	Initial PV module temperature	
43	V_MODTEMPTONC(C)	NOCT	
44	V_QBACK(VDC)	Total dissipated heat flux (rear)	
45	V_QFRONT(VDC)	Total dissipated heat flux (frontal)	
46	V_QMODAMB(VDC)	Heat flux (PV module - ambient)	
47	V_QMODAMBB(VDC)	Rear heat flux (PV module - ambient)	
48	V_QMODAMBF(VDC)	Frontal heat flux (PV module - ambient)	
49	V_QMODGRND(VDC)	Heat flux (PV module - ground)	
50	V_QMODGRNDB(VDC)	Rear heat flux (PV module - ground)	
51	V_QMODGRNDF(VDC)	Frontal heat flux (PV module - ground)	
52	V_QMODSKY(VDC)	Heat flux (PV module - sky)	
53	V_QMODSKYB(VDC)	Rear heat flux (PV module - sky)	
54	V_QMODSKYF(VDC)	Frontal heat flux (PV module - sky)	
55	V_QMODWIND(VDC)	Heat flux (PV module - wind)	
56	V_QMODWINDB(VDC)	Rear heat flux (PV module - wind)	
57	V_QMODWINDF(VDC)	Frontal heat flux (PV module - wind)	
58	V_QTOT(VDC)	Total dissipated heat flux	
59	V_RAYLEIGHN(VDC)	Rayleigh number	
60	V_REYNOLDSN(VDC)	Reynolds number	
61	V_RTHMODAMB(VDC)	Thermal resistance (PV module - ambient)	
62	V_RTHMODAMBB(VDC)	Rear thermal resistance (PV module - ambient)	
63	V_RTHMODAMBF(VDC)	Frontal thermal resistance (PV module - ambient)	
64	V_RTHMODGRND(VDC)	Thermal resistance (PV module - ground)	
65	V_RTHMODGRNDB(VDC)	Rear thermal resistance (PV module - ground)	
66	V_RTHMODGRNDF(VDC)	Frontal thermal resistance (PV module - ground)	
67	V_RTHMODSKY(VDC)	Thermal resistance (PV module - sky)	
68	V_RTHMODSKYB(VDC)	Rear thermal resistance (PV module - sky)	
69	V_RTHMODSKYF(VDC)	Frontal thermal resistance (PV module - sky)	
70	V_RTHMODWIND(VDC)	Thermal resistance (PV module - wind)	
71	V_RTHMODWINDB(VDC)	Rear thermal resistance (PV module - wind)	
72	V_RTHMODWINDF(VDC)	Frontal thermal resistance (PV module - wind)	
73	V_SKYTEMPPPARCLEAR(C)	Sky temperature (affected by the clearness index)	
74	V_SKYTEMPTOTCLEAR(C)	Sky temperature (not affected by the clearness index)	
75	V_SLOPE(VDC)	PV module tilt angle	
76	V_TEMPBACK(C)	Rear module temperature	1
77	V_TEMPFRONT(C)	Frontal module temperature	1
78	V_TEQ(C)	Temperature of equilibrium	1
79	V_UTOT(VDC)	Global heat transmission coefficient (ref. ambient temperature)	
80	V_WETAIRSATPRESS(VDC)	Saturation pressure of wet air	
81	V_WINDSPEED(VDC)	Frontal wind speed	
82	V_WINDTEMP(C)	Wind temperature	

Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación DYNAMIC

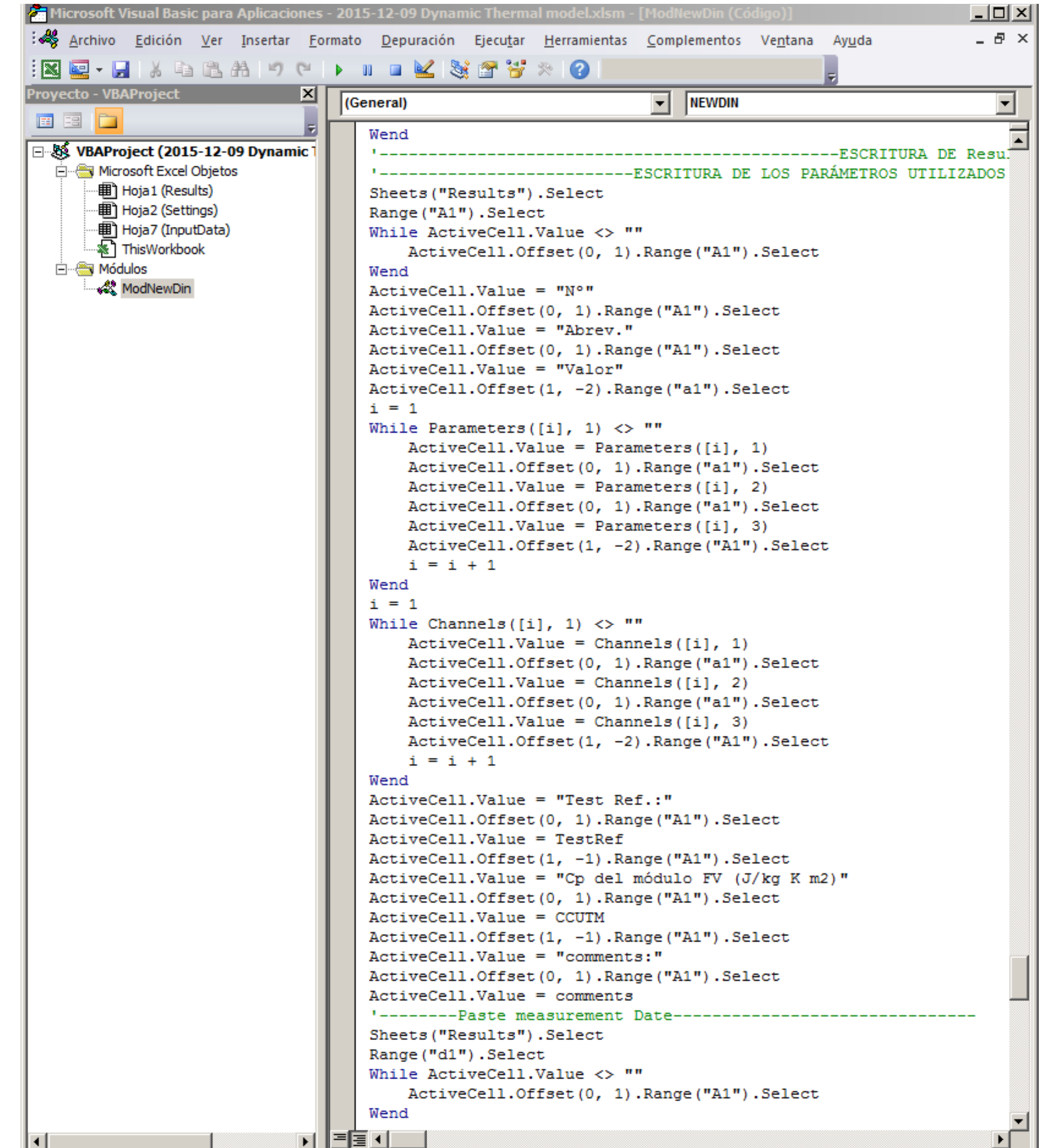
Ámbito de Trabajo: TESIS DOCTORAL.

Necesidades: MODELIZACIÓN TÉRMICA DE MÓDULOS FV EN RÉGIMEN DINÁMICO POR BALANCE DE ENERGÍA, EVOLUCIÓN DE TODAS LAS VARIABLES IMPLICADAS.

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: No
- Funciones Privadas (gestión de eventos): No
- Funciones Públicas (módulos exportables ".bas"): Si (1)
- Implementación de cálculos: código VB.

Capacidades: Separación total de cálculos (código VB) Vs. interface gráfica (Excel). Almacenamiento de cálculos en vectores (exportable para todas las variables).



```
Wend
'-----ESCRITURA DE Resultados-----
'-----ESCRITURA DE LOS PARÁMETROS UTILIZADOS-----
Sheets("Results").Select
Range("A1").Select
While ActiveCell.Value <> ""
    ActiveCell.Offset(0, 1).Range("A1").Select
Wend
ActiveCell.Value = "Nº"
ActiveCell.Offset(0, 1).Range("A1").Select
ActiveCell.Value = "Abrev."
ActiveCell.Offset(0, 1).Range("A1").Select
ActiveCell.Value = "Valor"
ActiveCell.Offset(1, -2).Range("a1").Select
i = 1
While Parameters([i], 1) <> ""
    ActiveCell.Value = Parameters([i], 1)
    ActiveCell.Offset(0, 1).Range("a1").Select
    ActiveCell.Value = Parameters([i], 2)
    ActiveCell.Offset(0, 1).Range("a1").Select
    ActiveCell.Value = Parameters([i], 3)
    ActiveCell.Offset(1, -2).Range("A1").Select
    i = i + 1
Wend
i = 1
While Channels([i], 1) <> ""
    ActiveCell.Value = Channels([i], 1)
    ActiveCell.Offset(0, 1).Range("a1").Select
    ActiveCell.Value = Channels([i], 2)
    ActiveCell.Offset(0, 1).Range("a1").Select
    ActiveCell.Value = Channels([i], 3)
    ActiveCell.Offset(1, -2).Range("A1").Select
    i = i + 1
Wend
ActiveCell.Value = "Test Ref.:"
ActiveCell.Offset(0, 1).Range("A1").Select
ActiveCell.Value = TestRef
ActiveCell.Offset(1, -1).Range("A1").Select
ActiveCell.Value = "Cp del módulo FV (J/kg K m2)"
ActiveCell.Offset(0, 1).Range("A1").Select
ActiveCell.Value = CCUTM
ActiveCell.Offset(1, -1).Range("A1").Select
ActiveCell.Value = "comments:"
ActiveCell.Offset(0, 1).Range("A1").Select
ActiveCell.Value = comments
'-----Paste measurement Date-----
Sheets("Results").Select
Range("d1").Select
While ActiveCell.Value <> ""
    ActiveCell.Offset(0, 1).Range("A1").Select
Wend
```


Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación DYNAMIC

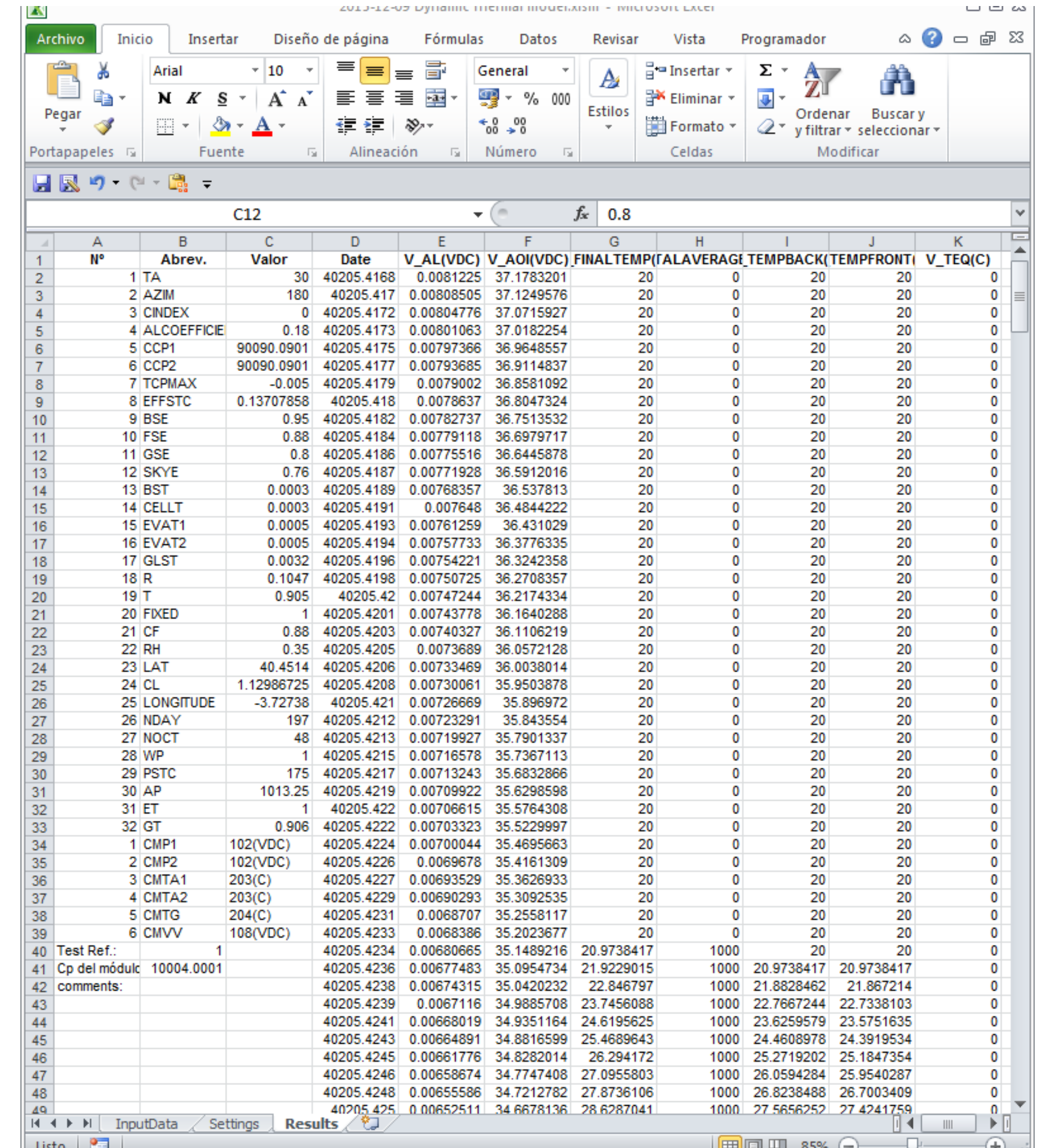
Ámbito de Trabajo: TESIS DOCTORAL.

Necesidades: MODELIZACIÓN TÉRMICA DE MÓDULOS FV EN RÉGIMEN DINÁMICO POR BALANCE DE ENERGÍA, EVOLUCIÓN DE TODAS LAS VARIABLES IMPLICADAS.

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: No
- Funciones Privadas (gestión de eventos): No
- Funciones Públicas (módulos exportables ".bas"): Si (1)
- Implementación de cálculos: código VB.

Capacidades: Separación total de cálculos (código VB) Vs. interface gráfica (Excel). Almacenamiento de cálculos en vectores (exportable para todas las variables).



	A	B	C	D	E	F	G	H	I	J	K
	Nº	Abrev.	Valor	Date	V_AL(VDC)	V_AOI(VDC)	FINAL TEMP	T(AVERAGE	TEMPBACK	TEMPFRONT	V_TEQ(C)
1	1	TA	30	40205.4168	0.0081225	37.1783201	20	0	20	20	0
2	2	AZIM	180	40205.417	0.00808505	37.1249576	20	0	20	20	0
3	3	CINDEX	0	40205.4172	0.00804776	37.0715927	20	0	20	20	0
4	4	ALCOEFFICIE	0.18	40205.4173	0.00801063	37.0182254	20	0	20	20	0
5	5	CCP1	90090.0901	40205.4175	0.00797366	36.9648557	20	0	20	20	0
6	6	CCP2	90090.0901	40205.4177	0.00793685	36.9114837	20	0	20	20	0
7	7	TCPMAX	-0.005	40205.4179	0.0079002	36.8581092	20	0	20	20	0
8	8	EFFSTC	0.13707858	40205.418	0.0078637	36.8047324	20	0	20	20	0
9	9	BSE	0.95	40205.4182	0.00782737	36.7513532	20	0	20	20	0
10	10	FSE	0.88	40205.4184	0.00779118	36.6979717	20	0	20	20	0
11	11	GSE	0.8	40205.4186	0.00775516	36.6445878	20	0	20	20	0
12	12	SKYE	0.76	40205.4187	0.00771928	36.5912016	20	0	20	20	0
13	13	BST	0.0003	40205.4189	0.00768357	36.537813	20	0	20	20	0
14	14	CELLT	0.0003	40205.4191	0.007648	36.4844222	20	0	20	20	0
15	15	EVAT1	0.0005	40205.4193	0.00761259	36.431029	20	0	20	20	0
16	16	EVAT2	0.0005	40205.4194	0.00757733	36.3776335	20	0	20	20	0
17	17	GLST	0.0032	40205.4196	0.00754221	36.3242358	20	0	20	20	0
18	18	R	0.1047	40205.4198	0.00750725	36.2708357	20	0	20	20	0
19	19	T	0.905	40205.42	0.00747244	36.2174334	20	0	20	20	0
20	20	FIXED	1	40205.4201	0.00743778	36.1640288	20	0	20	20	0
21	21	CF	0.88	40205.4203	0.00740327	36.1106219	20	0	20	20	0
22	22	RH	0.35	40205.4205	0.0073689	36.0572128	20	0	20	20	0
23	23	LAT	40.4514	40205.4206	0.00733469	36.0038014	20	0	20	20	0
24	24	CL	1.12986725	40205.4208	0.00730061	35.9503878	20	0	20	20	0
25	25	LONGITUDE	-3.72738	40205.421	0.00726669	35.896972	20	0	20	20	0
26	26	NDAY	197	40205.4212	0.00723291	35.843554	20	0	20	20	0
27	27	NOCT	48	40205.4213	0.00719927	35.7901337	20	0	20	20	0
28	28	WP	1	40205.4215	0.00716578	35.7367113	20	0	20	20	0
29	29	PSTC	175	40205.4217	0.00713243	35.6832866	20	0	20	20	0
30	30	AP	1013.25	40205.4219	0.00709922	35.6298598	20	0	20	20	0
31	31	ET	1	40205.422	0.00706615	35.5764308	20	0	20	20	0
32	32	GT	0.906	40205.4222	0.00703323	35.5229997	20	0	20	20	0
33	1	CMP1	102(VDC)	40205.4224	0.00700044	35.4695663	20	0	20	20	0
34	2	CMP2	102(VDC)	40205.4226	0.0069678	35.4161309	20	0	20	20	0
35	3	CMTA1	203(C)	40205.4227	0.00693529	35.3626933	20	0	20	20	0
36	4	CMTA2	203(C)	40205.4229	0.00690293	35.3092535	20	0	20	20	0
37	5	CMTG	204(C)	40205.4231	0.0068707	35.2558117	20	0	20	20	0
38	6	CMVV	108(VDC)	40205.4233	0.0068386	35.2023677	20	0	20	20	0
39	Test Ref.:	1		40205.4234	0.00680665	35.1489216	20.9738417	1000	20	20	0
40	Cp del módulo	10004.0001		40205.4236	0.00677483	35.0954734	21.9229015	1000	20.9738417	20.9738417	0
41	comments:			40205.4238	0.00674315	35.0420232	22.846797	1000	21.8828462	21.867214	0
42				40205.4239	0.0067116	34.9885708	23.7456088	1000	22.7667244	22.7338103	0
43				40205.4241	0.00668019	34.9351164	24.6195625	1000	23.6259579	23.5751635	0
44				40205.4243	0.00664891	34.8816599	25.4689643	1000	24.4608978	24.3919534	0
45				40205.4245	0.00661776	34.8282014	26.294172	1000	25.2719202	25.1847354	0
46				40205.4246	0.00658674	34.7747408	27.0955803	1000	26.0594284	25.9540287	0
47				40205.4248	0.00655586	34.7212782	27.8736106	1000	26.8238488	26.7003409	0
48				40205.425	0.00652511	34.6678136	28.6287041	1000	27.5656252	27.4241759	0

Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación PVPOWER

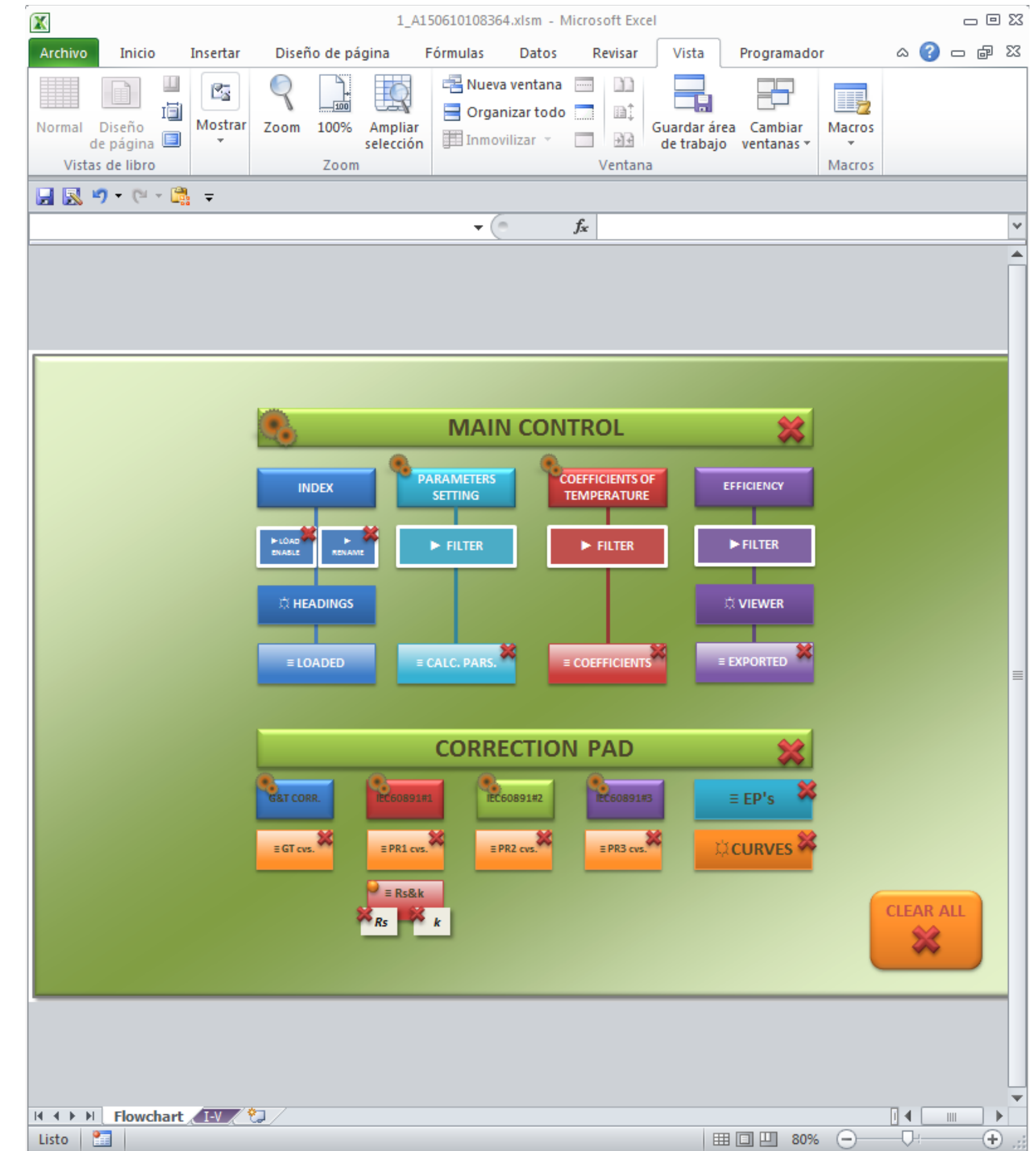
Ámbito de Trabajo: S.E. CALIBRACIÓN MÓDULOS FV, PROYECTO EVADIFOT, PROYECTO INNDISOL.

Necesidades: TRATAMIENTO DE DATOS DE MEDIDA DE CURVAS I-V DE MÓDULOS FV.

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: Si (3)
- Funciones Privadas (gestión de eventos): Si
- Funciones Públicas (módulos exportables ".bas"): Si (214 aprox.)
- Implementación de cálculos: celdas.

Capacidades: Interpolación, extrapolación y ajuste polinómico (automáticos), generación de tablas de resultados, exportación, ajuste a funciones teóricas, regresiones lineales y logarítmicas, funciones de filtrado en rangos variables, cálculo, visualización y almacenamiento de parámetros, etc.



Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación PVPOWER

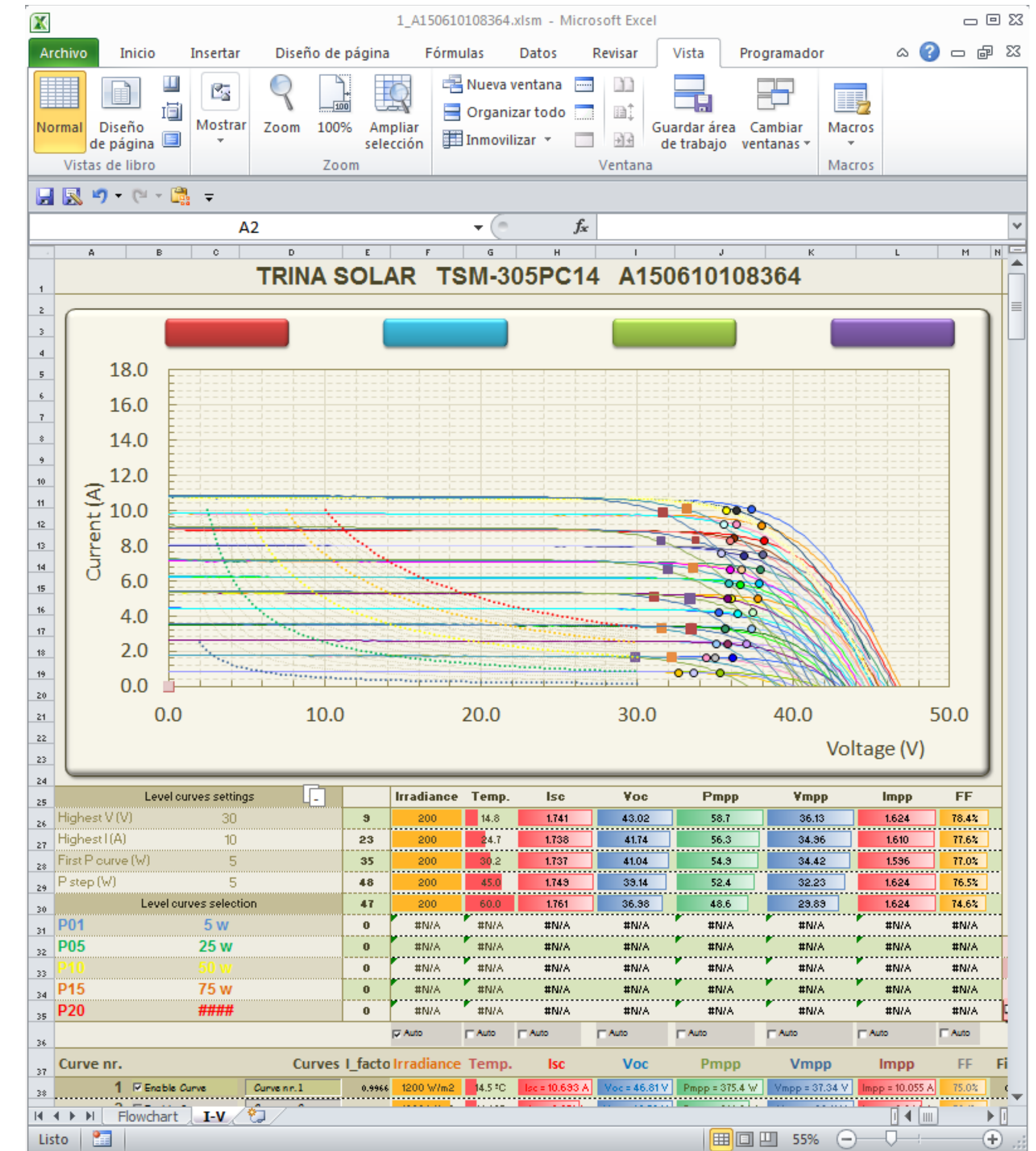
Ámbito de Trabajo: S.E. CALIBRACIÓN MÓDULOS FV, PROYECTO EVADIFOT, PROYECTO INNDISOL.

Necesidades: TRATAMIENTO DE DATOS DE MEDIDA DE CURVAS I-V DE MÓDULOS FV.

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: Si (3)
- Funciones Privadas (gestión de eventos): Si
- Funciones Públicas (módulos exportables ".bas"): Si (214 aprox.)
- Implementación de cálculos: celdas.

Capacidades: Interpolación, extrapolación y ajuste polinómico (automáticos), generación de tablas de resultados, exportación, ajuste a funciones teóricas, regresiones lineales y logarítmicas, funciones de filtrado en rangos variables, cálculo, visualización y almacenamiento de parámetros, etc.



Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación PVPOWER

Ámbito de Trabajo: S.E. CALIBRACIÓN MÓDULOS FV, PROYECTO EVADIFOT, PROYECTO INNDISOL.

Necesidades: TRATAMIENTO DE DATOS DE MEDIDA DE CURVAS I-V DE MÓDULOS FV.

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: Si (3)
- Funciones Privadas (gestión de eventos): Si
- Funciones Públicas (módulos exportables ".bas"): Si (214 aprox.)
- Implementación de cálculos: celdas.

Capacidades: Interpolación, extrapolación y ajuste polinómico (automáticos), generación de tablas de resultados, exportación, ajuste a funciones teóricas, regresiones lineales y logarítmicas, funciones de filtrado en rangos variables, cálculo, visualización y almacenamiento de parámetros, etc.

Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación PVPOWER

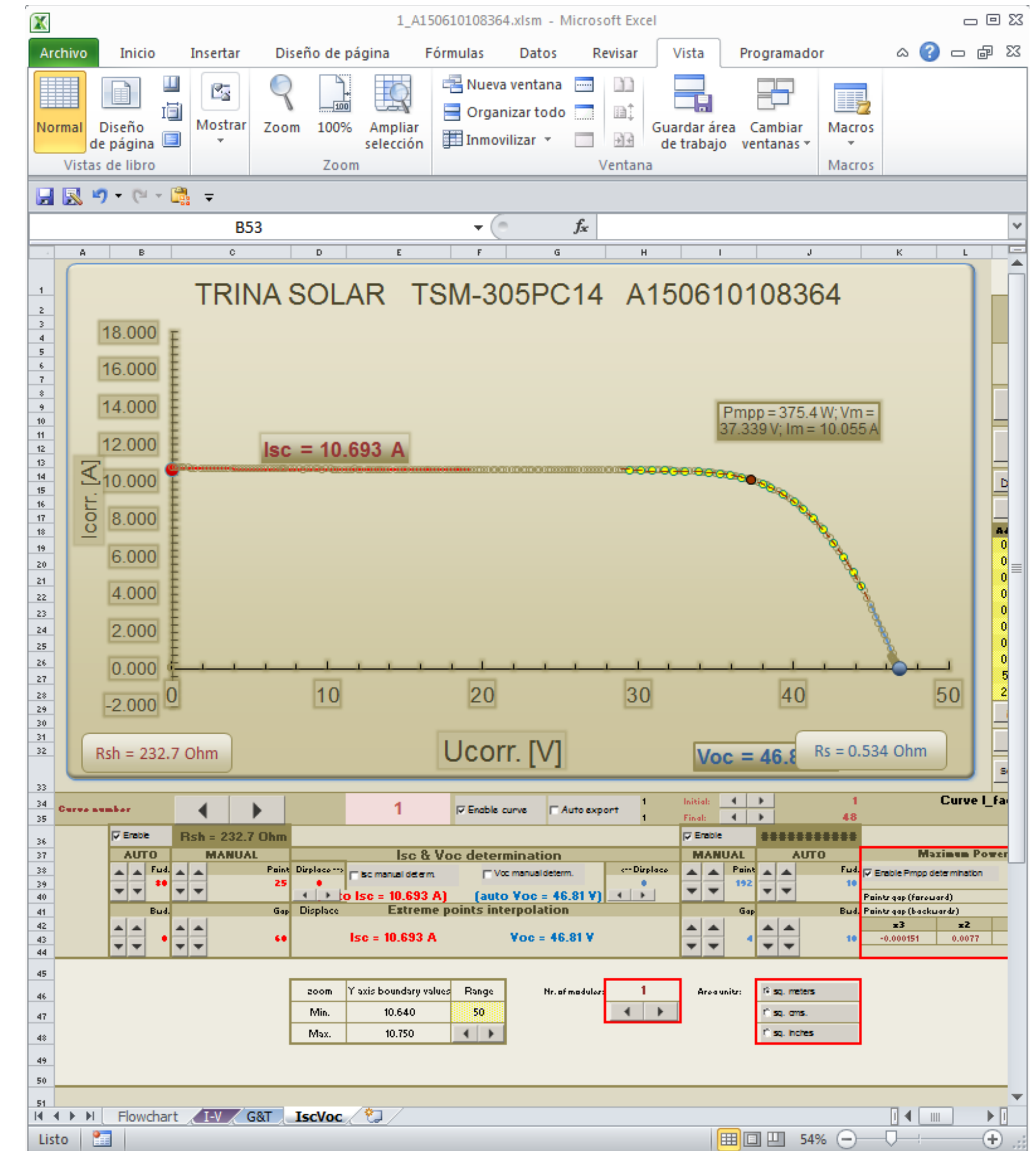
Ámbito de Trabajo: S.E. CALIBRACIÓN MÓDULOS FV, PROYECTO EVADIFOT, PROYECTO INNDISOL.

Necesidades: TRATAMIENTO DE DATOS DE MEDIDA DE CURVAS I-V DE MÓDULOS FV.

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: Si (3)
- Funciones Privadas (gestión de eventos): Si
- Funciones Públicas (módulos exportables ".bas"): Si (214 aprox.)
- Implementación de cálculos: celdas.

Capacidades: Interpolación, extrapolación y ajuste polinómico (automáticos), generación de tablas de resultados, exportación, ajuste a funciones teóricas, regresiones lineales y logarítmicas, funciones de filtrado en rangos variables, cálculo, visualización y almacenamiento de parámetros, etc.



Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación PVPOWER

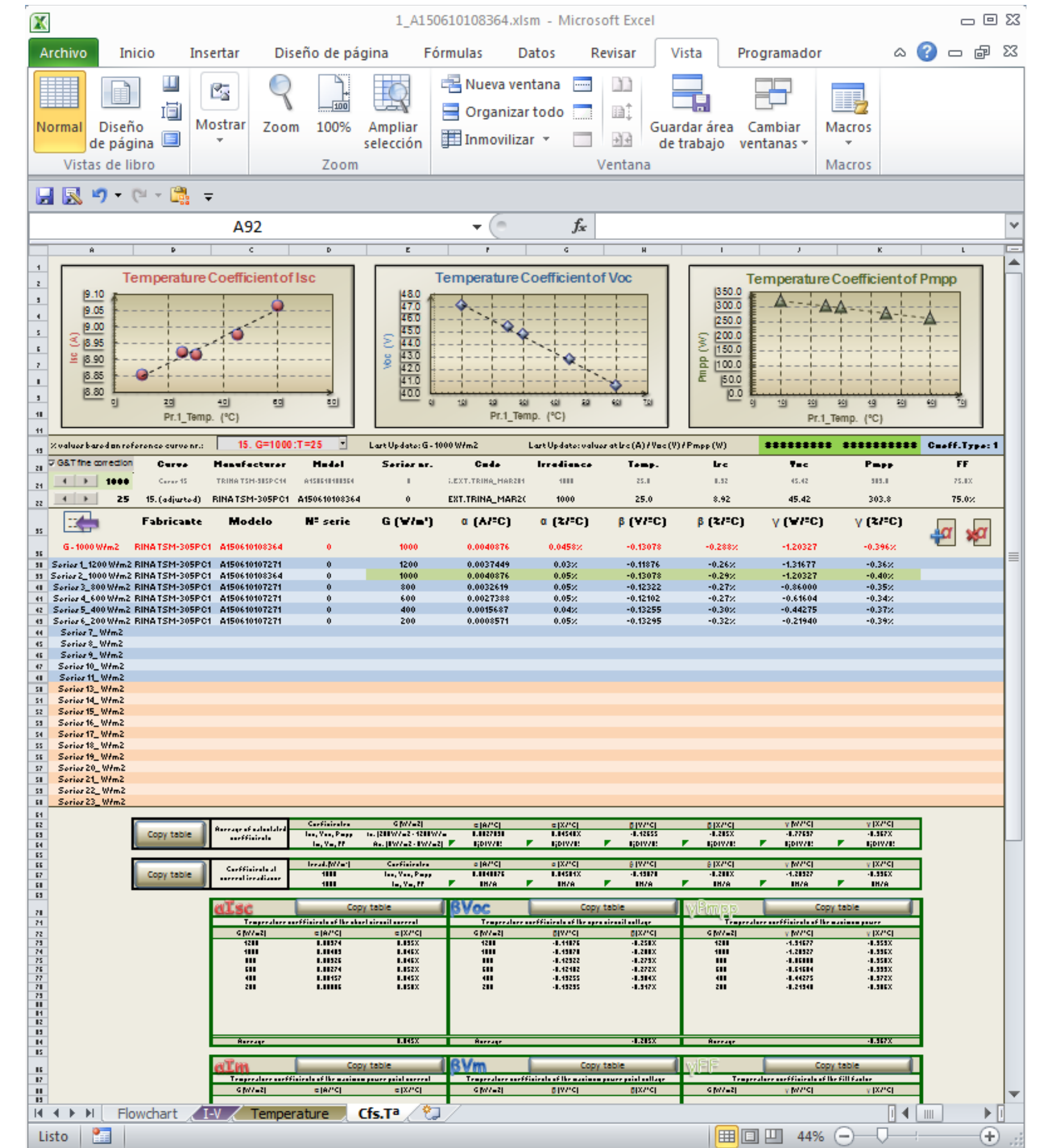
Ámbito de Trabajo: S.E. CALIBRACIÓN MÓDULOS FV, PROYECTO EVADIFOT, PROYECTO INNDISOL.

Necesidades: TRATAMIENTO DE DATOS DE MEDIDA DE CURVAS I-V DE MÓDULOS FV.

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: Si (3)
- Funciones Privadas (gestión de eventos): Si
- Funciones Públicas (módulos exportables ".bas"): Si (214 aprox.)
- Implementación de cálculos: celdas.

Capacidades: Interpolación, extrapolación y ajuste polinómico (automáticos), generación de tablas de resultados, exportación, ajuste a funciones teóricas, regresiones lineales y logarítmicas, funciones de filtrado en rangos variables, cálculo, visualización y almacenamiento de parámetros, etc.



Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación PVPOWER

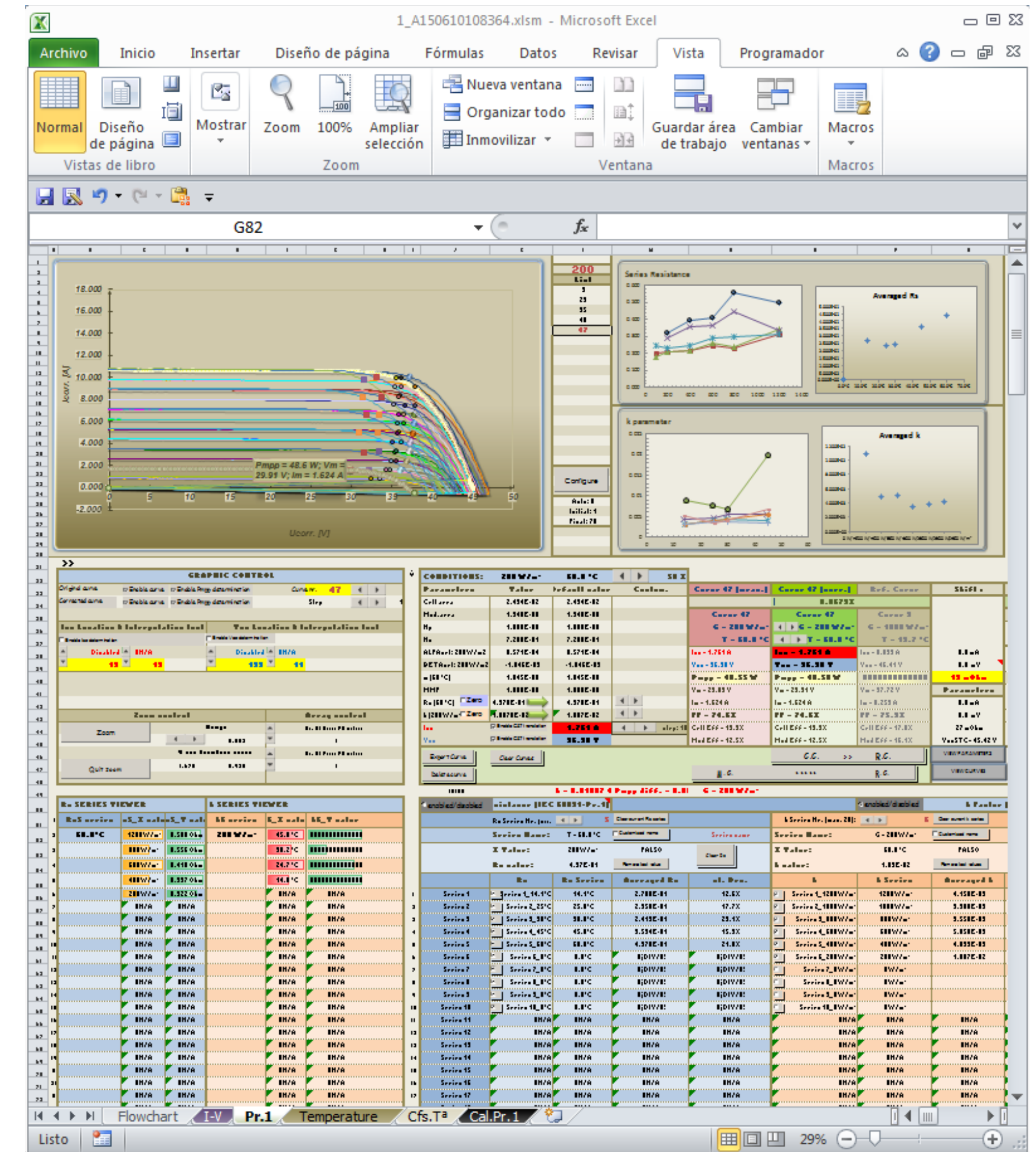
Ámbito de Trabajo: S.E. CALIBRACIÓN MÓDULOS FV, PROYECTO EVADIFOT, PROYECTO INNDISOL.

Necesidades: TRATAMIENTO DE DATOS DE MEDIDA DE CURVAS I-V DE MÓDULOS FV.

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: Si (3)
- Funciones Privadas (gestión de eventos): Si
- Funciones Públicas (módulos exportables ".bas"): Si (214 aprox.)
- Implementación de cálculos: celdas.

Capacidades: Interpolación, extrapolación y ajuste polinómico (automáticos), generación de tablas de resultados, exportación, ajuste a funciones teóricas, regresiones lineales y logarítmicas, funciones de filtrado en rangos variables, cálculo, visualización y almacenamiento de parámetros, etc.



Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación PVPOWER

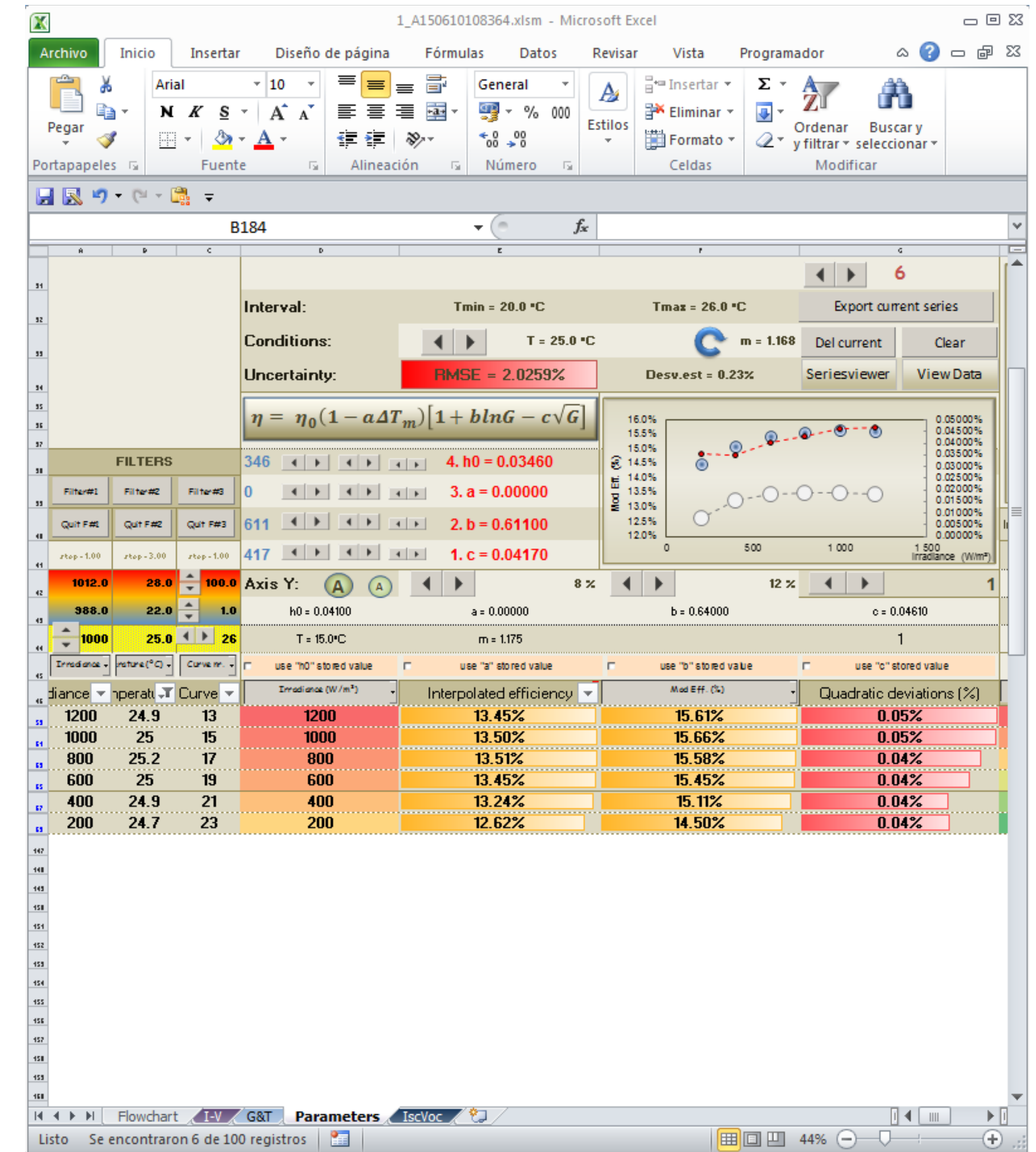
Ámbito de Trabajo: S.E. CALIBRACIÓN MÓDULOS FV, PROYECTO EVADIFOT, PROYECTO INNDISOL.

Necesidades: TRATAMIENTO DE DATOS DE MEDIDA DE CURVAS I-V DE MÓDULOS FV.

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: Si (3)
- Funciones Privadas (gestión de eventos): Si
- Funciones Públicas (módulos exportables ".bas"): Si (214 aprox.)
- Implementación de cálculos: celdas.

Capacidades: Interpolación, extrapolación y ajuste polinómico (automáticos), generación de tablas de resultados, exportación, ajuste a funciones teóricas, regresiones lineales y logarítmicas, funciones de filtrado en rangos variables, cálculo, visualización y almacenamiento de parámetros, etc.



Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación PVPOWER

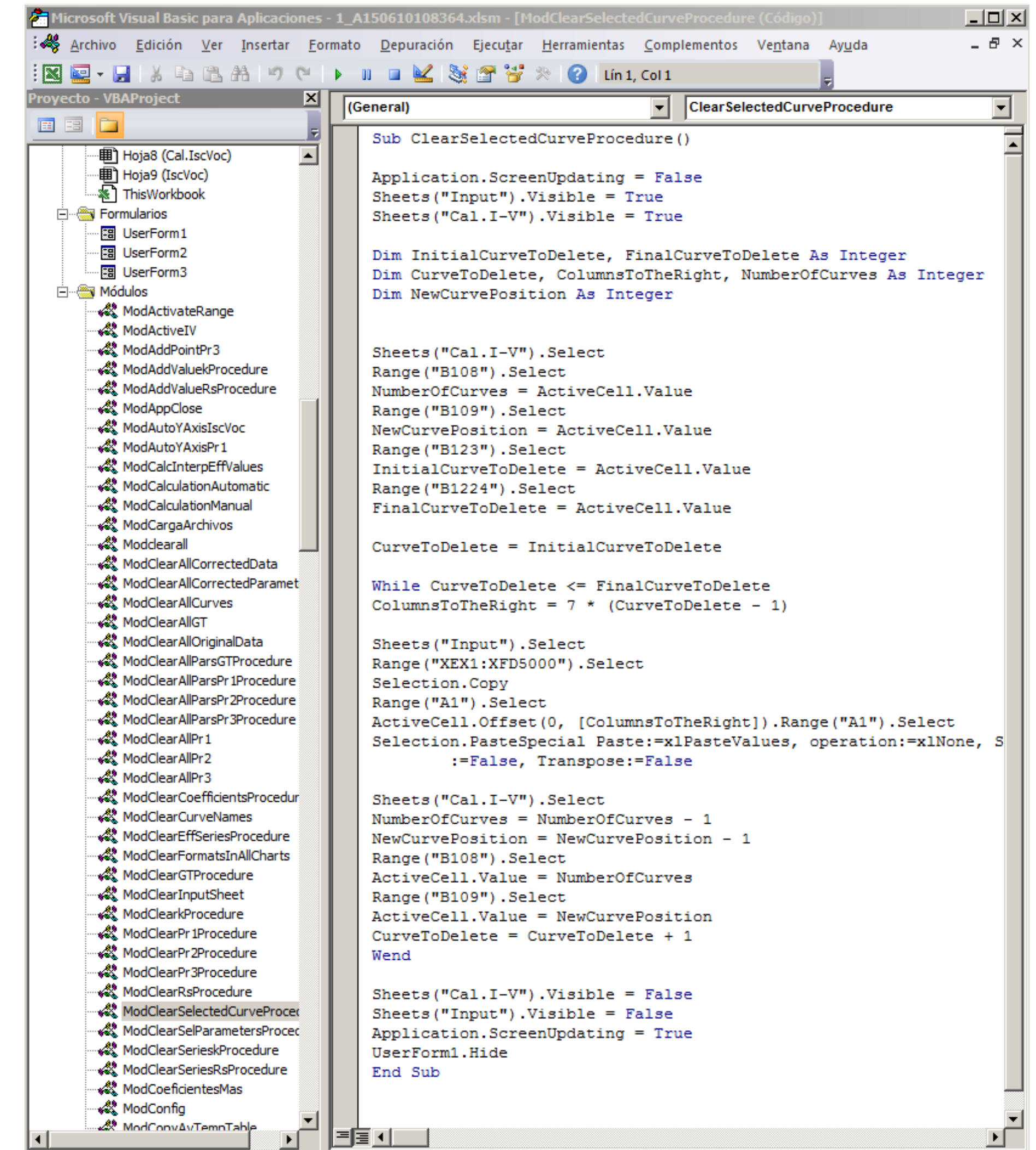
Ámbito de Trabajo: S.E. CALIBRACIÓN MÓDULOS FV, PROYECTO EVADIFOT, PROYECTO INNDISOL.

Necesidades: TRATAMIENTO DE DATOS DE MEDIDA DE CURVAS I-V DE MÓDULOS FV.

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: Si (3)
- Funciones Privadas (gestión de eventos): Si
- Funciones Públicas (módulos exportables ".bas"): Si (214 aprox.)
- Implementación de cálculos: celdas.

Capacidades: Interpolación, extrapolación y ajuste polinómico (automáticos), generación de tablas de resultados, exportación, ajuste a funciones teóricas, regresiones lineales y logarítmicas, funciones de filtrado en rangos variables, cálculo, visualización y almacenamiento de parámetros, etc.



Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación PVPOWER

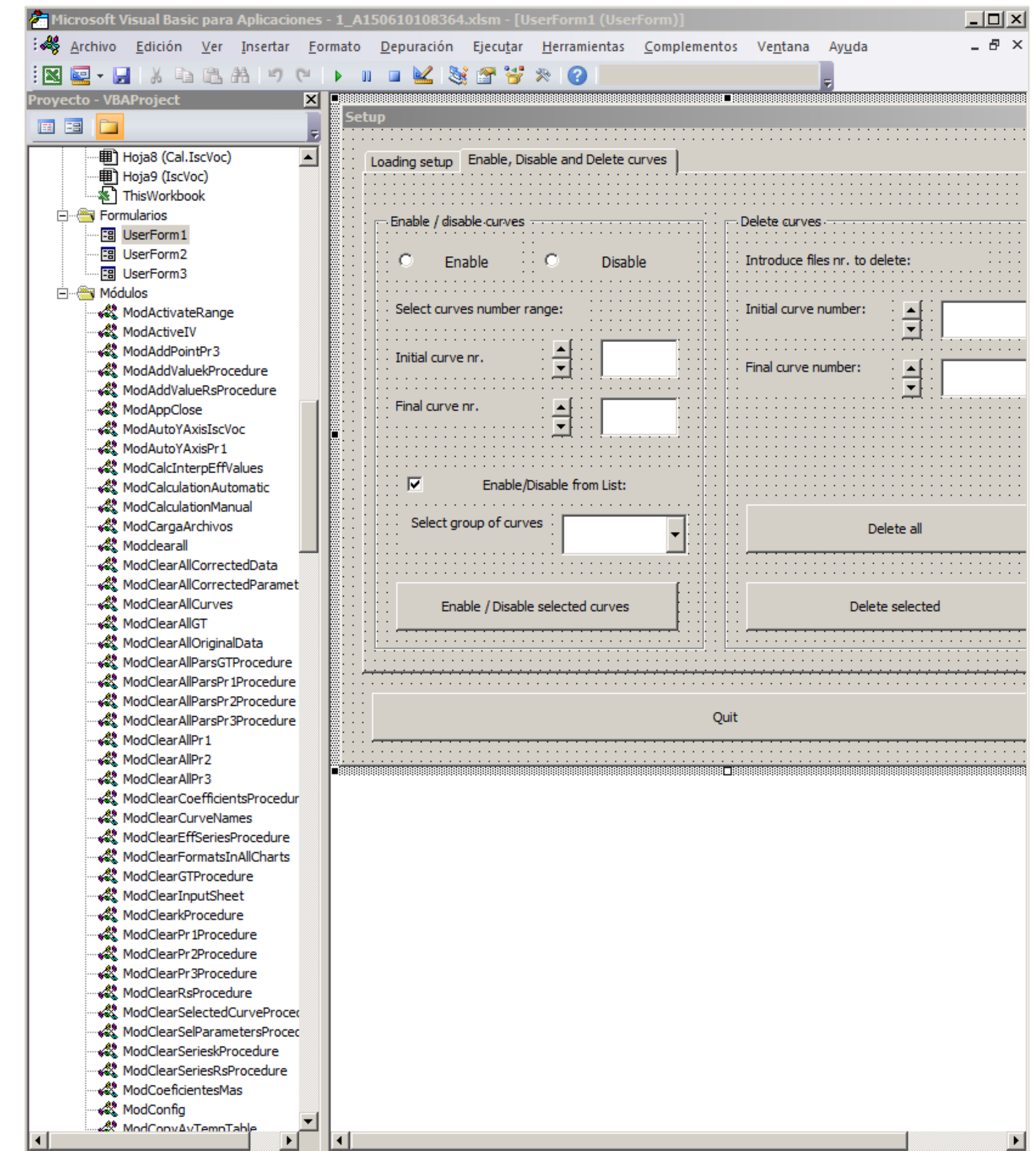
Ámbito de Trabajo: S.E. CALIBRACIÓN MÓDULOS FV, PROYECTO EVADIFOT, PROYECTO INNDISOL.

Necesidades: TRATAMIENTO DE DATOS DE MEDIDA DE CURVAS I-V DE MÓDULOS FV.

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: Si (3)
- Funciones Privadas (gestión de eventos): Si
- Funciones Públicas (módulos exportables ".bas"): Si (214 aprox.)
- Implementación de cálculos: celdas.

Capacidades: Interpolación, extrapolación y ajuste polinómico (automáticos), generación de tablas de resultados, exportación, ajuste a funciones teóricas, regresiones lineales y logarítmicas, funciones de filtrado en rangos variables, cálculo, visualización y almacenamiento de parámetros, etc.



Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación CALENER

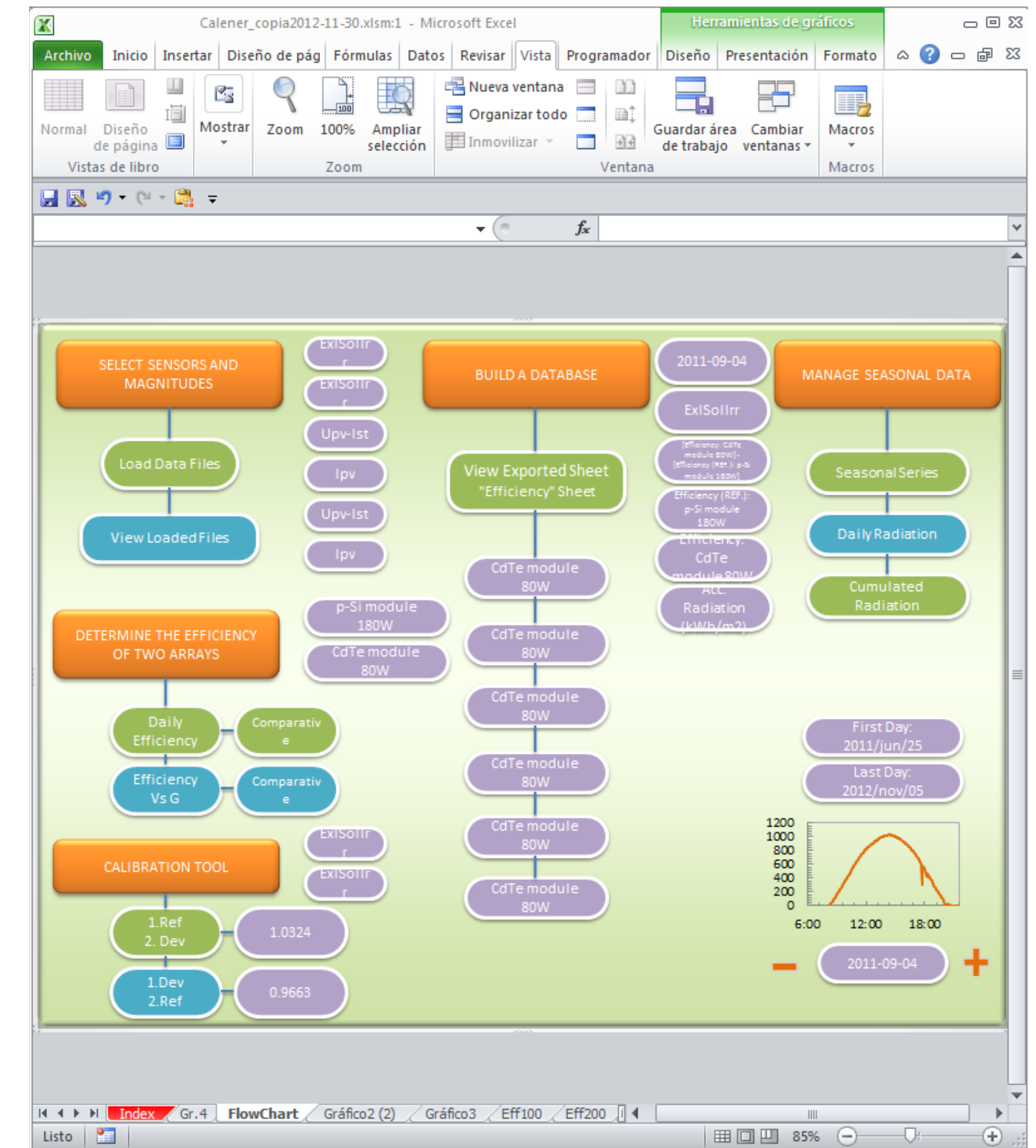
Ámbito de Trabajo: PROYECTO CALENER.

Necesidades: TRATAMIENTO DE DATOS DE MONITORIZACIÓN DE INSTALACIONES FV (degradación a medio plazo).

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: Si (1)
- Funciones Privadas (gestión de eventos): No
- Funciones Públicas (módulos exportables ".bas"): Si (9)
- Implementación de cálculos: celdas.

Capacidades: Automatización de la importación de archivos numéricos (.csv), construcción de vínculos *ad hoc* (independencia datos fuente Vs. aplicación), extracción de parámetros, cálculo, almacenamiento y visualización de resultados.



Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación CALENER

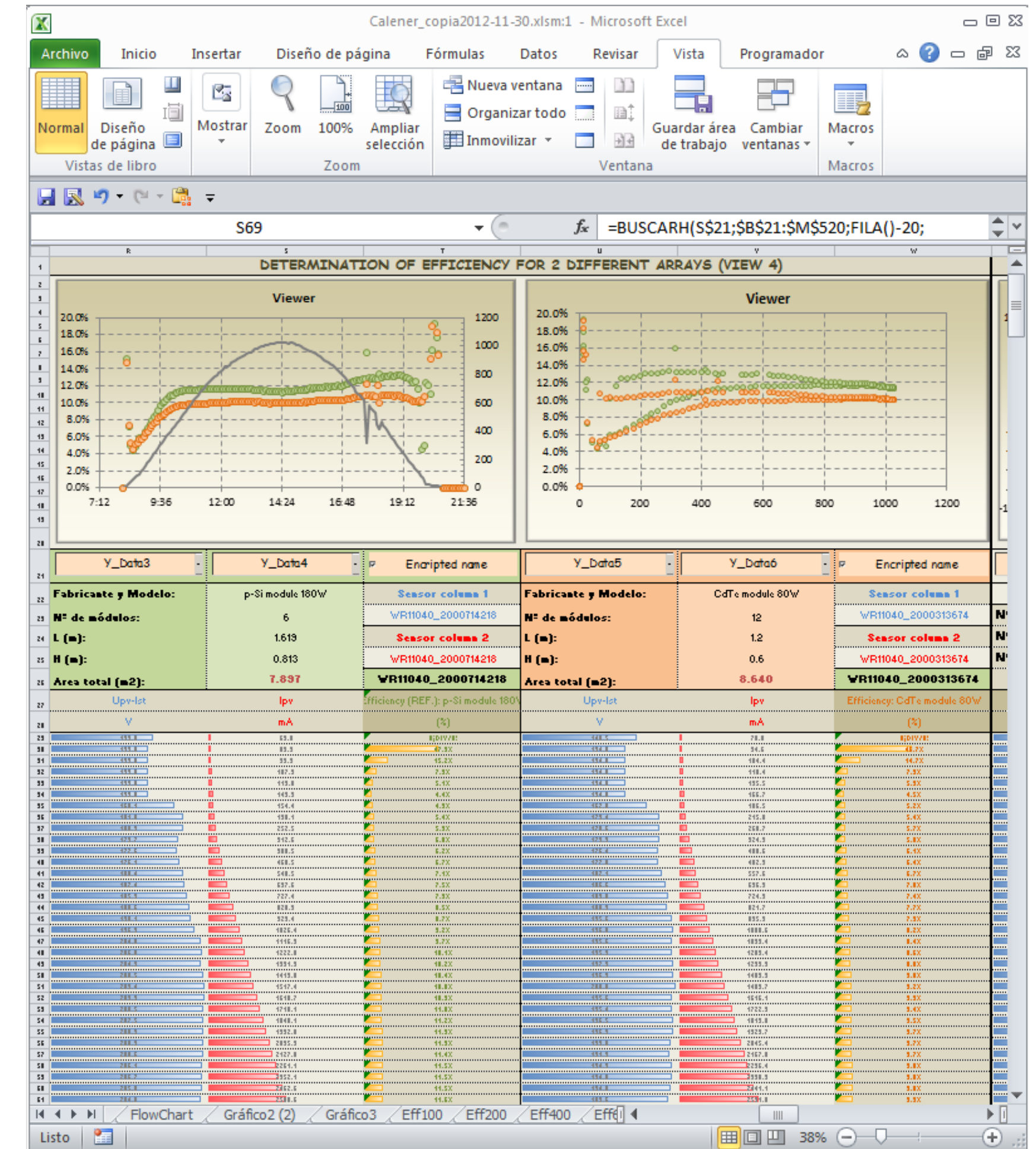
Ámbito de Trabajo: PROYECTO CALENER.

Necesidades: TRATAMIENTO DE DATOS DE MONITORIZACIÓN DE INSTALACIONES FV (degradación a medio plazo).

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: Si (1)
- Funciones Privadas (gestión de eventos): No
- Funciones Públicas (módulos exportables ".bas"): Si (9)
- Implementación de cálculos: celdas.

Capacidades: Automatización de la importación de archivos numéricos (.csv), construcción de vínculos *ad hoc* (independencia datos fuente Vs. aplicación), extracción de parámetros, cálculo, almacenamiento y visualización de resultados.



Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación CALENER

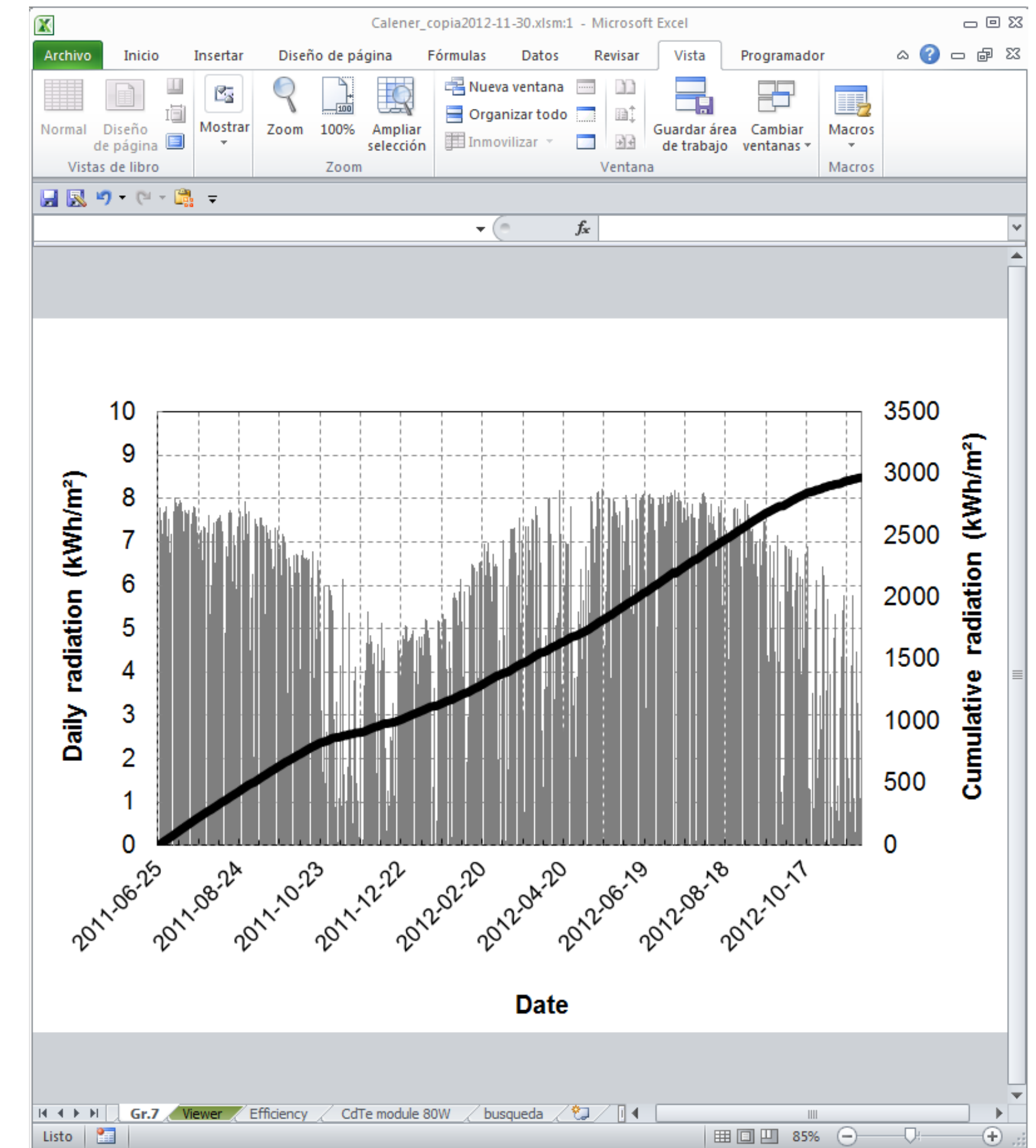
Ámbito de Trabajo: PROYECTO CALENER.

Necesidades: TRATAMIENTO DE DATOS DE MONITORIZACIÓN DE INSTALACIONES FV (degradación a medio plazo).

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: Si (1)
- Funciones Privadas (gestión de eventos): No
- Funciones Públicas (módulos exportables ".bas"): Si (9)
- Implementación de cálculos: celdas.

Capacidades: Automatización de la importación de archivos numéricos (.csv), construcción de vínculos *ad hoc* (independencia datos fuente Vs. aplicación), extracción de parámetros, cálculo, almacenamiento y visualización de resultados.



Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación CALENER

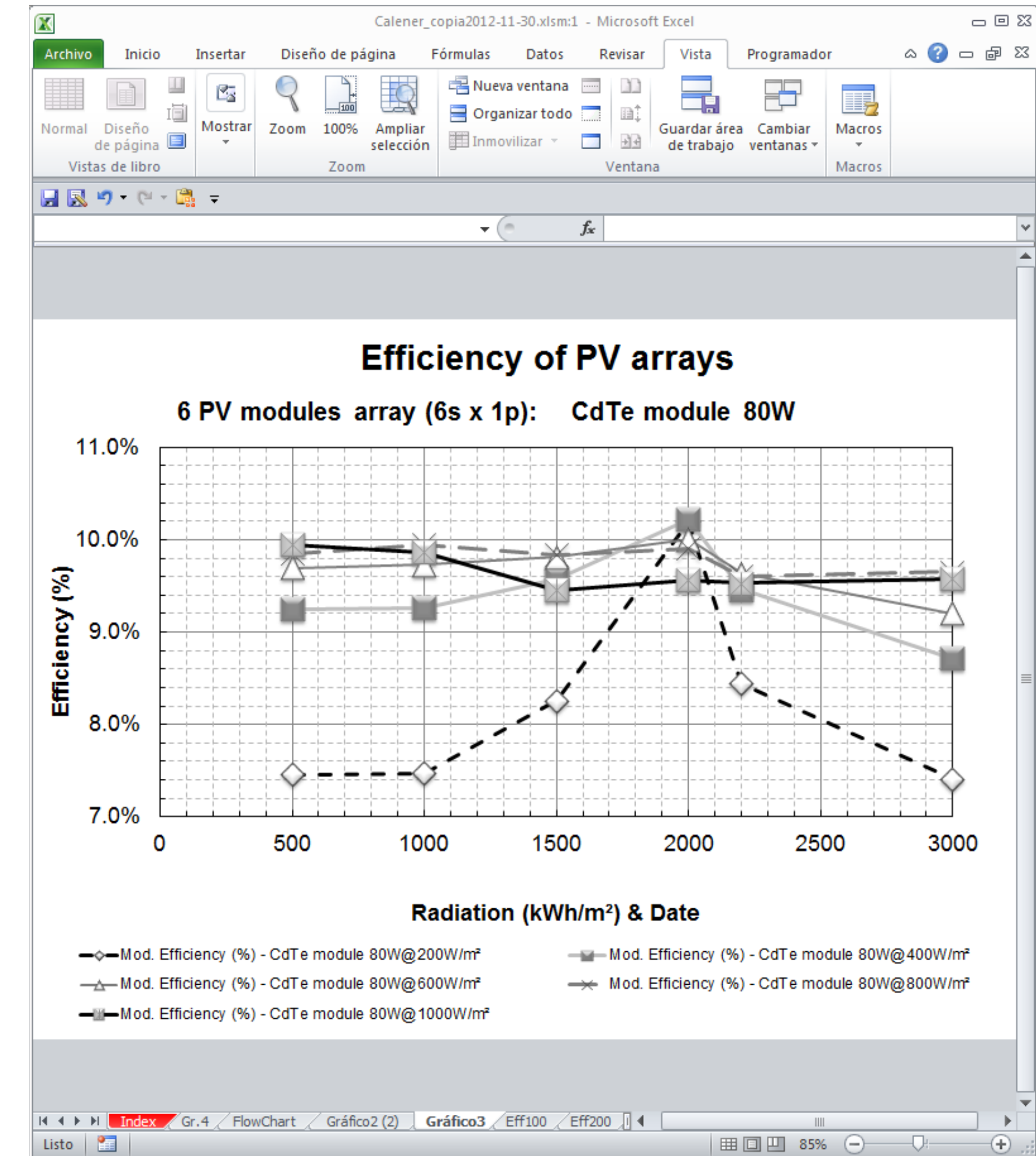
Ámbito de Trabajo: PROYECTO CALENER.

Necesidades: TRATAMIENTO DE DATOS DE MONITORIZACIÓN DE INSTALACIONES FV (degradación a medio plazo).

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: Si (1)
- Funciones Privadas (gestión de eventos): No
- Funciones Públicas (módulos exportables ".bas"): Si (9)
- Implementación de cálculos: celdas.

Capacidades: Automatización de la importación de archivos numéricos (.csv), construcción de vínculos *ad hoc* (independencia datos fuente Vs. aplicación), extracción de parámetros, cálculo, almacenamiento y visualización de resultados.



Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación CALENER

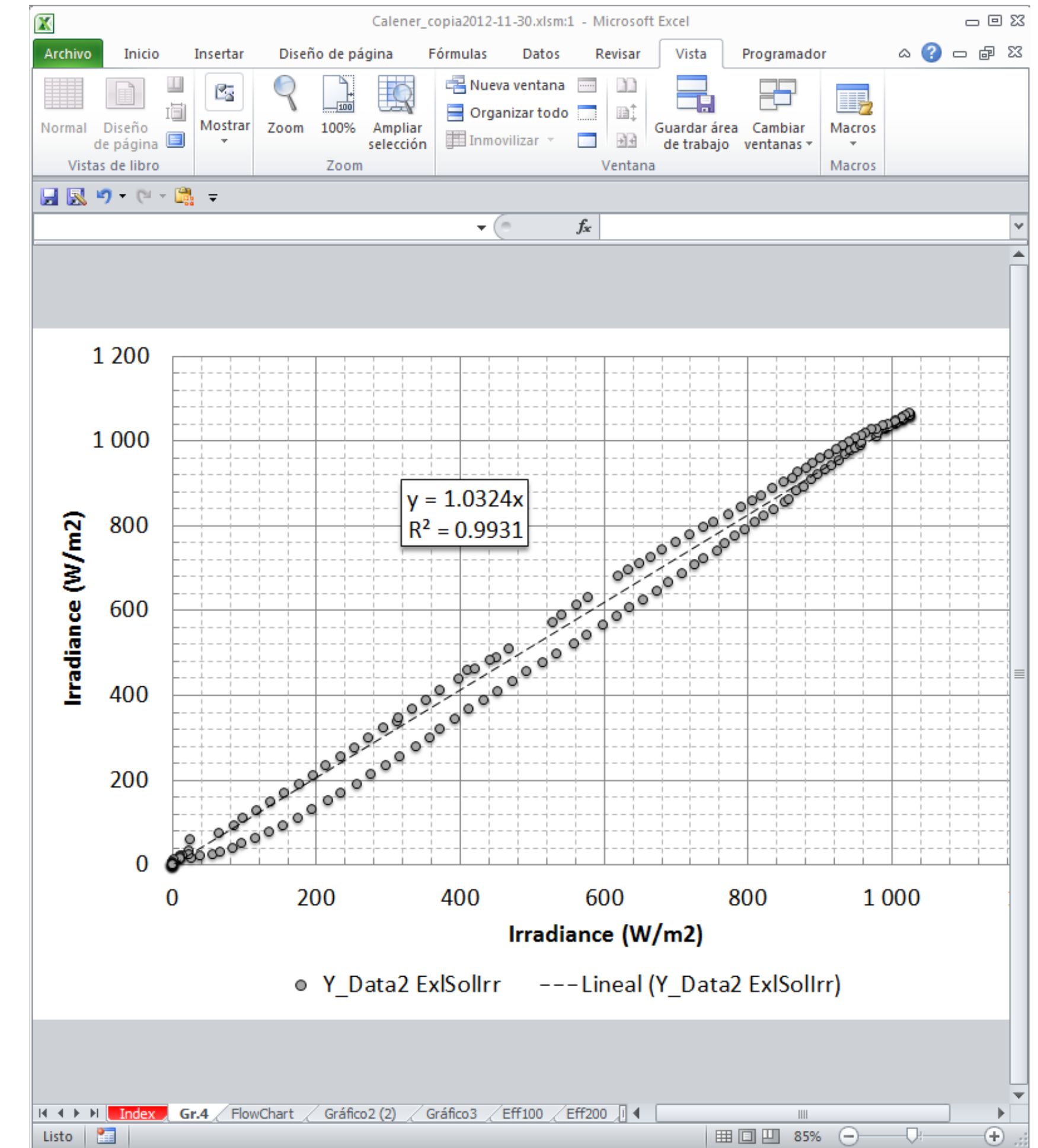
Ámbito de Trabajo: PROYECTO CALENER.

Necesidades: TRATAMIENTO DE DATOS DE MONITORIZACIÓN DE INSTALACIONES FV (degradación a medio plazo).

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: Si (1)
- Funciones Privadas (gestión de eventos): No
- Funciones Públicas (módulos exportables ".bas"): Si (9)
- Implementación de cálculos: celdas.

Capacidades: Automatización de la importación de archivos numéricos (.csv), construcción de vínculos *ad hoc* (independencia datos fuente Vs. aplicación), extracción de parámetros, cálculo, almacenamiento y visualización de resultados.



Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación SENSORS (versión 2013)

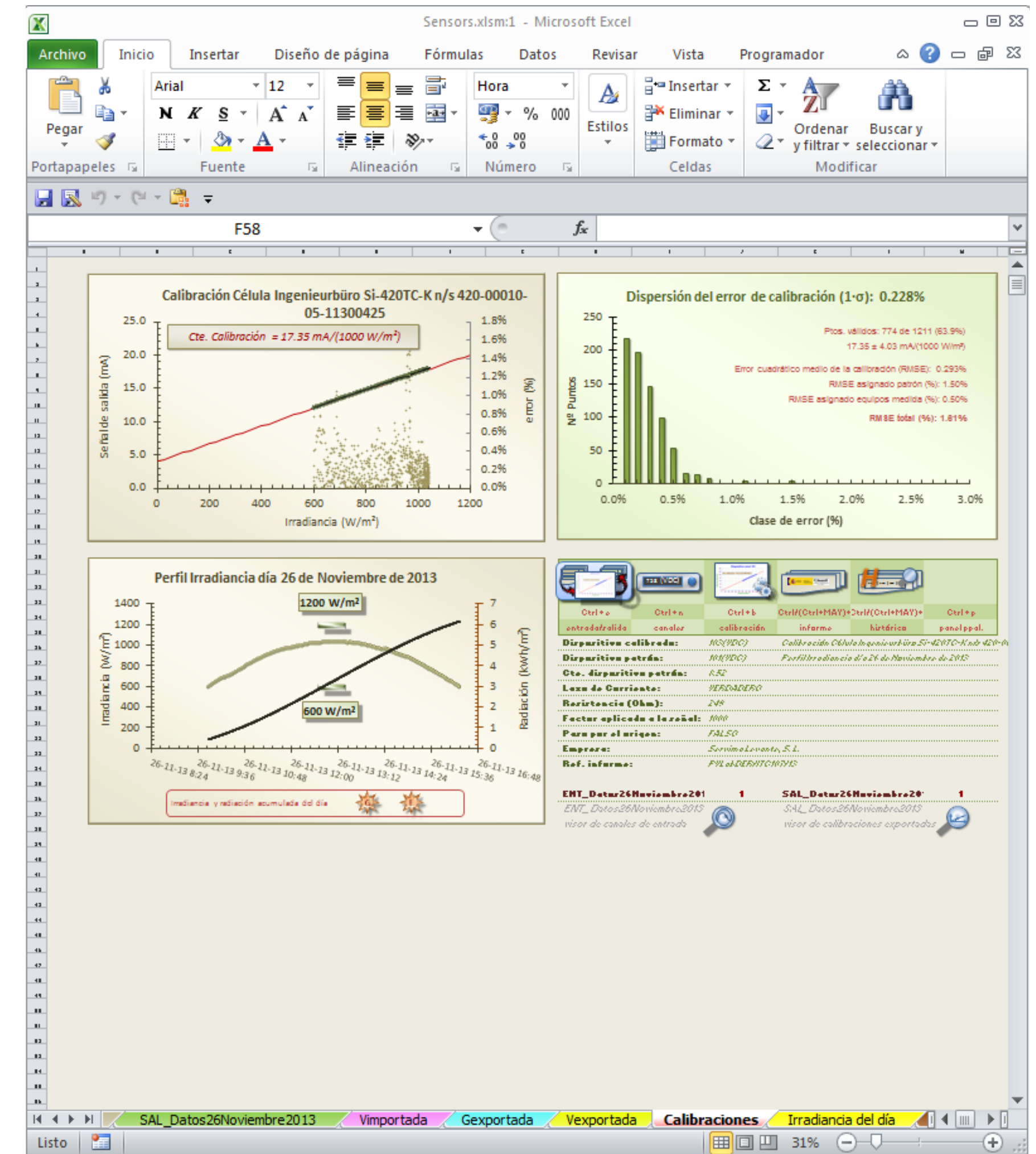
Ámbito de Trabajo: S.E. CALIBRACIÓN PIRANÓMETROS Y PIRHELIÓMETROS.

Necesidades: TRATAMIENTO DE DATOS DE MEDIDA DE SENSORES DE IRRADIANCIA SOLAR.

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: Si (9)
- Funciones Privadas (gestión de eventos): Si
- Funciones Públicas (módulos exportables ".bas"): Si (131)
- Implementación de cálculos: celdas / código VB.

Capacidades: Automatización: I/O archivos medida, configuración campos, gráficos de resultados, condiciones de medida, histograma de error, etc., generación de texto (resultados), tabla acumulativa (histórico), valores acumulados (radiación), etc., automatización de la *interface* gráfica (gestión ventanas y pantallas).



Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación SENSORS (versión 2013)

Ámbito de Trabajo: S.E. CALIBRACIÓN PIRANÓMETROS Y PIRHELIÓMETROS.

Necesidades: TRATAMIENTO DE DATOS DE MEDIDA DE SENSORES DE IRRADIANCIA SOLAR.

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: Si (9)
- Funciones Privadas (gestión de eventos): Si
- Funciones Públicas (módulos exportables ".bas"): Si (131)
- Implementación de cálculos: celdas / código VB.

Capacidades: Automatización: I/O archivos medida, configuración campos, gráficos de resultados, condiciones de medida, histograma de error, etc., generación de texto (resultados), tabla acumulativa (histórico), valores acumulados (radiación), etc., automatización de la *interface* gráfica (gestión ventanas y pantallas).

Control de los parámetros de la calibración

Control y Filtrado de Datos | Resumen de la Calibración | Información Adicional | Acciones

Canal Patrón: Perfil Irradiancia día 26 de Noviembre de 2013

Canal a Calibrar: Calibración Célula Ingenieurbüro Si-420TC-K n/s 420-00010-05-11300425

Canal activo: 103(VDC)

Descripción	Valor
Cte. Calibración del Patrón (mV/kW/m²):	8.52
Factor de escala de la señal:	1000
☒ Lazo de Corriente (Ohm):	249
☐ Err. max. (%):	1.5
☐ Paso por origen	

Filtro	Valor	Filtro	Valor
G min. (W/m²):	600	G max. (W/m²):	1200

Filtro	Medida Nº	Fecha/Hora	G (W/m²)	Go (kWh/m²)
☐ Pto. Inicial	2	26/11/2013 7:57:06	0.3	0
☐ Pto. Final	1211	26/11/2013 18:02:06	0.2	6.5

Aplicar Cambios Exportar calibración

Salir

Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación SENSORS (versión 2013)

Ámbito de Trabajo: S.E. CALIBRACIÓN PIRANÓMETROS Y PIRHELIÓMETROS.

Necesidades: TRATAMIENTO DE DATOS DE MEDIDA DE SENSORES DE IRRADIANCIA SOLAR.

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: Si (9)
- Funciones Privadas (gestión de eventos): Si
- Funciones Públicas (módulos exportables ".bas"): Si (131)
- Implementación de cálculos: celdas / código VB.

Capacidades: Automatización: I/O archivos medida, configuración campos, gráficos de resultados, condiciones de medida, histograma de error, etc., generación de texto (resultados), tabla acumulativa (histórico), valores acumulados (radiación), etc., automatización de la *interface* gráfica (gestión ventanas y pantallas).

The screenshot shows a software window titled "Control de los parámetros de la calibración". It has four tabs: "Control y Filtrado de Datos", "Resumen de la Calibración", "Información Adicional", and "Acciones". The "Resumen de la Calibración" tab is active, displaying the following information:

- Se ha calibrado el dispositivo **Calibración Célula Ingenieurbüro Si-420TC-K n/s**
- por comparación con **420-00010-05-11300425**
- de constante de calibración **8.52 mV/(1000 W/m²)**
- entre las fechas **26/11/2013 7:56:36** y **26/11/2013 18:02:06**,
- habiéndose registrado un número de puntos válidos igual a **774** de un total de **1211**.

Below this, there is a section for the calibration constant:

Constante de Calibración

17.35 ± 4.03 mA/(1000 W/m²)

Further down, it shows:

- El valor de la pendiente es **0.013343**
- y la ordenada en el origen **4.001878**.
- La desviación estándar (SIGMA) de la calibración ha sido **0.228%**
- y el error cuadrático medio (RMSE) **0.293%**.

At the bottom of the window, there is a "Salir" button.

Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación SENSORS (versión 2013)

Ámbito de Trabajo: S.E. CALIBRACIÓN PIRANÓMETROS Y PIRHELIÓMETROS.

Necesidades: TRATAMIENTO DE DATOS DE MEDIDA DE SENSORES DE IRRADIANCIA SOLAR.

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: Si (9)
- Funciones Privadas (gestión de eventos): Si
- Funciones Públicas (módulos exportables ".bas"): Si (131)
- Implementación de cálculos: celdas / código VB.

Capacidades: Automatización: I/O archivos medida, configuración campos, gráficos de resultados, condiciones de medida, histograma de error, etc., generación de texto (resultados), tabla acumulativa (histórico), valores acumulados (radiación), etc., automatización de la *interface* gráfica (gestión ventanas y pantallas).

Control de los parámetros de la calibración

Control y Filtrado de Datos | Resumen de la Calibración | Información Adicional | Acciones

DATOS DE LA EMPRESA

Empresa: Servima Levante, S. L.
Calle: C/ Charles Darwin, 20.
Polígono industrial: Polígono Industrial Valdeacederas
Población: C.P. 46980. Paterna, Valencia.
Persona de contacto: Att: D. José Sotillo Toca
Teléfono de contacto: Tel. 961318003/Fax 961318344
Correo electrónico de contacto: sotillo@nobesol.com

DATOS DE LA CALIBRACIÓN

MUESTRA A MEDIR
Descripción: Sensor fotovoltaico
Marca: TRITEC
Modelo: SPECTRON 200
Identificación/Nº de serie: X00511001875
Constante nominal de calibración: 71.0 mV/(1000 W/m²)
Dispositivo calibrado: Calibración Célula Ingenieurbüro Si-420TC-K n/s 420-01
Dispositivo patrón: Perfil Irradiancia día 26 de Noviembre de 2013
Cte. dispositivo patrón: 8.52 mV/(1000 W/m²)
Lazo de Corriente: VERDADERO
Resistencia (Ohm): 249.000
Inv. Resistencia: 0.004

Unidades de la constante: mV/(1000 W/m²) Nº decimales: 2

Rótulo Eje X: Irradiancia (W/m²) Rótulo Eje Y: Señal de salida (mV)

Guardar preferencias

Salir

Listo

Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación SENSORS (versión 2013)

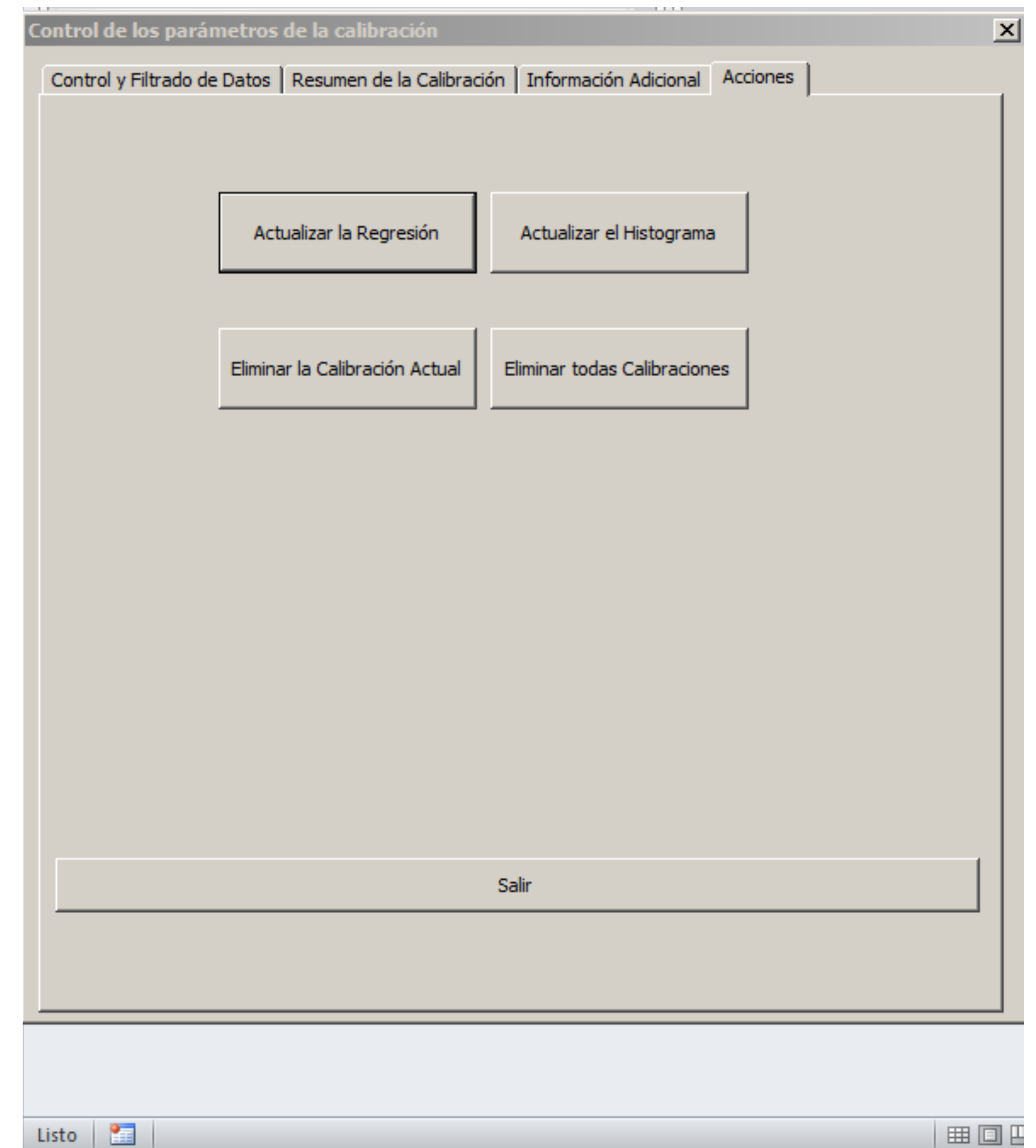
Ámbito de Trabajo: S.E. CALIBRACIÓN PIRANÓMETROS Y PIRHELIÓMETROS.

Necesidades: TRATAMIENTO DE DATOS DE MEDIDA DE SENSORES DE IRRADIANCIA SOLAR.

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: Si (9)
- Funciones Privadas (gestión de eventos): Si
- Funciones Públicas (módulos exportables ".bas"): Si (131)
- Implementación de cálculos: celdas / código VB.

Capacidades: Automatización: I/O archivos medida, configuración campos, gráficos de resultados, condiciones de medida, histograma de error, etc., generación de texto (resultados), tabla acumulativa (histórico), valores acumulados (radiación), etc., automatización de la *interface* gráfica (gestión ventanas y pantallas).



Desarrollo de Software Científico - Técnico

Antecedentes

Aplicación SENSORS (versión 2013)

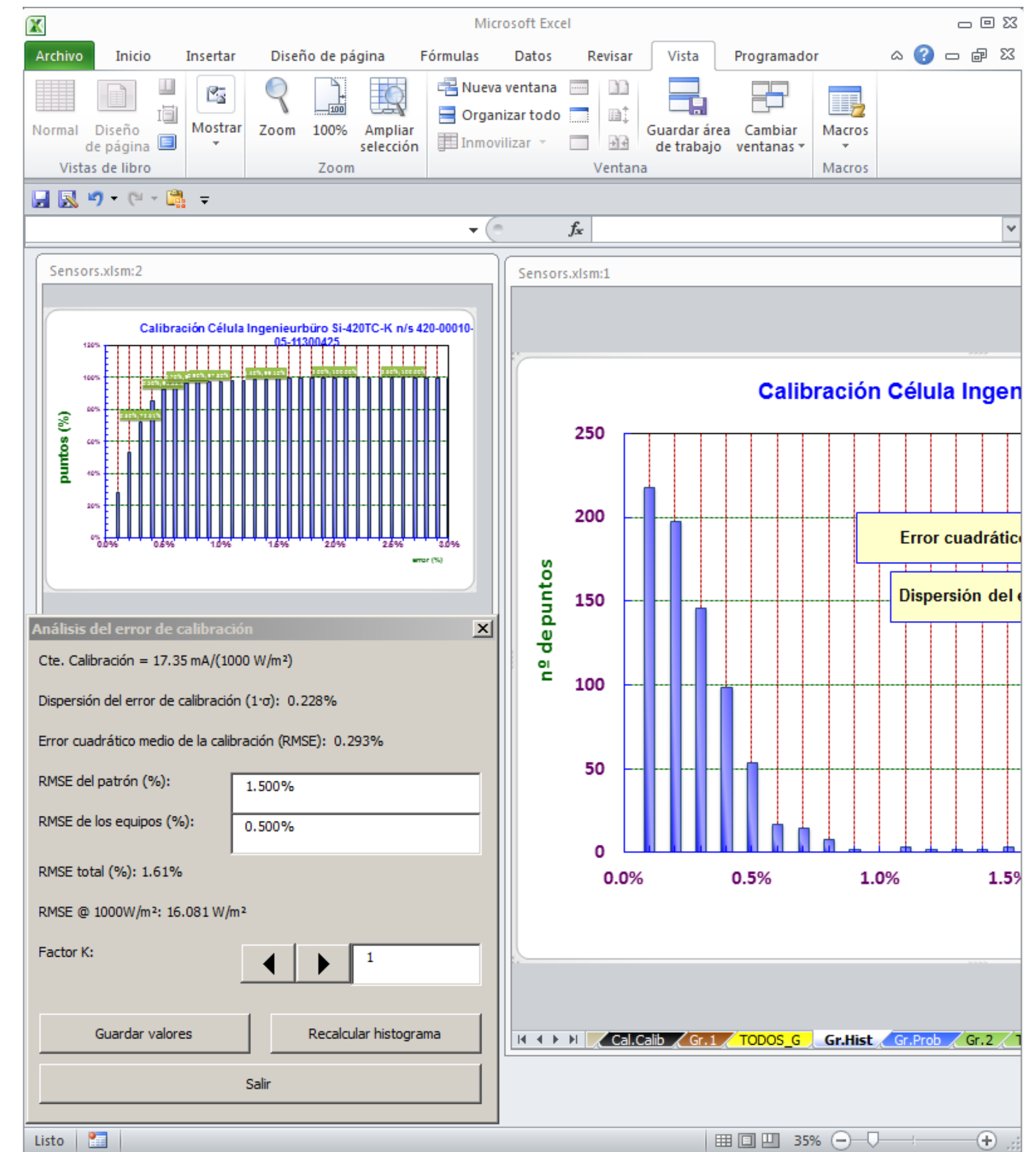
Ámbito de Trabajo: S.E. CALIBRACIÓN PIRANÓMETROS Y PIRHELIÓMETROS.

Necesidades: TRATAMIENTO DE DATOS DE MEDIDA DE SENSORES DE IRRADIANCIA SOLAR.

Plataforma: Microsoft Excel + Visual Basic para Aplicaciones (VBA):

- Controles ActiveX: Si
- Diseño/Controles de Formulario: Si (9)
- Funciones Privadas (gestión de eventos): Si
- Funciones Públicas (módulos exportables ".bas"): Si (131)
- Implementación de cálculos: celdas / código VB.

Capacidades: Automatización: I/O archivos medida, configuración campos, gráficos de resultados, condiciones de medida, histograma de error, etc., generación de texto (resultados), tabla acumulativa (histórico), valores acumulados (radiación), etc., automatización de la *interface* gráfica (gestión ventanas y pantallas).



(1 - 4/6) **Análisis de la Actividad y su Impacto**

Desarrollo de Software Científico - Técnico

Análisis

Por un lado, el trabajo realizado ha cumplido en cada ocasión con su cometido. Desafortunadamente, a pesar del esfuerzo empleado, en relación con su proyección externa, la actividad se caracteriza por:



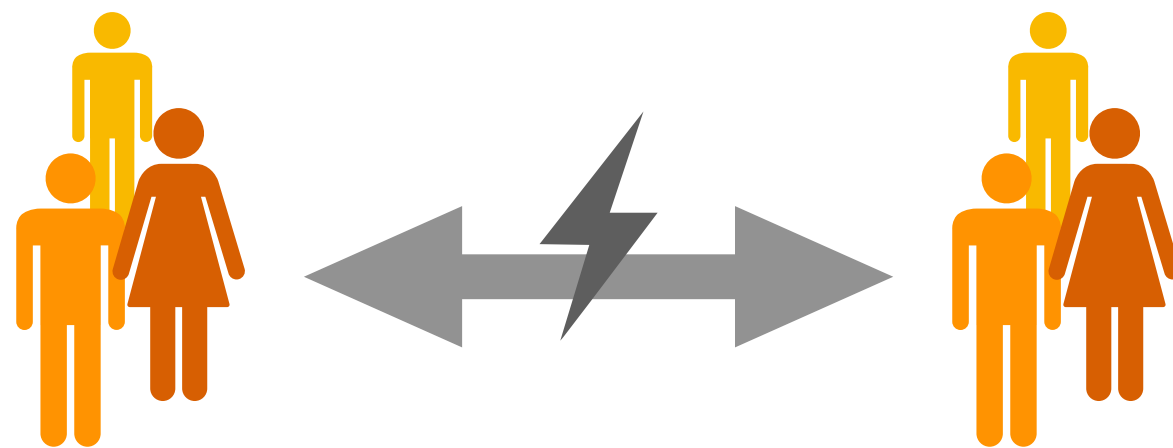
NO ATRIBUIBLE



INACCESIBLE



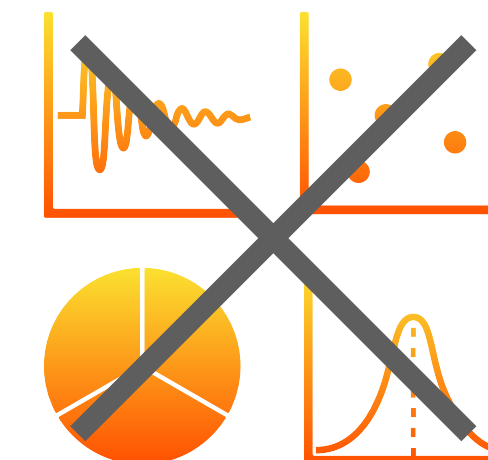
**¿ DIVULGABLE ?
¿ PUBLICABLE ?**



El desarrollo de software es una **capacidad implícita** que no genera un producto tangible y atribuible, y/o NO es una de las actividades atribuidas al laboratorio, por lo que no es fruto de otras colaboraciones, solicitudes, o participaciones, dentro o fuera del centro.



NO se realiza una actividad de desarrollo de software en acceso abierto, excluyéndonos de una potencial **comunidad de desarrolladores y usuarios interesados**; las herramientas generadas no tienen aplicación de cara al exterior.



El trabajo de software desarrollado adolece de muchas objeciones de cara a ser publicado. Por tanto, no ha sido sometido a peer review, no resulta **referenciable** y los avances realizados no encuentran **impacto** en la comunidad científica.

Desarrollo de Software Científico - Técnico

Análisis

Las cualidades estimadas como deseables para la actividad, *globalmente* tienden a generar **respuestas afirmativas** a las siguientes preguntas:



¿Genera un **producto estándar**?, ¿Tiene en cuenta los **criterios técnicos / buenas prácticas** del lenguaje y entorno global de trabajo?



¿**Integrada** en una cadena de valor?, ¿Aprovecha **recursos existentes** o esfuerzos anteriores?, ¿Tiene **aplicaciones** adicionales?



¿Orientada a la resolución de "**problemas universales**" o "generalizables"?



¿Orientada a permitir su empleo en potenciales trabajos o **actividades futuras**?



¿Se ha realizado un **proyecto documental**? ¿La documentación es **completa**? ¿Contiene **ejemplos válidos**?



¿Existe algún procedimiento para **evaluar los resultados** (eficacia, eficiencia, versatilidad, ...) de la actividad?, ¿Los resultados son **compatibles** tras modificaciones?



¿Está descrita en una **planificación** u **objetivos**?, ¿Se analizan los **recursos** necesarios?, ¿Se trabaja sobre el **diseño** y **desarrollo**?



¿Resultaría **inteligible** para una tercera persona? (¿Resulta inteligible para el propio autor pasado un tiempo?)



¿Resulta un producto **tangible** de la tarea?, ¿Puede constituir un **servicio**?, ¿Puede formar parte de un **proyecto**?, ¿Puede ser **publicable**?, ¿Puede ser objeto de **formación** interna/externa?, ¿Puede ser un **elemento atractor** para el laboratorio/grupo/centro?

Desarrollo de Software Científico - Técnico

Análisis de la actividad

En definitiva, la expectativa a futuro pretende ser cambiada por el siguiente:



El desarrollo de software es una **actividad explícita** que genera productos tangibles, con una atribución basada en referencias aceptadas (DOI). El laboratorio colabora, participa en proyectos, provee formación o realiza servicios de desarrollo de software, dentro o fuera del centro.



Se genera software en acceso abierto, el laboratorio participa en **comunidades de desarrolladores**, aprovechando recursos ajenos y a su vez creando herramientas que encuentran aplicación en la comunidad de usuarios (p.ej. PVLIB - NREL).



Los desarrollos de software realizados por el laboratorio pueden ser **referenciados como producción científica** a todos los efectos (particular a autores, grupo, proyectos, etc.), y los avances realizados encuentran **impacto** en la comunidad científica.

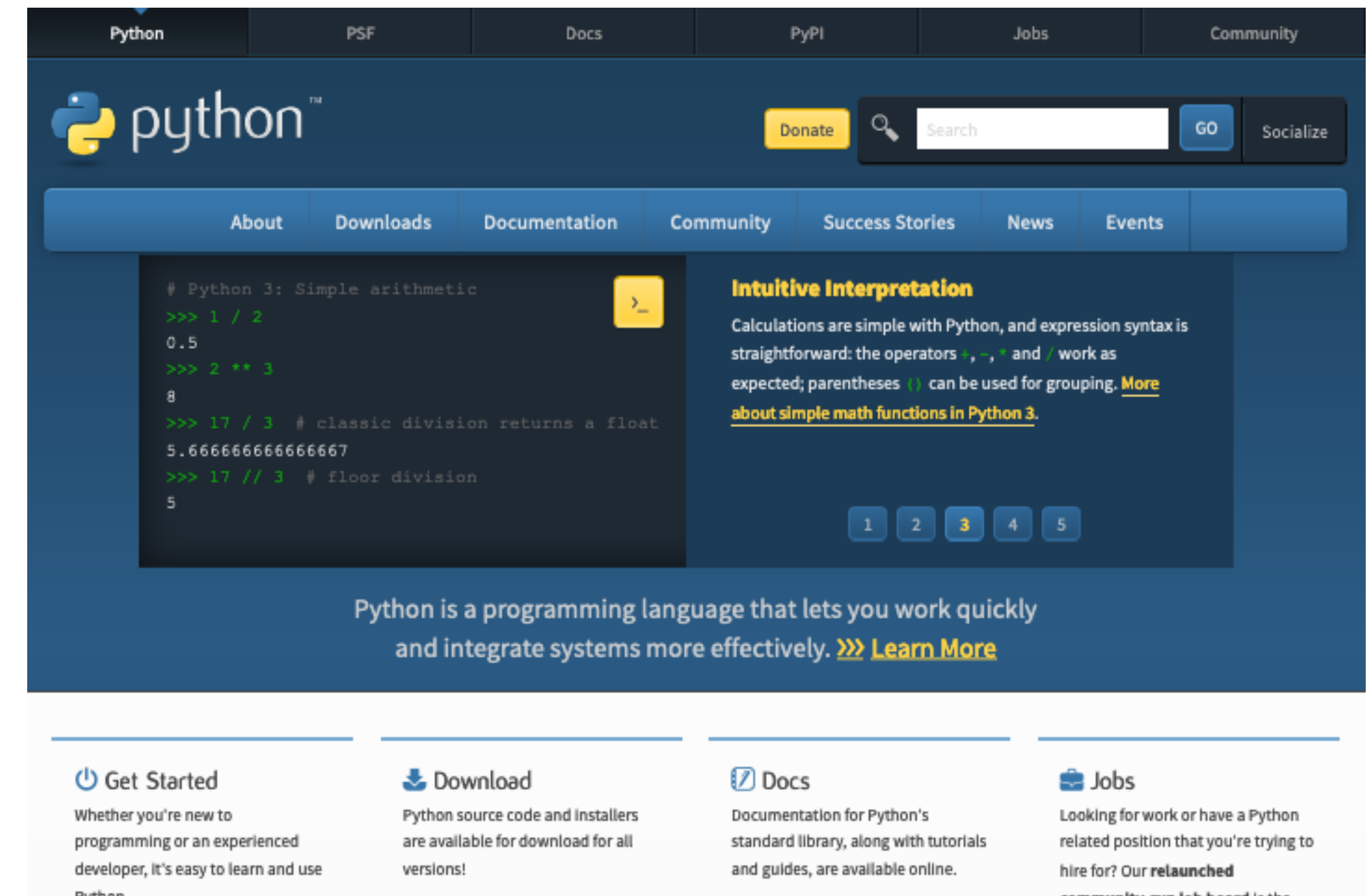
(I - 5/6) Evaluación del Escenario Actual

Desarrollo de Software Científico - Técnico

Escenario

Con la revolución de las nuevas tecnologías, en la última década se han producido una serie de avances disruptivos:

Los **lenguajes de programación de código abierto** (*Python*, ...), han alcanzado un importante grado de madurez y robustez: multi-plataforma, desarrollo colaborativo y coherente, guiado por entidades que promueven su desarrollo técnico y buenas prácticas.

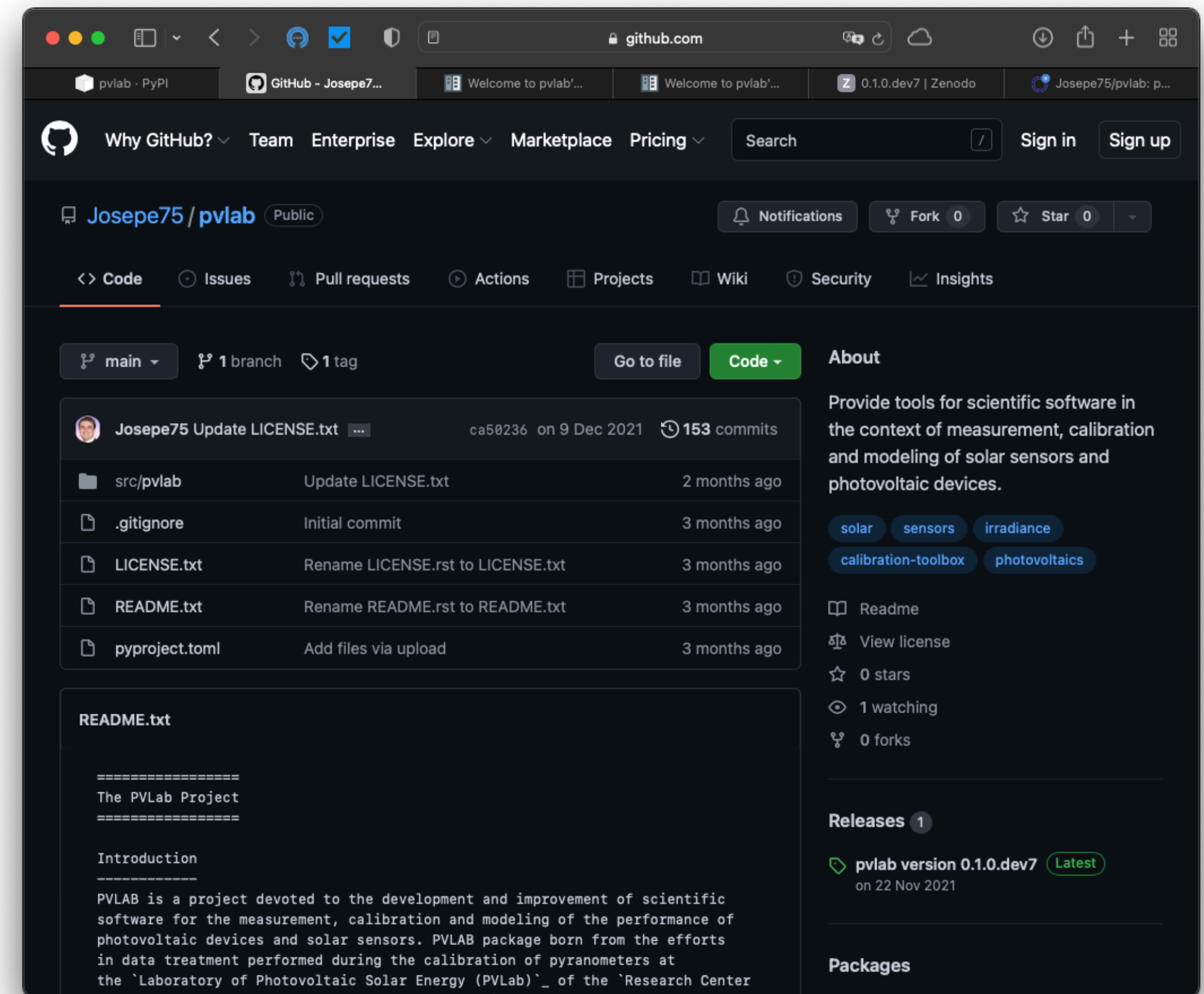


Desarrollo de Software Científico - Técnico

Escenario

Los Servidores de Control de Versión (VCS), como **Git**, han sistematizado el desarrollo de proyectos colaborativos (multitarea) de un modo sencillo, coherente y 100% trazable, apoyados por:

- Editores de código (**Atom**) capaces de generar las propuestas de incorporación de cambios (*pull requests*) y gestionarlas.
- Plataformas (**ReadtheDocs**) capaces de extraer el sub-proyecto documental (si existe) para generar automáticamente la documentación (mediante herramientas como **sphinx**) y proveer su alojamiento en internet.



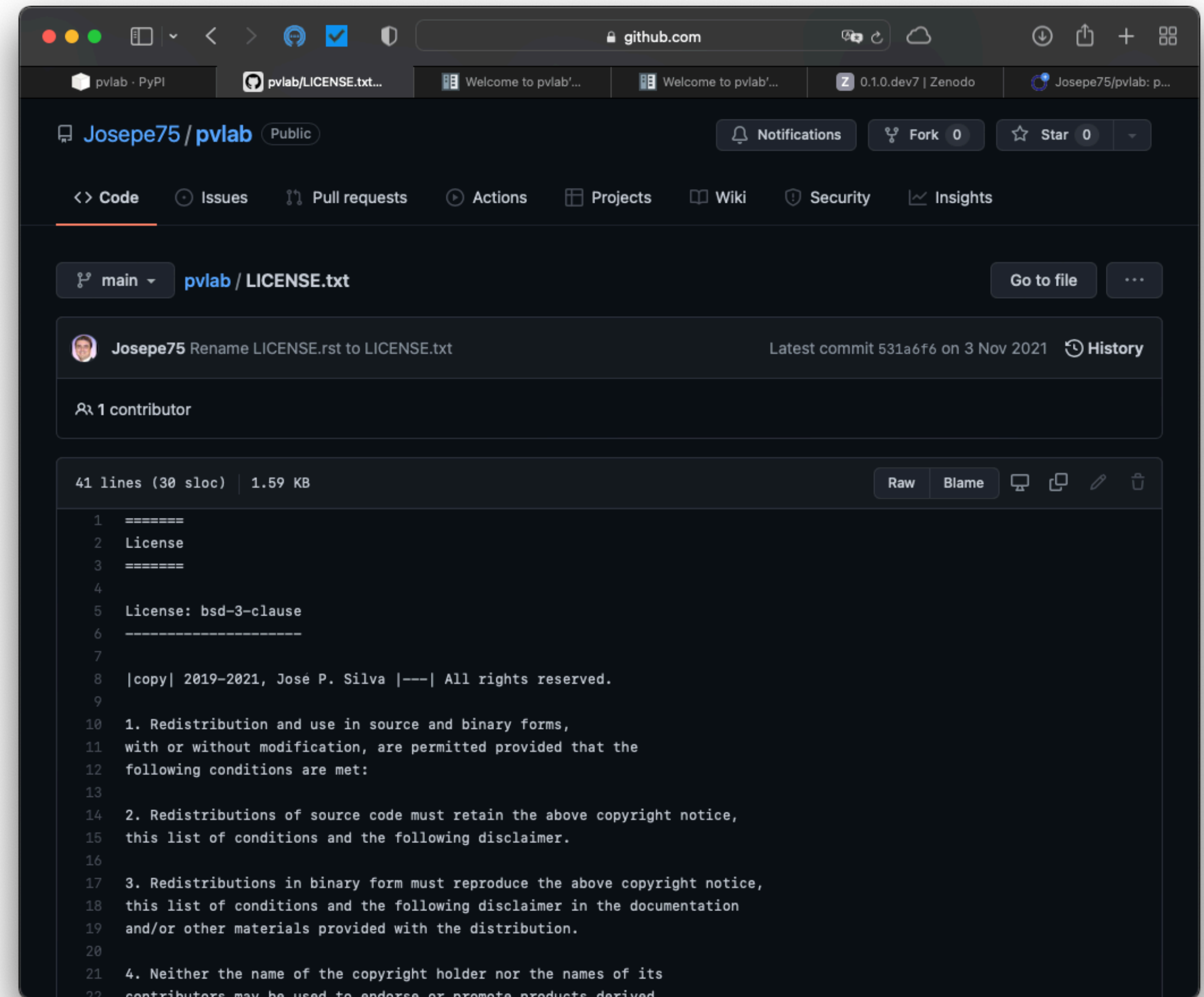
Pantalla de inicio del proyecto **pvlab** en **GitHub**.

Desarrollo de Software Científico - Técnico

Escenario

Los Servidores de Control de Versión (VCS), como **Git**, han sistematizado el desarrollo de proyectos colaborativos (multitarea) de un modo sencillo, coherente y 100% trazable, apoyados por:

- Editores de código (**Atom**) capaces de generar las propuestas de incorporación de cambios (*pull requests*) y gestionarlas.
- Plataformas (**ReadtheDocs**) capaces de extraer el sub-proyecto documental (si existe) para generar automáticamente la documentación (mediante herramientas como **sphinx**) y proveer su alojamiento en internet.



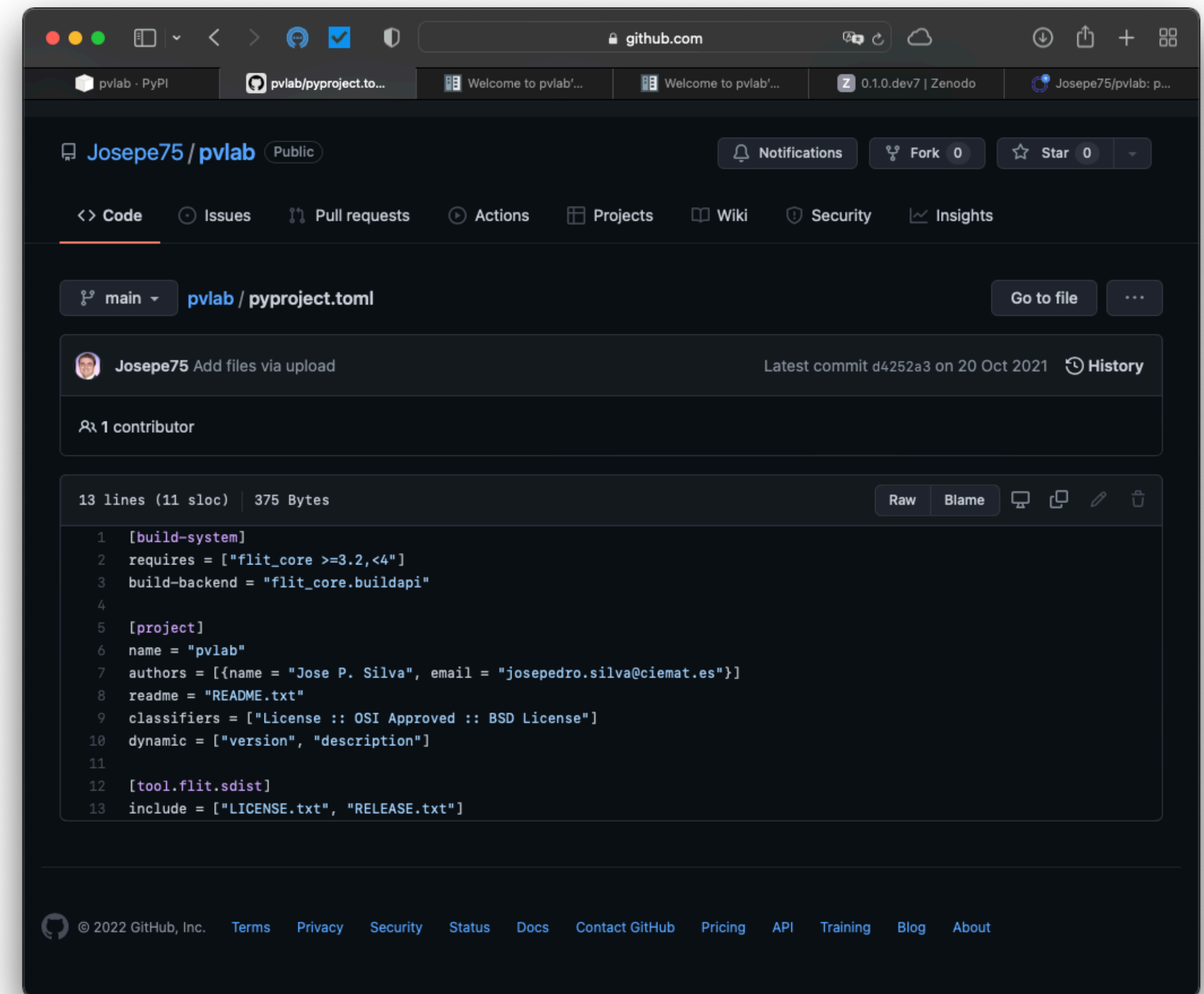
```
1  =====
2  License
3  =====
4
5  License: bsd-3-clause
6  -----
7
8  |copy| 2019-2021, José P. Silva |---| All rights reserved.
9
10 1. Redistribution and use in source and binary forms,
11 with or without modification, are permitted provided that the
12 following conditions are met:
13
14 2. Redistributions of source code must retain the above copyright notice,
15 this list of conditions and the following disclaimer.
16
17 3. Redistributions in binary form must reproduce the above copyright notice,
18 this list of conditions and the following disclaimer in the documentation
19 and/or other materials provided with the distribution.
20
21 4. Neither the name of the copyright holder nor the names of its
22 contributors may be used to endorse or promote products derived
```

Desarrollo de Software Científico - Técnico

Escenario

Los Servidores de Control de Versión (VCS), como **Git**, han sistematizado el desarrollo de proyectos colaborativos (multitarea) de un modo sencillo, coherente y 100% trazable, apoyados por:

- Editores de código (**Atom**) capaces de generar las propuestas de incorporación de cambios (*pull requests*) y gestionarlas.
- Plataformas (**ReadtheDocs**) capaces de extraer el sub-proyecto documental (si existe) para generar automáticamente la documentación (mediante herramientas como **sphinx**) y proveer su alojamiento en internet.



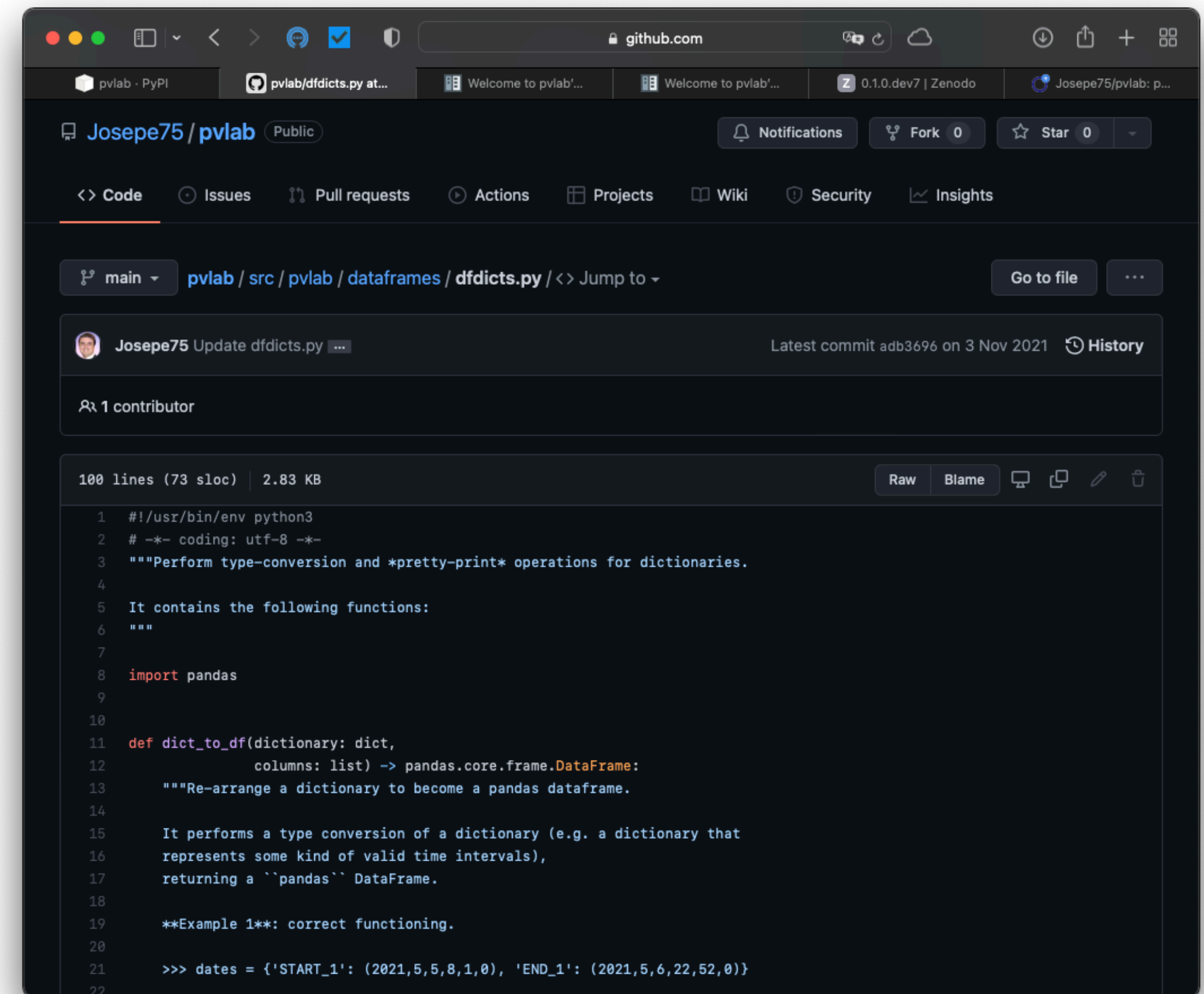
Vista del archivo de configuración **pyproject.toml**.

Desarrollo de Software Científico - Técnico

Escenario

Los Servidores de Control de Versión (VCS), como **Git**, han sistematizado el desarrollo de proyectos colaborativos (multitarea) de un modo sencillo, coherente y 100% trazable, apoyados por:

- Editores de código (**Atom**) capaces de generar las propuestas de incorporación de cambios (*pull requests*) y gestionarlas.
- Plataformas (**ReadtheDocs**) capaces de extraer el sub-proyecto documental (si existe) para generar automáticamente la documentación (mediante herramientas como **sphinx**) y proveer su alojamiento en internet.



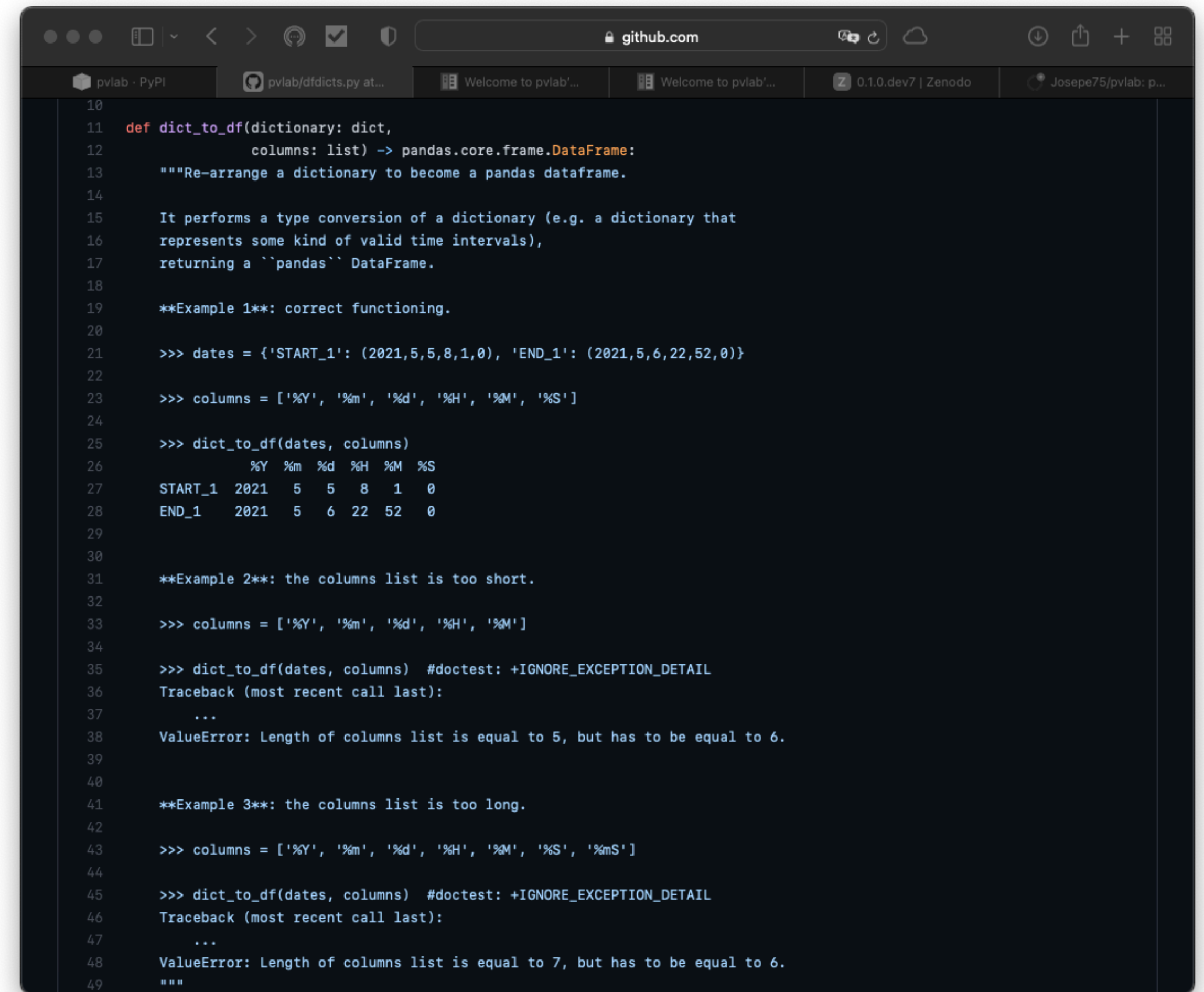
```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  """Perform type-conversion and *pretty-print* operations for dictionaries.
4
5  It contains the following functions:
6  """
7
8  import pandas
9
10
11 def dict_to_df(dictionary: dict,
12                columns: list) -> pandas.core.frame.DataFrame:
13     """Re-arrange a dictionary to become a pandas dataframe.
14
15     It performs a type conversion of a dictionary (e.g. a dictionary that
16     represents some kind of valid time intervals),
17     returning a ``pandas`` DataFrame.
18
19     **Example 1**:: correct functioning.
20
21     >>> dates = {'START_1': (2021,5,5,8,1,0), 'END_1': (2021,5,6,22,52,0)}
```


Desarrollo de Software Científico - Técnico

Escenario

Los Servidores de Control de Versión (VCS), como **Git**, han sistematizado el desarrollo de proyectos colaborativos (multitarea) de un modo sencillo, coherente y 100% trazable, apoyados por:

- Editores de código (**Atom**) capaces de generar las propuestas de incorporación de cambios (*pull requests*) y gestionarlas.
- Plataformas (**ReadtheDocs**) capaces de extraer el sub-proyecto documental (si existe) para generar automáticamente la documentación (mediante herramientas como **sphinx**) y proveer su alojamiento en internet.



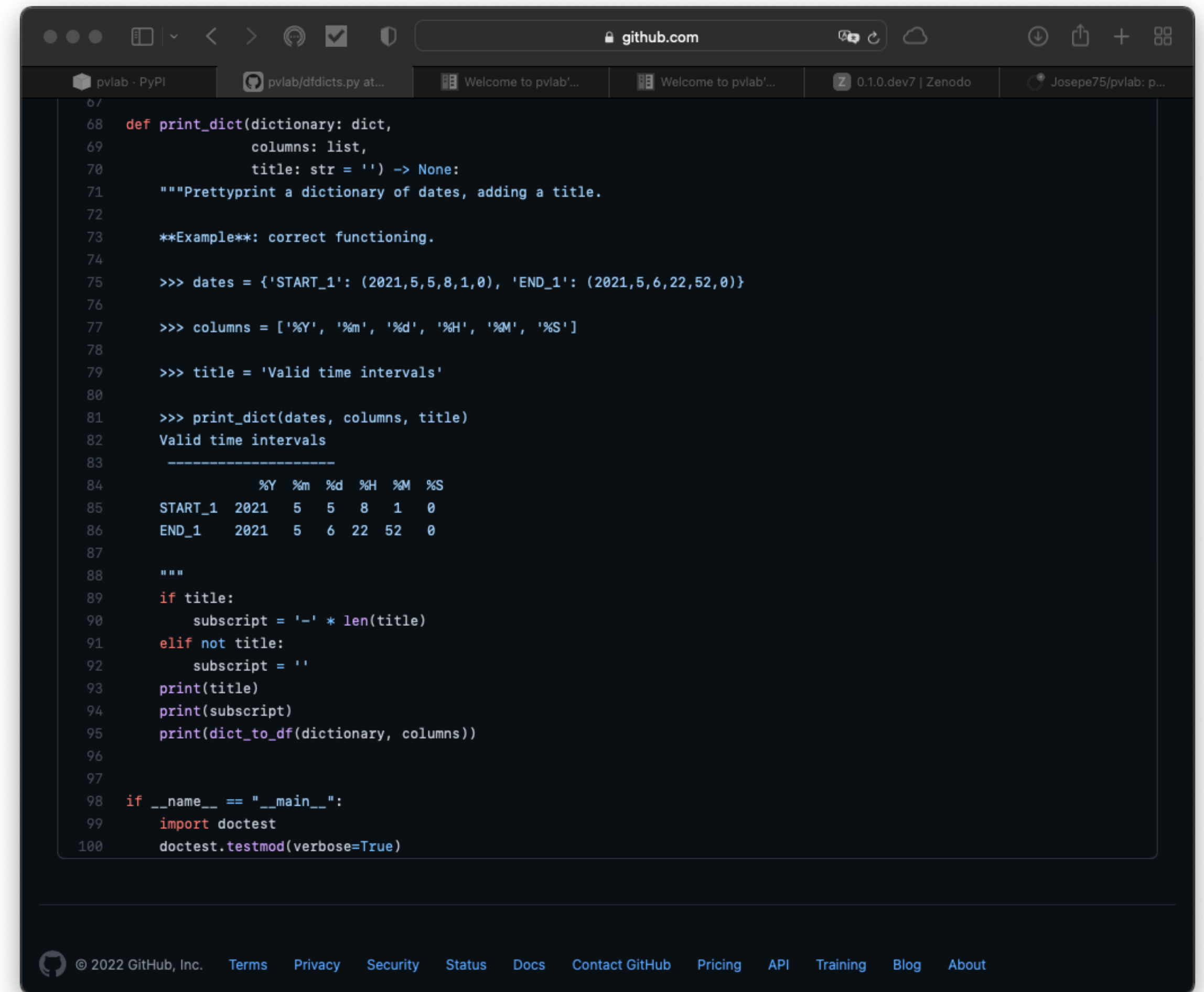
```
10
11 def dict_to_df(dictionary: dict,
12                 columns: list) -> pandas.core.frame.DataFrame:
13     """Re-arrange a dictionary to become a pandas dataframe.
14
15     It performs a type conversion of a dictionary (e.g. a dictionary that
16     represents some kind of valid time intervals),
17     returning a ``pandas`` DataFrame.
18
19     **Example 1**:: correct functioning.
20
21     >>> dates = {'START_1': (2021,5,5,8,1,0), 'END_1': (2021,5,6,22,52,0)}
22
23     >>> columns = ['%Y', '%m', '%d', '%H', '%M', '%S']
24
25     >>> dict_to_df(dates, columns)
26           %Y  %m  %d  %H  %M  %S
27     START_1 2021   5   5   8   1   0
28     END_1   2021   5   6  22  52   0
29
30
31     **Example 2**:: the columns list is too short.
32
33     >>> columns = ['%Y', '%m', '%d', '%H', '%M']
34
35     >>> dict_to_df(dates, columns) #doctest: +IGNORE_EXCEPTION_DETAIL
36     Traceback (most recent call last):
37         ...
38     ValueError: Length of columns list is equal to 5, but has to be equal to 6.
39
40
41     **Example 3**:: the columns list is too long.
42
43     >>> columns = ['%Y', '%m', '%d', '%H', '%M', '%S', '%mS']
44
45     >>> dict_to_df(dates, columns) #doctest: +IGNORE_EXCEPTION_DETAIL
46     Traceback (most recent call last):
47         ...
48     ValueError: Length of columns list is equal to 7, but has to be equal to 6.
49     """
```

Desarrollo de Software Científico - Técnico

Escenario

Los Servidores de Control de Versión (VCS), como **Git**, han sistematizado el desarrollo de proyectos colaborativos (multitarea) de un modo sencillo, coherente y 100% trazable, apoyados por:

- Editores de código (**Atom**) capaces de generar las propuestas de incorporación de cambios (*pull requests*) y gestionarlas.
- Plataformas (**ReadtheDocs**) capaces de extraer el sub-proyecto documental (si existe) para generar automáticamente la documentación (mediante herramientas como **sphinx**) y proveer su alojamiento en internet.



```
67
68 def print_dict(dictionary: dict,
69                 columns: list,
70                 title: str = '') -> None:
71     """Prettyprint a dictionary of dates, adding a title.
72
73     **Example**: correct functioning.
74
75     >>> dates = {'START_1': (2021,5,5,8,1,0), 'END_1': (2021,5,6,22,52,0)}
76
77     >>> columns = ['%Y', '%m', '%d', '%H', '%M', '%S']
78
79     >>> title = 'Valid time intervals'
80
81     >>> print_dict(dates, columns, title)
82     Valid time intervals
83     -----
84             %Y %m %d %H %M %S
85     START_1 2021  5  5  8  1  0
86     END_1   2021  5  6 22 52  0
87
88     """
89     if title:
90         subscript = '-' * len(title)
91     elif not title:
92         subscript = ''
93     print(title)
94     print(subscript)
95     print(dict_to_df(dictionary, columns))
96
97
98 if __name__ == "__main__":
99     import doctest
100     doctest.testmod(verbose=True)
```

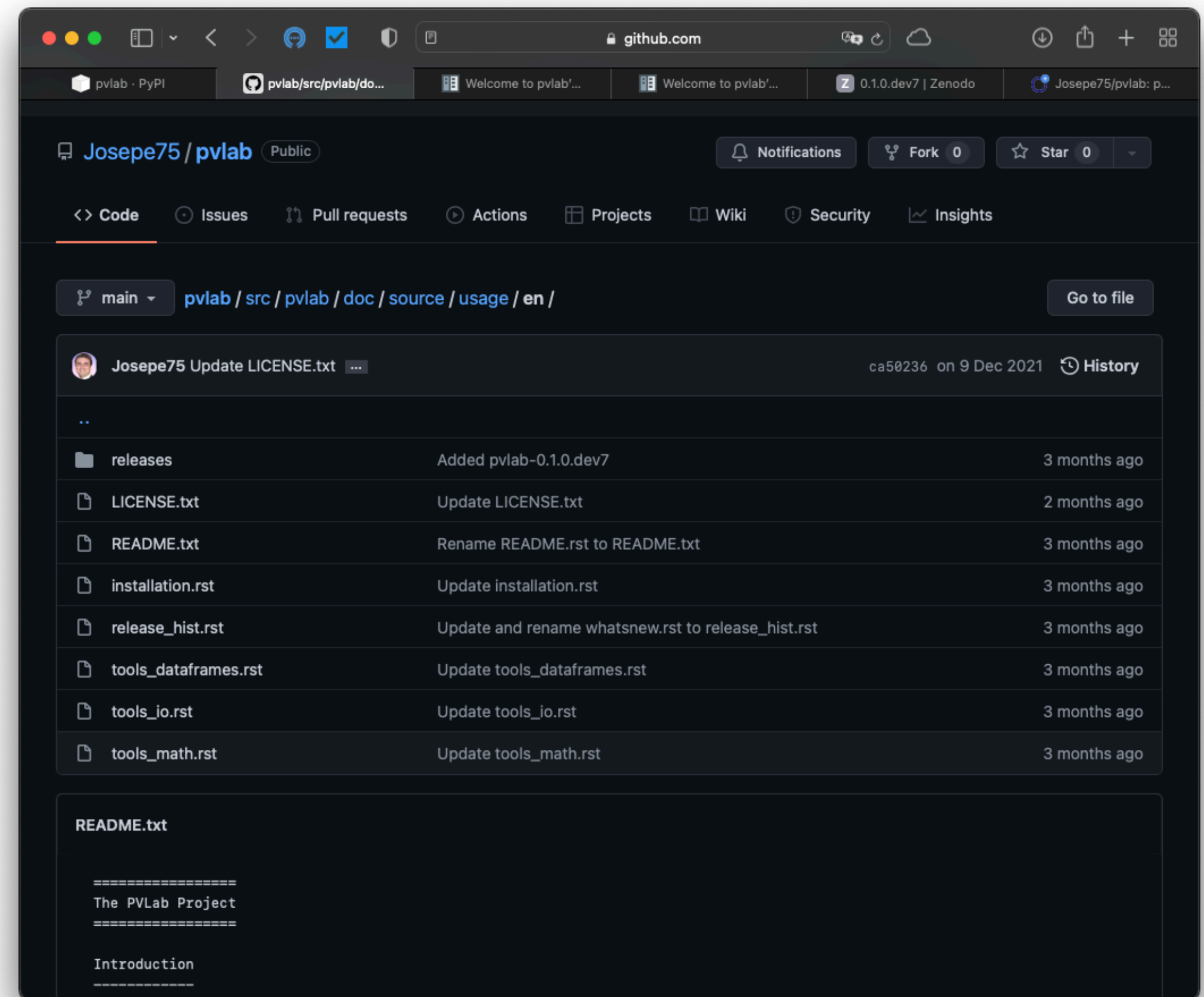
Código de la función **print_dict** y cláusula de ejecución.

Desarrollo de Software Científico - Técnico

Escenario

Los Servidores de Control de Versión (VCS), como **Git**, han sistematizado el desarrollo de proyectos colaborativos (multitarea) de un modo sencillo, coherente y 100% trazable, apoyados por:

- Editores de código (**Atom**) capaces de generar las propuestas de incorporación de cambios (*pull requests*) y gestionarlas.
- Plataformas (**ReadtheDocs**) capaces de extraer el sub-proyecto documental (si existe) para generar automáticamente la documentación (mediante herramientas como **sphinx**) y proveer su alojamiento en internet.

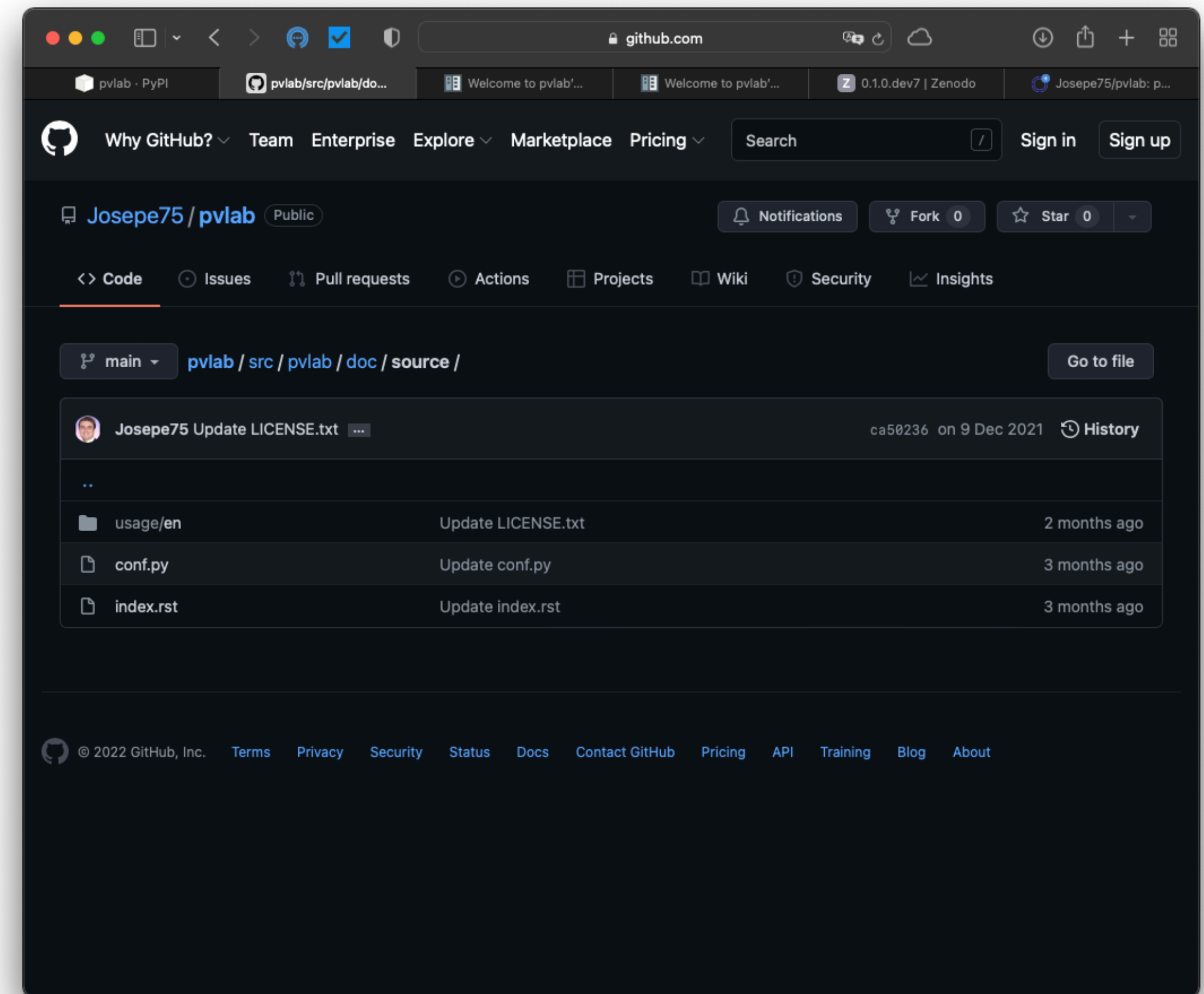


Desarrollo de Software Científico - Técnico

Escenario

Los Servidores de Control de Versión (VCS), como **Git**, han sistematizado el desarrollo de proyectos colaborativos (multitarea) de un modo sencillo, coherente y 100% trazable, apoyados por:

- Editores de código (**Atom**) capaces de generar las propuestas de incorporación de cambios (*pull requests*) y gestionarlas.
- Plataformas (**ReadtheDocs**) capaces de extraer el sub-proyecto documental (si existe) para generar automáticamente la documentación (mediante herramientas como **sphinx**) y proveer su alojamiento en internet.

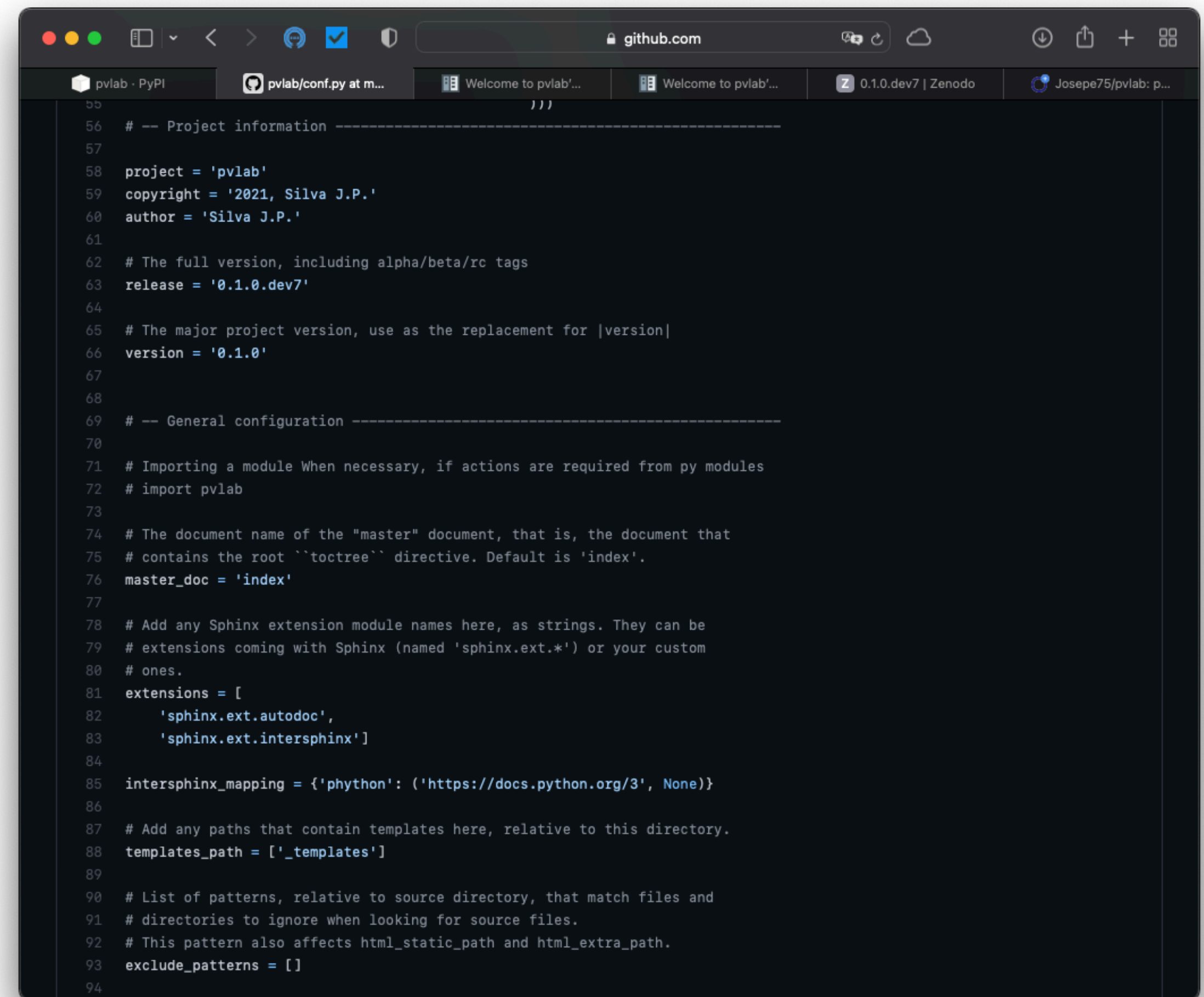


Desarrollo de Software Científico - Técnico

Escenario

Los Servidores de Control de Versión (VCS), como **Git**, han sistematizado el desarrollo de proyectos colaborativos (multitarea) de un modo sencillo, coherente y 100% trazable, apoyados por:

- Editores de código (**Atom**) capaces de generar las propuestas de incorporación de cambios (*pull requests*) y gestionarlas.
- Plataformas (**ReadtheDocs**) capaces de extraer el sub-proyecto documental (si existe) para generar automáticamente la documentación (mediante herramientas como **sphinx**) y proveer su alojamiento en internet.



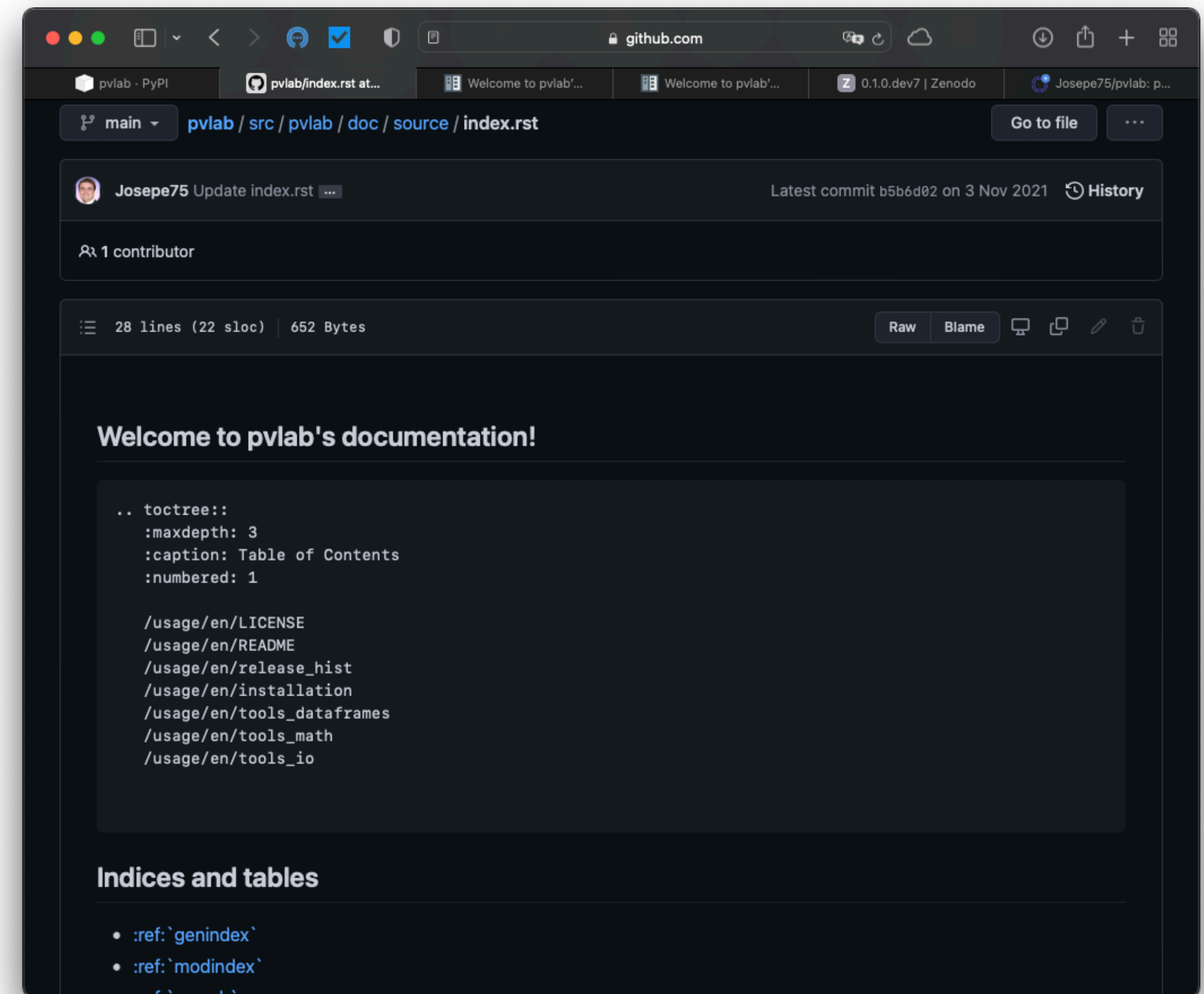
```
55
56 # -- Project information -----
57
58 project = 'pvlab'
59 copyright = '2021, Silva J.P.'
60 author = 'Silva J.P.'
61
62 # The full version, including alpha/beta/rc tags
63 release = '0.1.0.dev7'
64
65 # The major project version, use as the replacement for |version|
66 version = '0.1.0'
67
68
69 # -- General configuration -----
70
71 # Importing a module When necessary, if actions are required from py modules
72 # import pvlab
73
74 # The document name of the "master" document, that is, the document that
75 # contains the root ``toctree`` directive. Default is 'index'.
76 master_doc = 'index'
77
78 # Add any Sphinx extension module names here, as strings. They can be
79 # extensions coming with Sphinx (named 'sphinx.ext.*') or your custom
80 # ones.
81 extensions = [
82     'sphinx.ext.autodoc',
83     'sphinx.ext.intersphinx']
84
85 intersphinx_mapping = {'python': ('https://docs.python.org/3', None)}
86
87 # Add any paths that contain templates here, relative to this directory.
88 templates_path = ['_templates']
89
90 # List of patterns, relative to source directory, that match files and
91 # directories to ignore when looking for source files.
92 # This pattern also affects html_static_path and html_extra_path.
93 exclude_patterns = []
94
```


Desarrollo de Software Científico - Técnico

Escenario

Los Servidores de Control de Versión (VCS), como **Git**, han sistematizado el desarrollo de proyectos colaborativos (multitarea) de un modo sencillo, coherente y 100% trazable, apoyados por:

- Editores de código (**Atom**) capaces de generar las propuestas de incorporación de cambios (*pull requests*) y gestionarlas.
- Plataformas (**ReadtheDocs**) capaces de extraer el sub-proyecto documental (si existe) para generar automáticamente la documentación (mediante herramientas como **sphinx**) y proveer su alojamiento en internet.

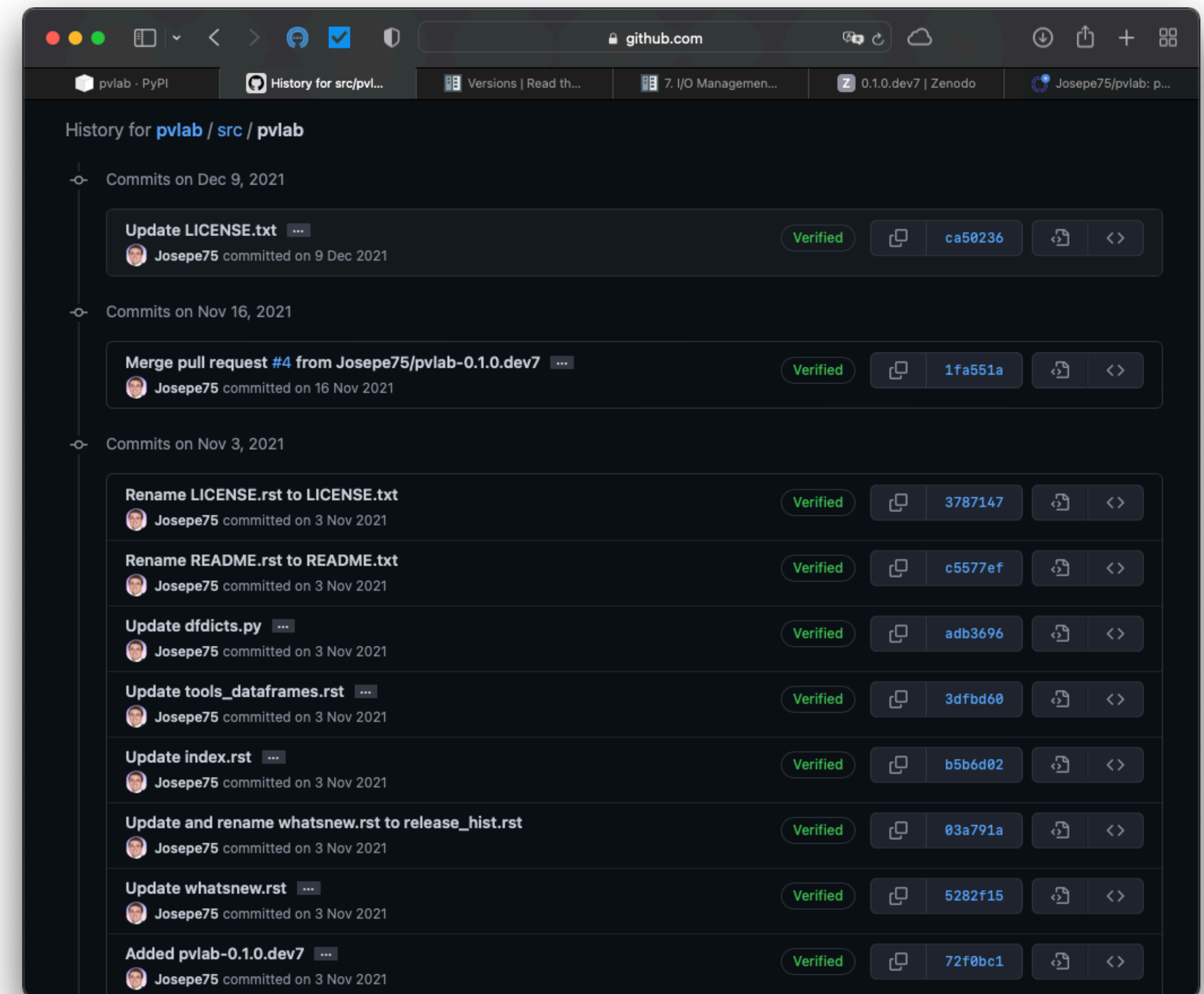


Desarrollo de Software Científico - Técnico

Escenario

Los Servidores de Control de Versión (VCS), como **Git**, han sistematizado el desarrollo de proyectos colaborativos (multitarea) de un modo sencillo, coherente y 100% trazable, apoyados por:

- Editores de código (**Atom**) capaces de generar las propuestas de incorporación de cambios (*pull requests*) y gestionarlas.
- Plataformas (**ReadtheDocs**) capaces de extraer el sub-proyecto documental (si existe) para generar automáticamente la documentación (mediante herramientas como **sphinx**) y proveer su alojamiento en internet.

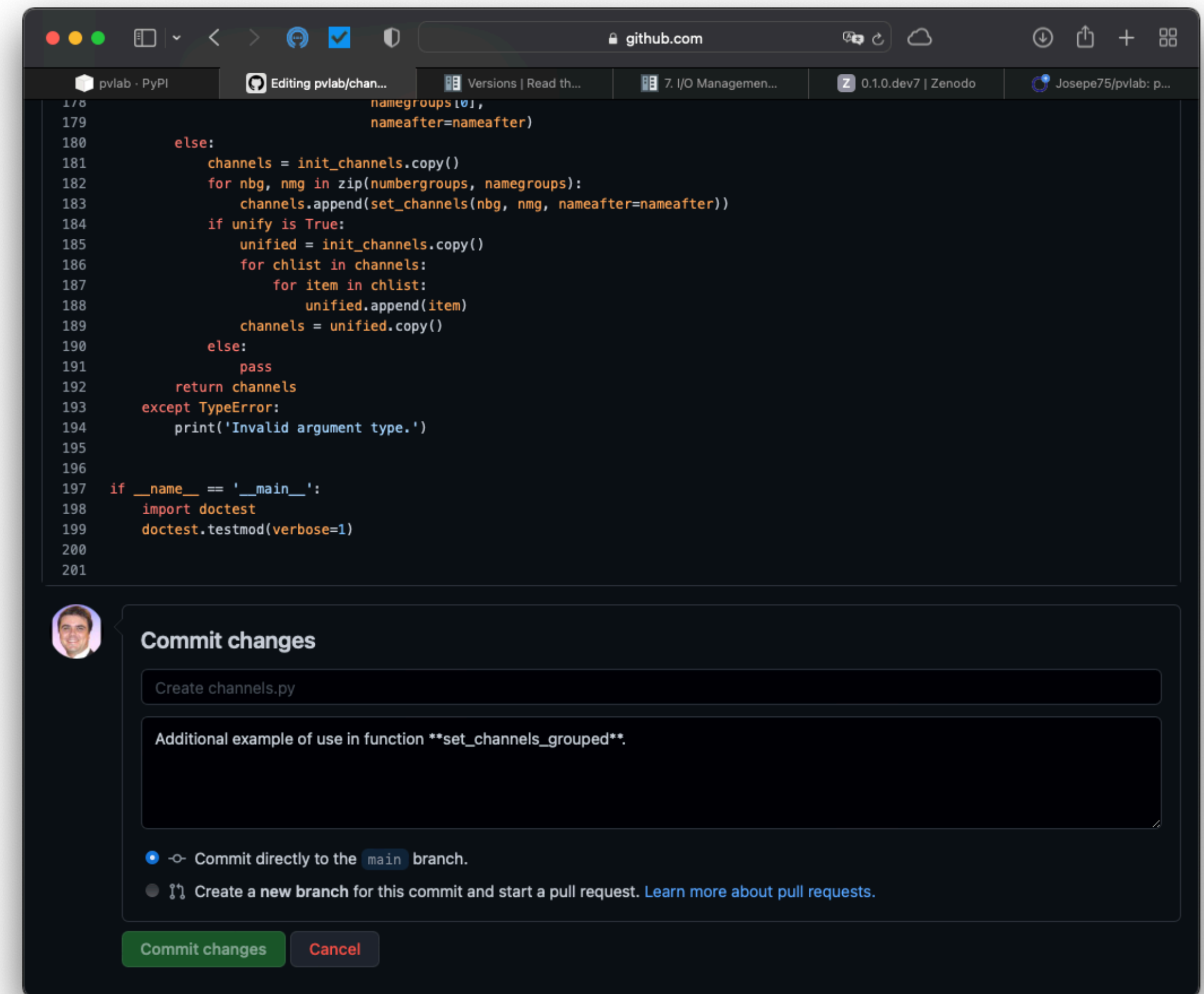


Desarrollo de Software Científico - Técnico

Escenario

Los Servidores de Control de Versión (VCS), como **Git**, han sistematizado el desarrollo de proyectos colaborativos (multitarea) de un modo sencillo, coherente y 100% trazable, apoyados por:

- Editores de código (**Atom**) capaces de generar las propuestas de incorporación de cambios (*pull requests*) y gestionarlas.
- Plataformas (**ReadtheDocs**) capaces de extraer el sub-proyecto documental (si existe) para generar automáticamente la documentación (mediante herramientas como **sphinx**) y proveer su alojamiento en internet.

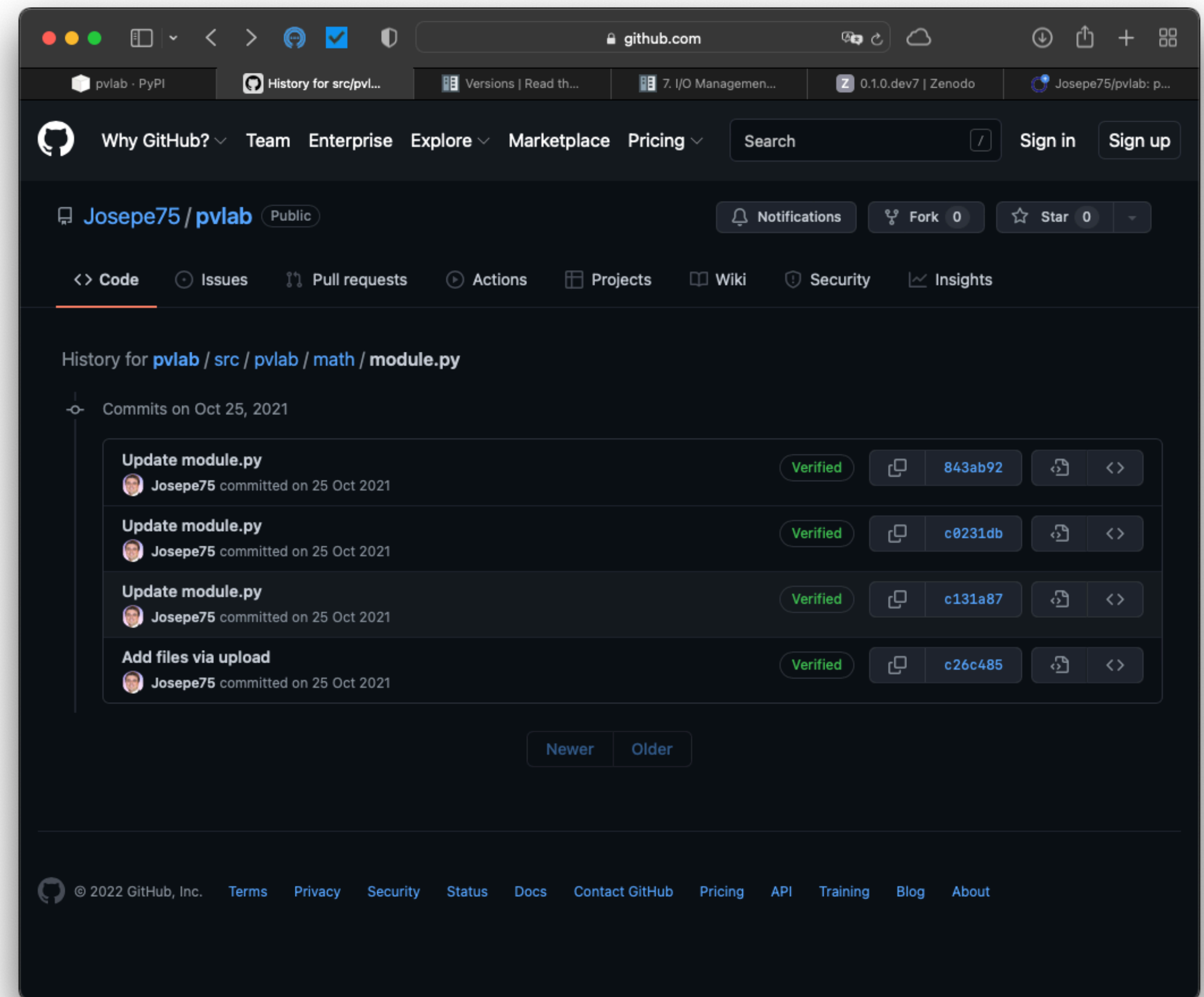


Desarrollo de Software Científico - Técnico

Escenario

Los Servidores de Control de Versión (VCS), como **Git**, han sistematizado el desarrollo de proyectos colaborativos (multitarea) de un modo sencillo, coherente y 100% trazable, apoyados por:

- Editores de código (**Atom**) capaces de generar las propuestas de incorporación de cambios (*pull requests*) y gestionarlas.
- Plataformas (**ReadtheDocs**) capaces de extraer el sub-proyecto documental (si existe) para generar automáticamente la documentación (mediante herramientas como **sphinx**) y proveer su alojamiento en internet.

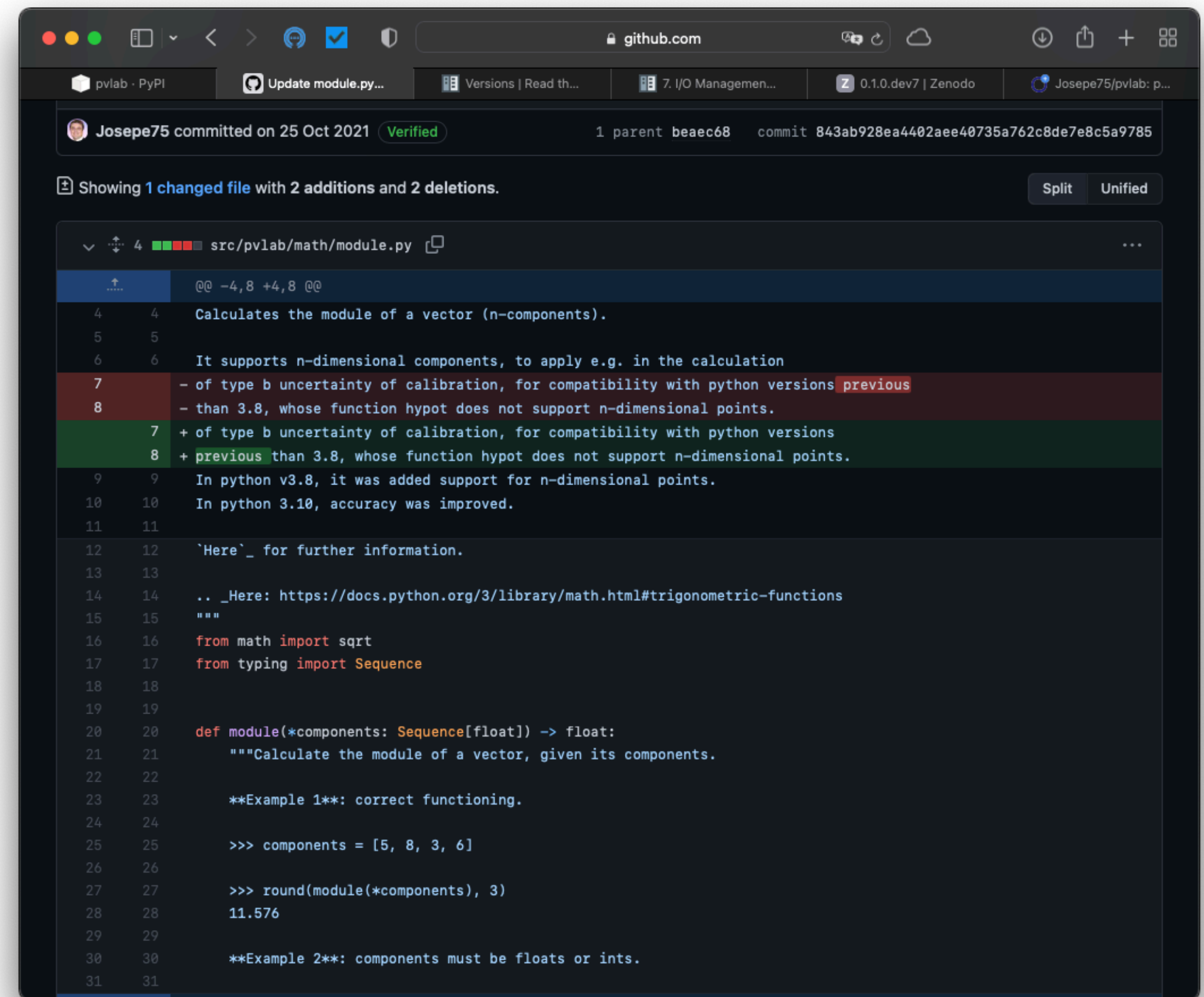


Desarrollo de Software Científico - Técnico

Escenario

Los Servidores de Control de Versión (VCS), como **Git**, han sistematizado el desarrollo de proyectos colaborativos (multitarea) de un modo sencillo, coherente y 100% trazable, apoyados por:

- Editores de código (**Atom**) capaces de generar las propuestas de incorporación de cambios (*pull requests*) y gestionarlas.
- Plataformas (**ReadtheDocs**) capaces de extraer el sub-proyecto documental (si existe) para generar automáticamente la documentación (mediante herramientas como **sphinx**) y proveer su alojamiento en internet.



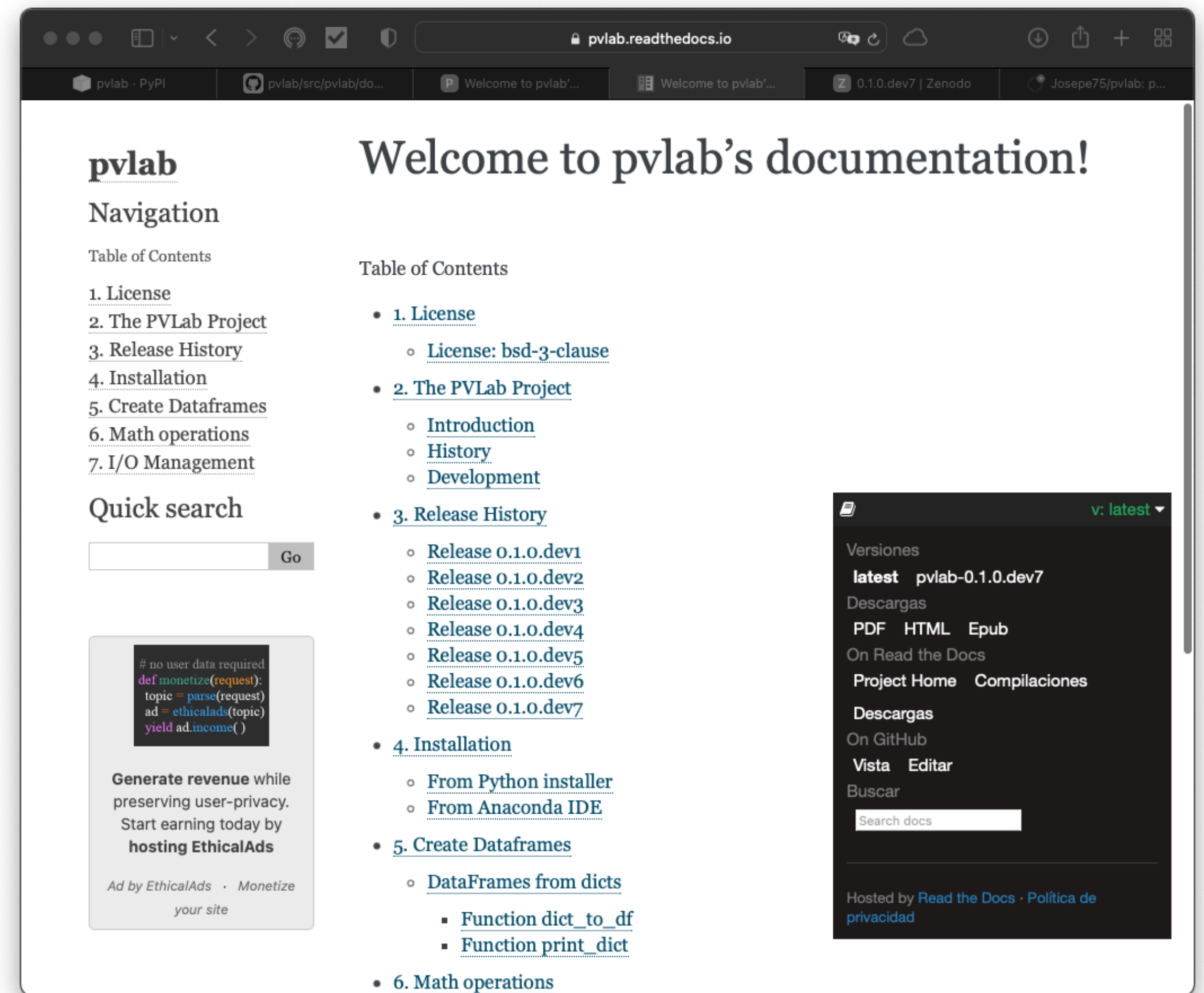
```
@@ -4,8 +4,8 @@
4 4 Calculates the module of a vector (n-components).
5 5
6 6 It supports n-dimensional components, to apply e.g. in the calculation
7 - of type b uncertainty of calibration, for compatibility with python versions previous
8 - than 3.8, whose function hypot does not support n-dimensional points.
7 + of type b uncertainty of calibration, for compatibility with python versions
8 + previous than 3.8, whose function hypot does not support n-dimensional points.
9 9 In python v3.8, it was added support for n-dimensional points.
10 10 In python 3.10, accuracy was improved.
11 11
12 12 `Here`_ for further information.
13 13
14 14 .. _Here: https://docs.python.org/3/library/math.html#trigonometric-functions
15 15 """
16 16 from math import sqrt
17 17 from typing import Sequence
18 18
19 19
20 20 def module(*components: Sequence[float]) -> float:
21 21     """Calculate the module of a vector, given its components.
22 22
23 23     **Example 1**: correct functioning.
24 24
25 25     >>> components = [5, 8, 3, 6]
26 26
27 27     >>> round(module(*components), 3)
28 28     11.576
29 29
30 30     **Example 2**: components must be floats or ints.
31 31
```

Desarrollo de Software Científico - Técnico

Escenario

Los Servidores de Control de Versión (VCS), como **Git**, han sistematizado el desarrollo de proyectos colaborativos (multitarea) de un modo sencillo, coherente y 100% trazable, apoyados por:

- Editores de código (**Atom**) capaces de generar las propuestas de incorporación de cambios (*pull requests*) y gestionarlas.
- Plataformas (**ReadtheDocs**) capaces de extraer el sub-proyecto documental (si existe) para generar automáticamente la documentación (mediante herramientas como **sphinx**) y proveer su alojamiento en internet.

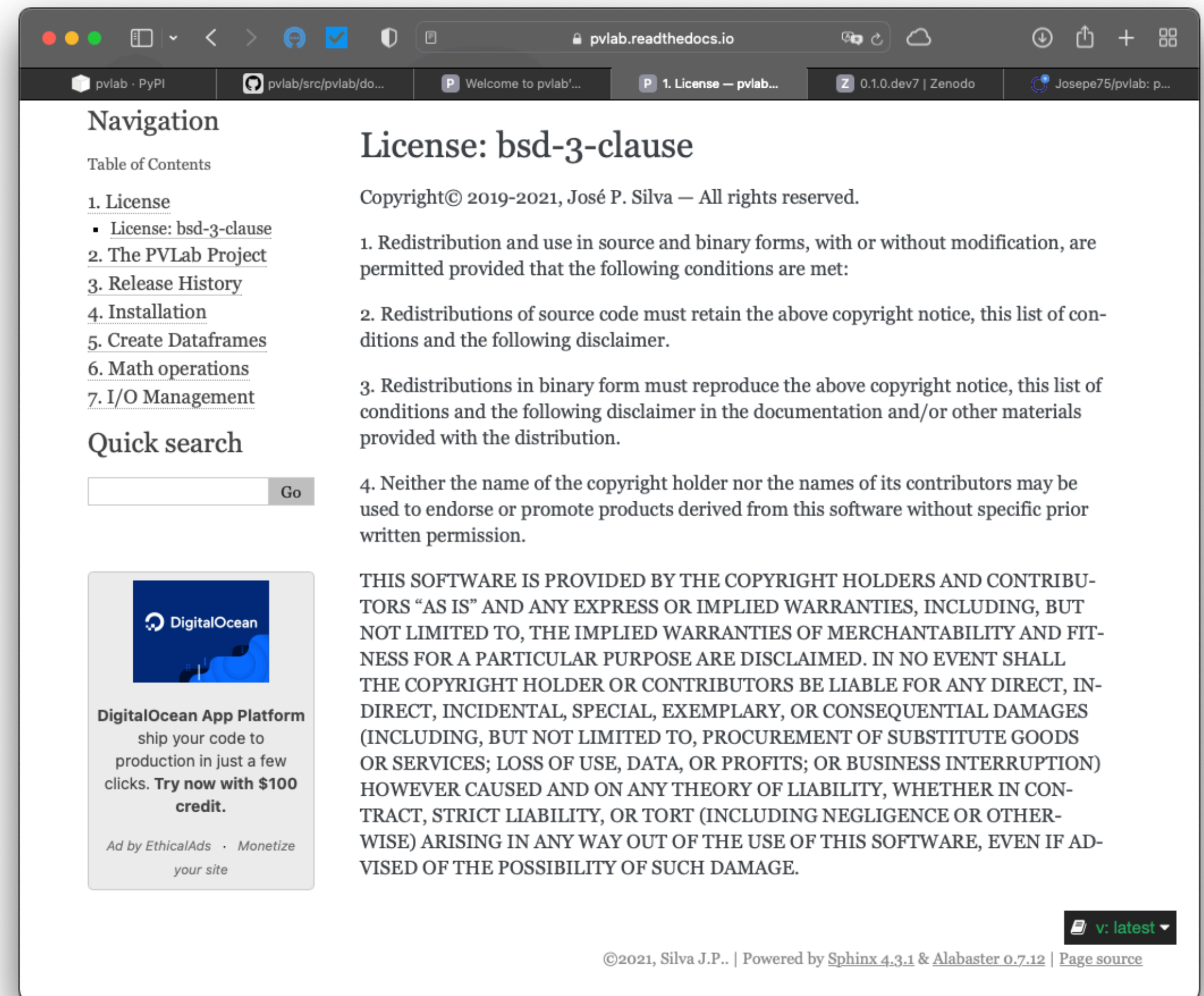


Desarrollo de Software Científico - Técnico

Escenario

Los Servidores de Control de Versión (VCS), como **Git**, han sistematizado el desarrollo de proyectos colaborativos (multitarea) de un modo sencillo, coherente y 100% trazable, apoyados por:

- Editores de código (**Atom**) capaces de generar las propuestas de incorporación de cambios (*pull requests*) y gestionarlas.
- Plataformas (**ReadtheDocs**) capaces de extraer el sub-proyecto documental (si existe) para generar automáticamente la documentación (mediante herramientas como **sphinx**) y proveer su alojamiento en internet.

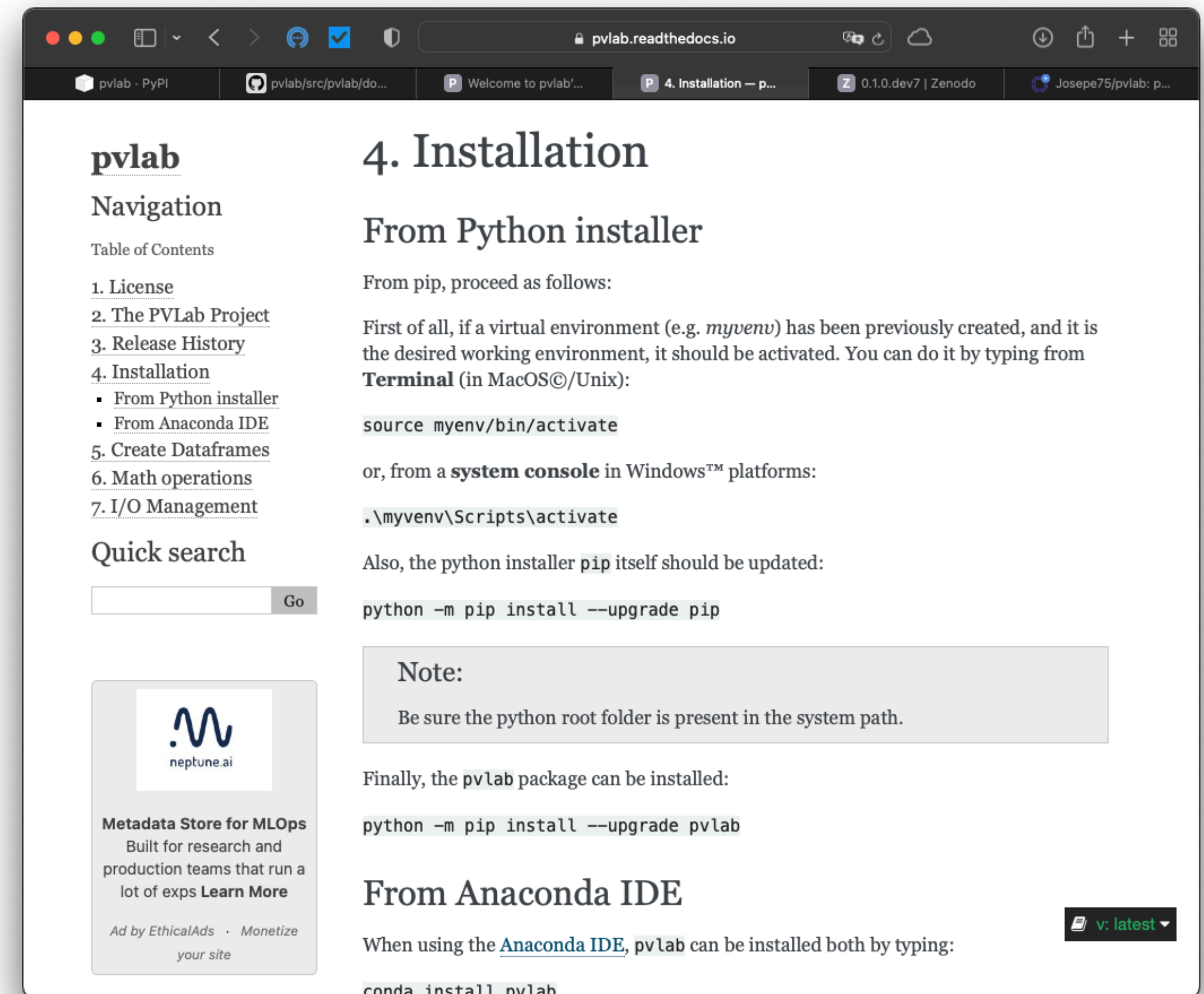


Desarrollo de Software Científico - Técnico

Escenario

Los Servidores de Control de Versión (VCS), como **Git**, han sistematizado el desarrollo de proyectos colaborativos (multitarea) de un modo sencillo, coherente y 100% trazable, apoyados por:

- Editores de código (**Atom**) capaces de generar las propuestas de incorporación de cambios (*pull requests*) y gestionarlas.
- Plataformas (**ReadtheDocs**) capaces de extraer el sub-proyecto documental (si existe) para generar automáticamente la documentación (mediante herramientas como **sphinx**) y proveer su alojamiento en internet.

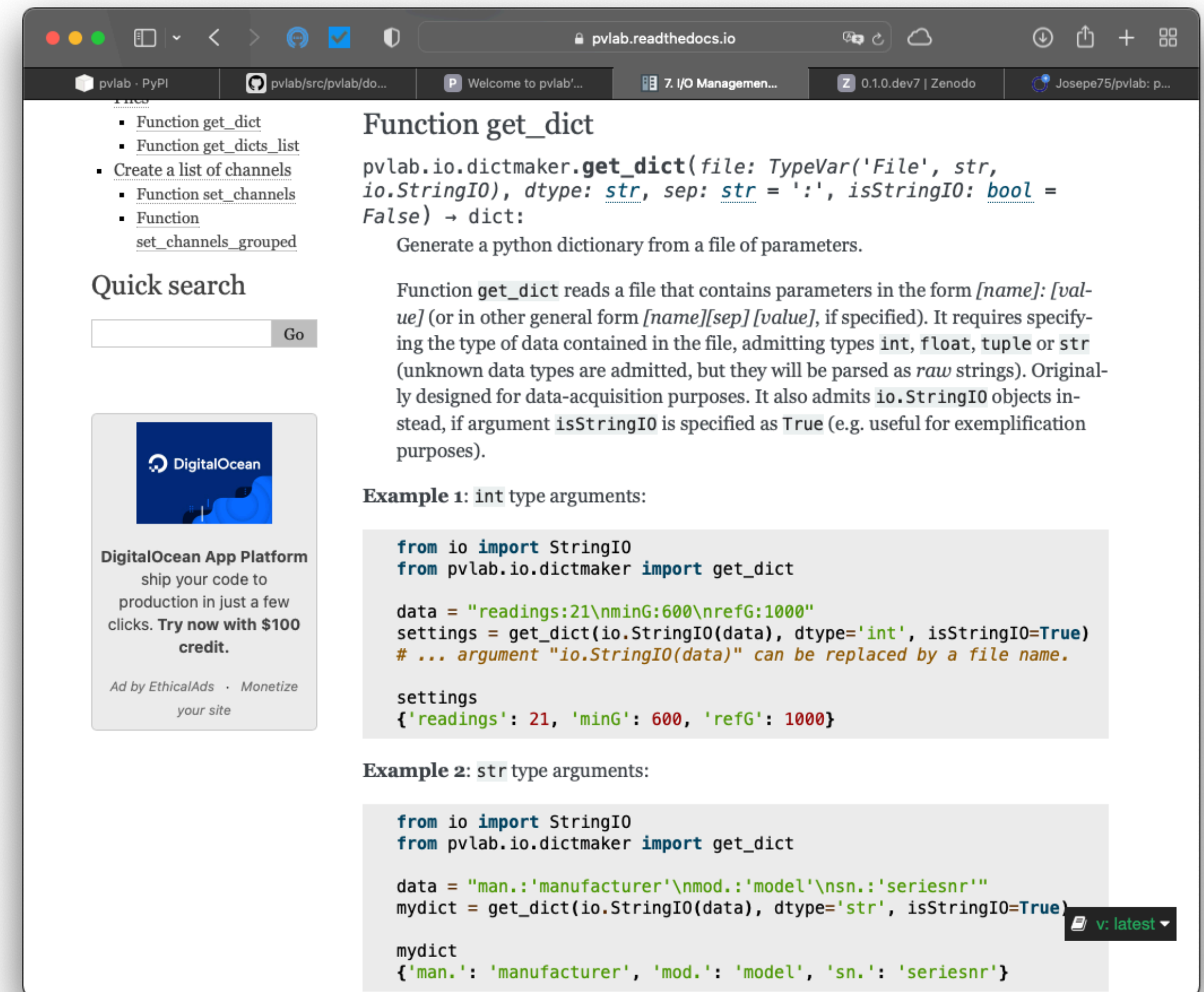


Desarrollo de Software Científico - Técnico

Escenario

Los Servidores de Control de Versión (VCS), como **Git**, han sistematizado el desarrollo de proyectos colaborativos (multitarea) de un modo sencillo, coherente y 100% trazable, apoyados por:

- Editores de código (**Atom**) capaces de generar las propuestas de incorporación de cambios (*pull requests*) y gestionarlas.
- Plataformas (**ReadtheDocs**) capaces de extraer el sub-proyecto documental (si existe) para generar automáticamente la documentación (mediante herramientas como **sphinx**) y proveer su alojamiento en internet.



The screenshot shows the Read the Docs website for the 'pvlab' project. The page displays the documentation for the 'get_dict' function. On the left, there is a sidebar with a table of contents and a search bar. The main content area shows the function signature and description. Below the description, there are two examples of how to use the function: 'Example 1: int type arguments' and 'Example 2: str type arguments'. The examples show code snippets that use the 'get_dict' function to parse data from a string or file into a dictionary.

Function `get_dict`

`pvlab.io.dictmaker.get_dict(file: TypeVar('File', str, io.StringIO), dtype: str, sep: str = ':', isStringIO: bool = False) → dict:`

Generate a python dictionary from a file of parameters.

Function `get_dict` reads a file that contains parameters in the form `[name]: [value]` (or in other general form `[name][sep] [value]`, if specified). It requires specifying the type of data contained in the file, admitting types `int`, `float`, `tuple` or `str` (unknown data types are admitted, but they will be parsed as *raw* strings). Originally designed for data-acquisition purposes. It also admits `io.StringIO` objects instead, if argument `isStringIO` is specified as `True` (e.g. useful for exemplification purposes).

Example 1: `int` type arguments:

```
from io import StringIO
from pvlab.io.dictmaker import get_dict

data = "readings:21\nminG:600\nrefG:1000"
settings = get_dict(io.StringIO(data), dtype='int', isStringIO=True)
# ... argument "io.StringIO(data)" can be replaced by a file name.

settings
{'readings': 21, 'minG': 600, 'refG': 1000}
```

Example 2: `str` type arguments:

```
from io import StringIO
from pvlab.io.dictmaker import get_dict

data = "man.: 'manufacturer'\nmod.: 'model'\nsn.: 'seriesnr'"
mydict = get_dict(io.StringIO(data), dtype='str', isStringIO=True)

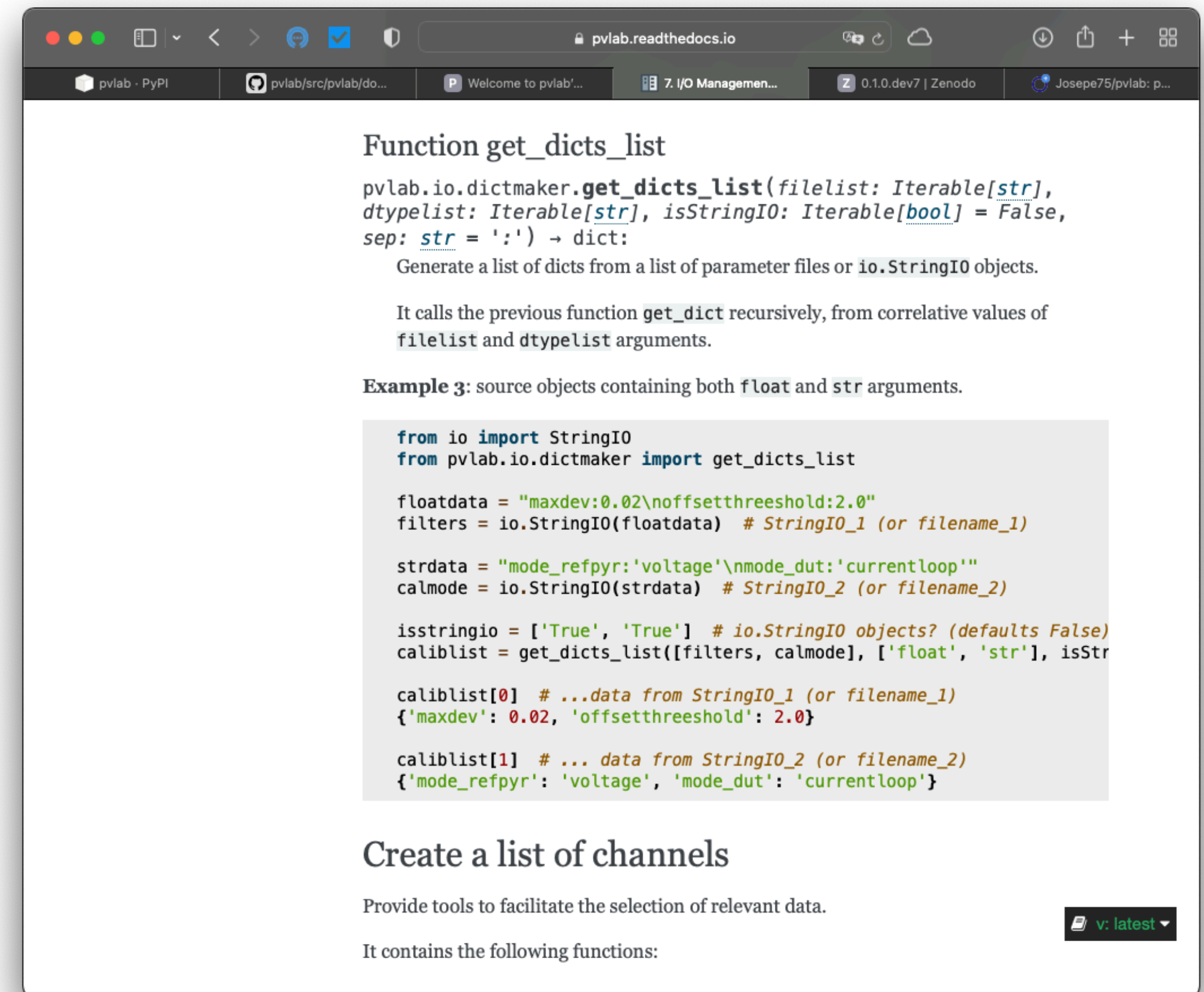
mydict
{'man.': 'manufacturer', 'mod.': 'model', 'sn.': 'seriesnr'}
```


Desarrollo de Software Científico - Técnico

Escenario

Los Servidores de Control de Versión (VCS), como **Git**, han sistematizado el desarrollo de proyectos colaborativos (multitarea) de un modo sencillo, coherente y 100% trazable, apoyados por:

- Editores de código (**Atom**) capaces de generar las propuestas de incorporación de cambios (*pull requests*) y gestionarlas.
- Plataformas (**ReadtheDocs**) capaces de extraer el sub-proyecto documental (si existe) para generar automáticamente la documentación (mediante herramientas como **sphinx**) y proveer su alojamiento en internet.

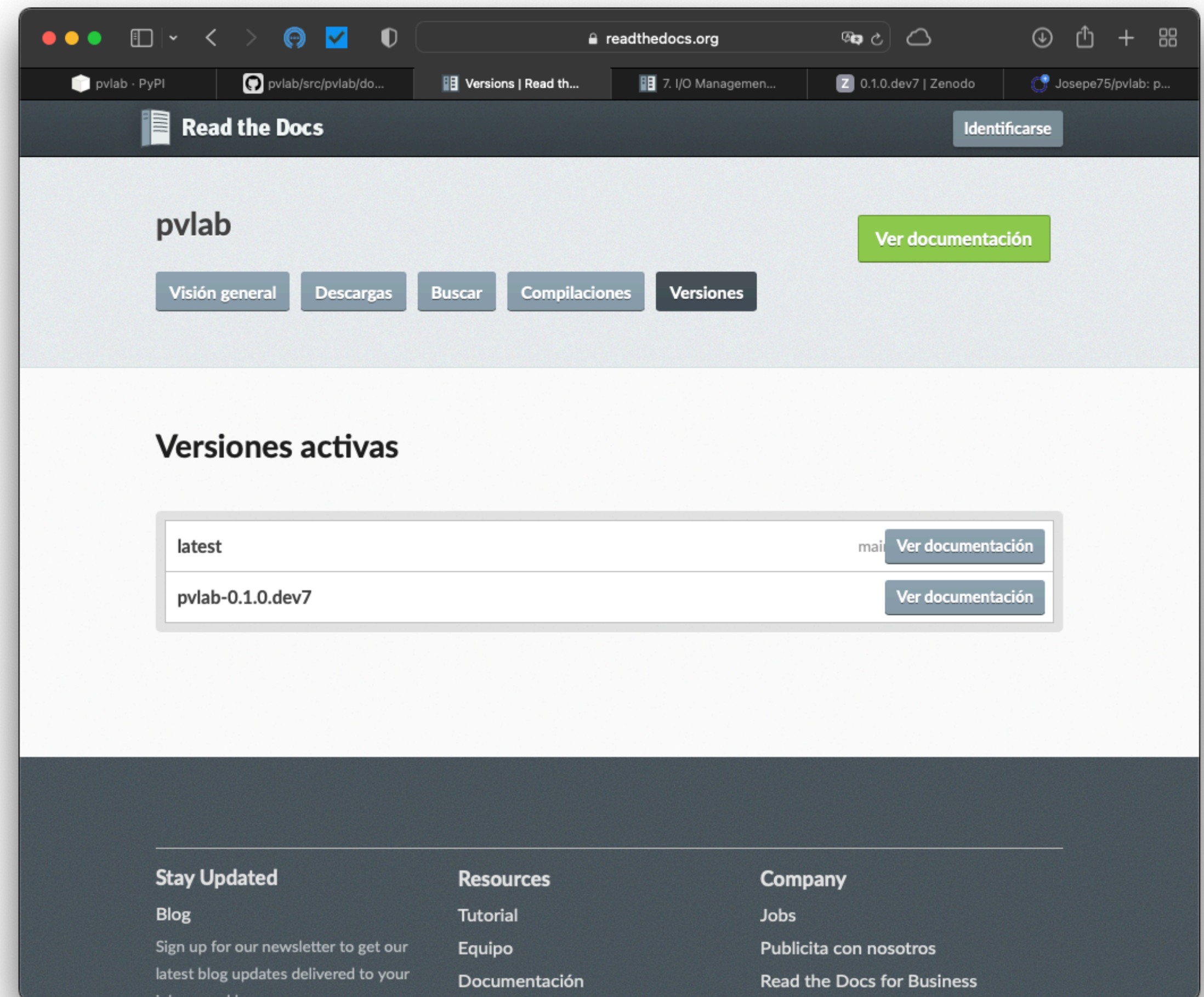


Desarrollo de Software Científico - Técnico

Escenario

Los Servidores de Control de Versión (VCS), como **Git**, han sistematizado el desarrollo de proyectos colaborativos (multitarea) de un modo sencillo, coherente y 100% trazable, apoyados por:

- Editores de código (**Atom**) capaces de generar las propuestas de incorporación de cambios (*pull requests*) y gestionarlas.
- Plataformas (**ReadtheDocs**) capaces de extraer el subproyecto documental (si existe) para generar automáticamente la documentación (mediante herramientas como **sphinx**) y proveer su alojamiento en internet.

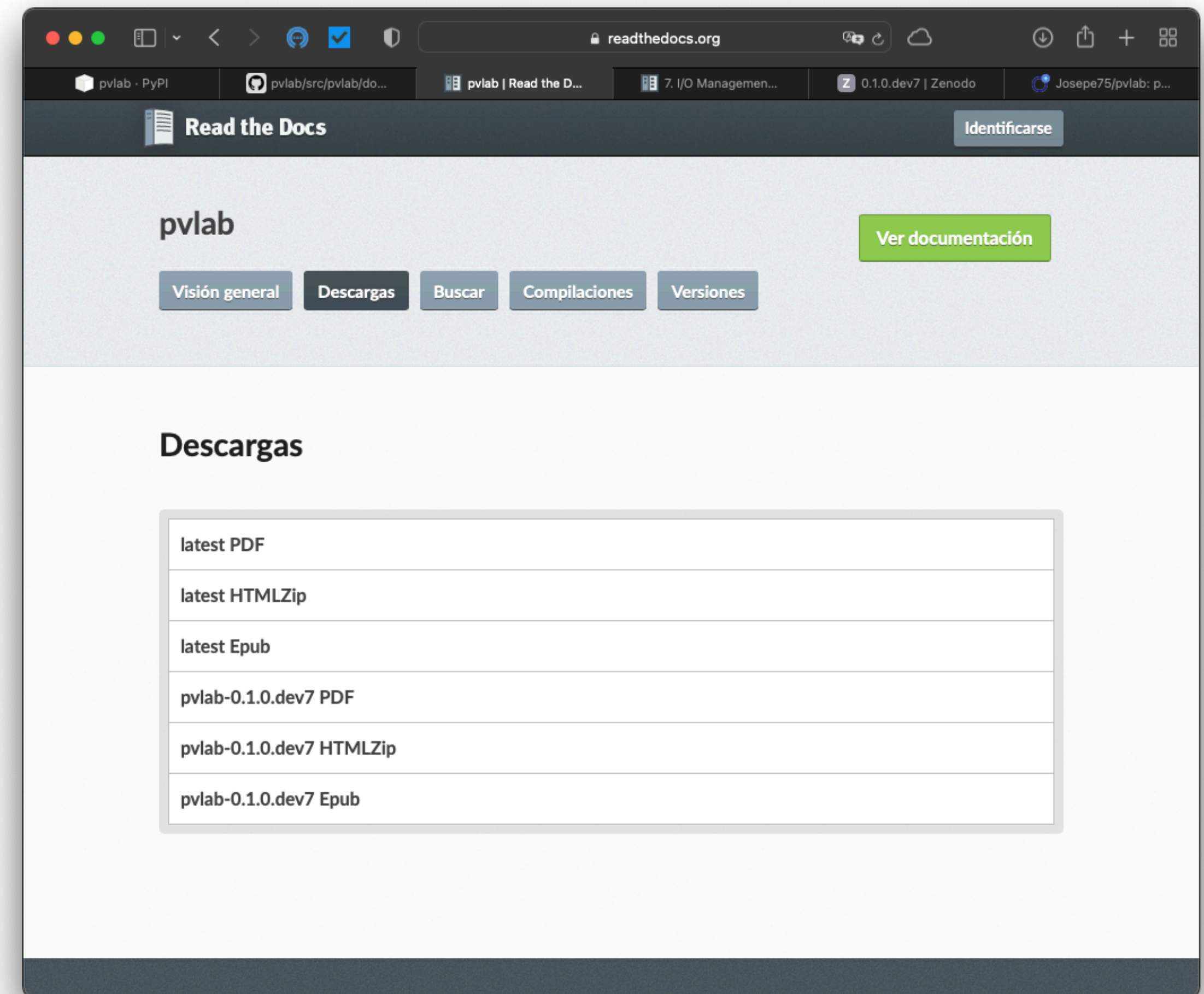


Desarrollo de Software Científico - Técnico

Escenario

Los Servidores de Control de Versión (VCS), como **Git**, han sistematizado el desarrollo de proyectos colaborativos (multitarea) de un modo sencillo, coherente y 100% trazable, apoyados por:

- Editores de código (**Atom**) capaces de generar las propuestas de incorporación de cambios (*pull requests*) y gestionarlas.
- Plataformas (**ReadtheDocs**) capaces de extraer el sub-proyecto documental (si existe) para generar automáticamente la documentación (mediante herramientas como **sphinx**) y proveer su alojamiento en internet.

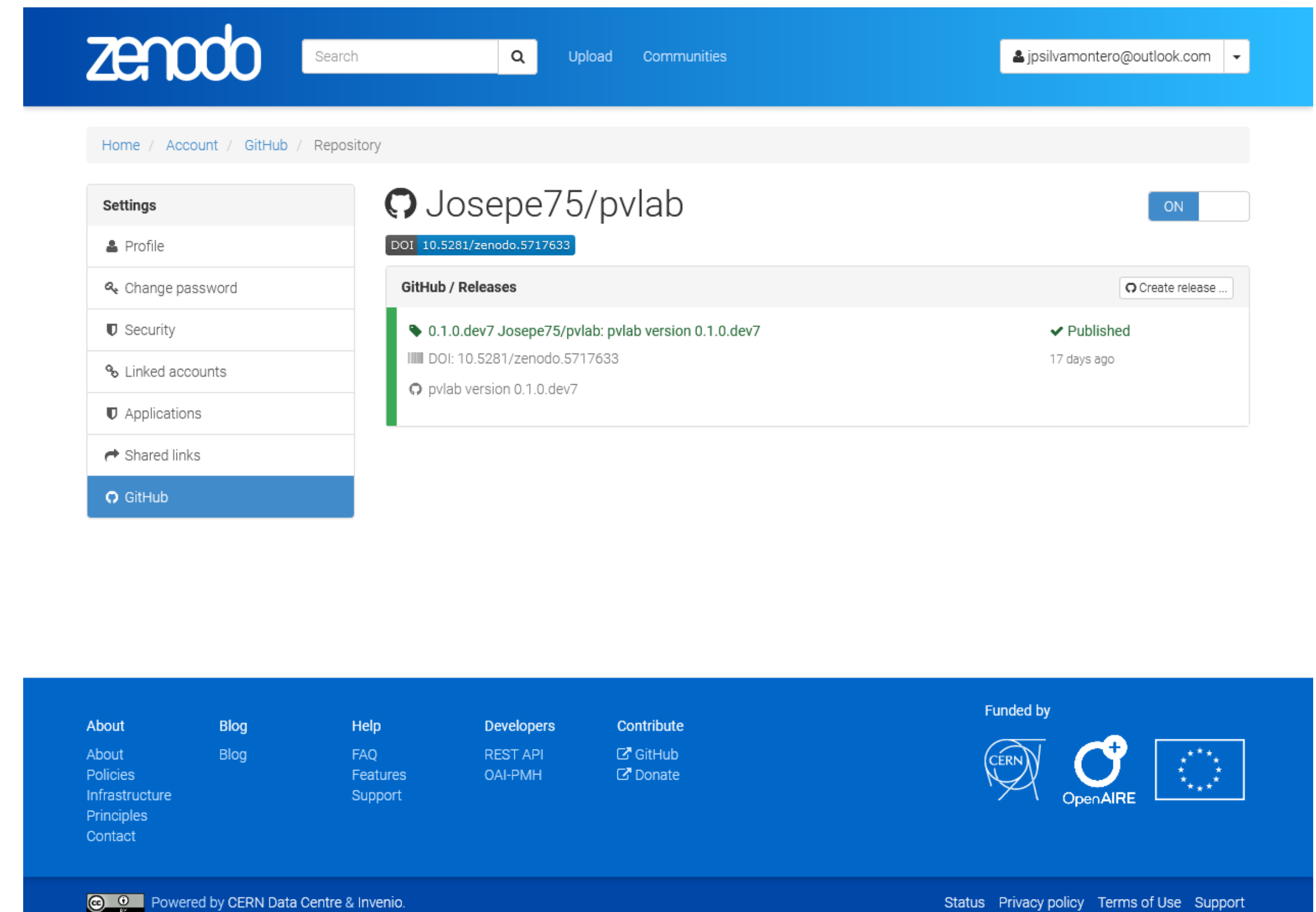


Desarrollo de Software Científico - Técnico

Escenario

Se ha consolidado la filosofía del **acceso abierto de la documentación** (Open Access), inspirado en la consigna *FAIR* (Findable, Accessible, Interoperable, Reusable), promovidas en textos legislativos y prácticas institucionales, apoyada por:

- **Licencias de acceso abierto**, bien para la documentación en general (p.ej. *Creative Commons*, CC) o más específicas del software (*GNU*, licencias *OSI - approved: BSD 3 - clause, Apache License*, etc.).
- **Repositorios documentales**, protegidos con licencias CC y con reconocimiento de la autoría mediante Links (*Documenta - Ciemat*), DOI (*Zenodo*), etc.



Desarrollo de Software Científico - Técnico

Escenario

Se han creado y desarrollado multitud de **publicaciones indexadas y revisadas por pares**, orientadas a:

- **Divulgar los desarrollos** concretos de software, su gestión (implementación web, distribución, ...) y aplicaciones.
- Valorar la **contribución científica** de avances en los propios procesos de computación (nuevos algoritmos, estructuras, procesos básicos, etc.).

IEEE TRANSACTIONS ON ARTIFICIAL INTELLIGENCE

A PUBLICATION OF
THE IEEE COMPUTATIONAL INTELLIGENCE SOCIETY
THE IEEE COMPUTER SOCIETY
THE IEEE SIGNAL PROCESSING SOCIETY
THE IEEE SYSTEMS, MAN, AND CYBERNETICS SOCIETY
THE IEEE ROBOTICS & AUTOMATION SOCIETY



FEBRUARY 2022

VOLUME 3

NUMBER 1

ITAICB

(ISSN 2691-4581)



IEEE TRANSACTIONS ON ARTIFICIAL INTELLIGENCE

Desarrollo de Software Científico - Técnico

Escenario

Se han creado y desarrollado multitud de **publicaciones indexadas y revisadas por pares**, orientadas a:

- **Divulgar los desarrollos** concretos de software, su gestión (implementación web, distribución, ...) y aplicaciones.
- Valorar la **contribución científica** de avances en los propios procesos de computación (nuevos algoritmos, estructuras, procesos básicos, etc.).

Home Popular Early Access Current Issue All Issues About Jou

Aims & Scope

The *IEEE Transactions on Artificial Intelligence* (TAI) is a multidisciplinary journal publishing papers on theories and methodologies of Artificial Intelligence. Applications of Artificial Intelligence are also considered.

The articles in this journal are peer reviewed in accordance with the requirements set forth in the *IEEE PSPB Operations Manual* (sections 8.2.1.C & 8.2.2.A). Each published article was reviewed by a minimum of two independent reviewers using a single-blind peer review process, where the identities of the reviewers are not known to the authors, but the reviewers know the identities of the authors. Articles will be screened for plagiarism before acceptance.

Corresponding authors from low-income countries are eligible for *waived or reduced APCs*.

Publication Details

Subjects

IEEE TRANSACTIONS ON ARTIFICIAL INTELLIGENCE

Desarrollo de Software Científico - Técnico

Escenario

Se han creado y desarrollado multitud de **publicaciones indexadas y revisadas por pares**, orientadas a:

- **Divulgar los desarrollos** concretos de software, su gestión (implementación web, distribución, ...) y aplicaciones.
- Valorar la **contribución científica** de avances en los propios procesos de computación (nuevos algoritmos, estructuras, procesos básicos, etc.).

Popular Articles	Latest Published Articles
AutoFac: The Perpetual Robot Machine Geoff Nitschke; David Howard	
Kolmogorov–Smirnov Test-Based Actively-Adaptive Thompson Sampling For Non-Stationary Bandits Gourab Ghatak; Hardhik Mohanty; Aniq Ur Rahman	
Playing Against Deep-Neural-Network-Based Object Detectors: A Novel Bidirectional Adversarial Attack Approach Xiang Li; Yuchen Jiang; Chenglin Liu; Shaochong Liu; Hao Luo; Shen Yin	
Bipartite Cooperative Coevolution For Energy-Aware Coverage Path Planning Of UAVs Xian-Xin Shao; Yue-Jiao Gong; Zhi-Hui Zhan; Jun Zhang	
Domain Adaptation And Autoencoder-Based Unsupervised Speech Enhancement Yi Li; Yang Sun; Kirill Horoshenkov; Syed Mohsen Naqvi	

Show All

Desarrollo de Software Científico - Técnico

Escenario

Se han creado y desarrollado multitud de **publicaciones indexadas y revisadas por pares**, orientadas a:

- **Divulgar los desarrollos** concretos de software, su gestión (implementación web, distribución, ...) y aplicaciones.
- Valorar la **contribución científica** de avances en los propios procesos de computación (nuevos algoritmos, estructuras, procesos básicos, etc.).



Data Science and Engineering

[Editorial board](#) [Aims & scope](#) [Journal updates](#)

Data Science and Engineering (DSE) is an international, peer-reviewed, open access journal published on behalf of the China Computer Federation (CCF), and is affiliated with CCF Technical Committee on Database (CCF TCDB). Focusing on the theoretical background and advanced engineering approaches, DSE aims to offer a prime forum for researchers, professionals, and industrial practitioners to share their knowledge in this rapidly growing area. It provides in-depth coverage of the latest advances in the closely related fields of data science and data engineering. — [show all](#)

Editor-in-Chief
Bin Cui, Timos Sellis

Publishing model
Open Access. [How to publish with us](#)

SPRINGER DATA SCIENCE AND ENGINEERING

Desarrollo de Software Científico - Técnico

Escenario

Se han creado y desarrollado multitud de **publicaciones indexadas y revisadas por pares**, orientadas a:

- **Divulgar los desarrollos** concretos de software, su gestión (implementación web, distribución, ...) y aplicaciones.
- Valorar la **contribución científica** de avances en los propios procesos de computación (nuevos algoritmos, estructuras, procesos básicos, etc.).

Aims and scope

The journal of Data Science and Engineering (DSE) responds to the remarkable change in the focus of information technology development from CPU-intensive computation to data-intensive computation, where the effective application of data, especially big data, becomes vital. The emerging discipline data science and engineering, an interdisciplinary field integrating theories and methods from computer science, statistics, information science, and other fields, focuses on the foundations and engineering of efficient and effective techniques and systems for data collection and management, for data integration and correlation, for information and knowledge extraction from massive data sets, and for data use in different application domains.

Focusing on the theoretical background and advanced engineering approaches, DSE aims to offer a prime forum for researchers, professionals, and industrial practitioners to share their knowledge in this rapidly growing area. It provides in-depth coverage of the latest advances in the closely related fields of data science and data engineering. More specifically, DSE covers four areas: (i) the data itself, i.e., the nature and quality of the data, especially big data; (ii) the principles of information extraction from data, especially big data; (iii) the theory behind data-intensive computing; and (iv) the techniques and systems used to analyze and manage big data. DSE welcomes papers that explore the above subjects. Specific topics include, but are not limited to:

(a) the nature and quality of data,

(b) the computational complexity of data-intensive computing,

(c) new methods for the design and analysis of the algorithms for solving problems with

SPRINGER DATA SCIENCE AND ENGINEERING

Desarrollo de Software Científico - Técnico

Escenario

Se han creado y desarrollado multitud de **publicaciones indexadas y revisadas por pares**, orientadas a:

- **Divulgar los desarrollos** concretos de software, su gestión (implementación web, distribución, ...) y aplicaciones.
- Valorar la **contribución científica** de avances en los propios procesos de computación (nuevos algoritmos, estructuras, procesos básicos, etc.).

(c) new methods for the design and analysis of the algorithms for solving problems with big data input,

(d) collection and integration of data collected from internet and sensing devices or sensor networks,

(e) representation, modeling, and visualization of big data,

(f) storage, transmission, and management of big data,

(g) methods and algorithms of data intensive computing, such as

- mining big data,
- online analysis processing of big data,
- big data-based machine learning,
- big data based decision-making,
- statistical computation of big data,
- graph-theoretic computation of big data,
- linear algebraic computation of big data, and
- big data-based optimization.

(h) hardware systems and software systems for data-intensive computing,

(i) data security, privacy, and trust, and

(j) novel applications of big data.

Desarrollo de Software Científico - Técnico

Escenario

Se han creado y desarrollado multitud de **publicaciones indexadas y revisadas por pares**, orientadas a:

- **Divulgar los desarrollos** concretos de software, su gestión (implementación web, distribución, ...) y aplicaciones.
- Valorar la **contribución científica** de avances en los propios procesos de computación (nuevos algoritmos, estructuras, procesos básicos, etc.).

Computing in Science & Engineering

[Submit Manuscript](#)[Home](#)[Popular](#)[Early Access](#)[Current Issue](#)**2.08**Impact
Factor**0.00204**

Eigenfactor

0.852Article
Influence
Score**3.6**CiteScore
Powered by Scopus

?

Author Resources

[Submission Guidelines](#)[Submit Manuscript](#)[Author Center](#)[Become a Reviewer](#)[Open Access Publishing Options](#)

Meet the Editor

Editor-in-Chief

Lorena A. Barba 

The George Washington University

labarba@gwu.edu

Popular Articles

What Is The Blockchain?

Massimo Di Pierro

Matplotlib: A 2D Graphics Environment

John D. Hunter

The End Of Moore's Law: A New Beginning For Information Technology

Thomas N. Theis; H.-S. Philip Wong

Data Visualization For The Understanding Of COVID-19

João L. D. Comba

Cloud-Native Repositories For Big Scientific Data

Ryan P. Abernathey; Tom Augspurger; Anderson Banihirwe; Charles C. Black Gentemann; Joseph J. Hamman; Naomi Henderson; Chiara Lepore; Theo A.

Desarrollo de Software Científico - Técnico

Escenario

Se han creado y desarrollado multitud de **publicaciones indexadas y revisadas por pares**, orientadas a:

- **Divulgar los desarrollos** concretos de software, su gestión (implementación web, distribución, ...) y aplicaciones.
- Valorar la **contribución científica** de avances en los propios procesos de computación (nuevos algoritmos, estructuras, procesos básicos, etc.).



Desarrollo de Software Científico - Técnico

Escenario

Se han creado y desarrollado multitud de **publicaciones indexadas y revisadas por pares**, orientadas a:

- **Divulgar los desarrollos** concretos de software, su gestión (implementación web, distribución, ...) y aplicaciones.
- Valorar la **contribución científica** de avances en los propios procesos de computación (nuevos algoritmos, estructuras, procesos básicos, etc.).

Browse Journals & Magazines > IEEE Transactions on Software ... ?

IEEE Transactions on Software Engineering

[Submit Manuscript](#) [Add Title To My Alerts](#) [Add to My Favorites](#)

Home	Popular	Early Access	Current Issue	All Issues	About Jou
------	---------	--------------	---------------	------------	-----------

6.226
Impact
Factor

0.00502
Eigenfactor

1.658
Article
Influence
Score

14.9
CiteScore
Powered by Scopus

?

Author Resources
[Submission Guidelines](#)
[Submit Manuscript](#)
[Author Center](#)
[Become a Reviewer](#)
[Additional Information](#)
[Open Access Publishing Options](#)

Popular Articles	Latest Published Articles
The Oracle Problem In Software Testing: A Survey Earl T. Barr; Mark Harman; Phil McMinn; Muzammil Shahbaz; Shin Yoo	
Evolution Of The Unix System Architecture: An Exploratory Case Study Dionidis Spinellis; Paris Avgeriou	
Automatic Software Repair: A Survey Luca Gazzola; Daniela Micucci; Leonardo Mariani	
Smart Contract Development: Challenges And Opportunities Weiqin Zou; David Lo; Pavneet Singh Kochhar; Xuan-Bach Dinh Le; Xin Xia; Yang Feng; Zhenyu Chen; Baowen Xu	
A Survey On Software Fault Localization W. Eric Wong; Ruizhi Gao; Yihao Li; Rui Abreu; Franz Wotawa	

Meet the Editor
Editor-in-Chief
Nenad Medvidovic 
University of Southern California
Viterbi School of Engineering,
Department of Computer Science
941 Bloom Walk, SAL 338
Los Angeles, CA 90089-0781
USA

S

IEEE TRANSACTIONS ON SOFTWARE ENGINEERING

Desarrollo de Software Científico - Técnico

Escenario

Se han creado y desarrollado multitud de **publicaciones indexadas y revisadas por pares**, orientadas a:

- **Divulgar los desarrollos** concretos de software, su gestión (implementación web, distribución, ...) y aplicaciones.
- Valorar la **contribución científica** de avances en los propios procesos de computación (nuevos algoritmos, estructuras, procesos básicos, etc.).

Browse Journals & Magazines > IEEE Transactions on Software ... ?

IEEE Transactions on Software Engineering

[Submit Manuscript](#) [Add Title To My Alerts](#) [Add to My Favorites](#)

Home	Popular	Early Access	Current Issue	All Issues	About Jou
------	---------	--------------	---------------	------------	-----------

6.226
Impact Factor

0.00502
Eigenfactor

1.658
Article Influence Score

14.9
CiteScore
Powered by Scopus

[Author Resources](#)

- [Submission Guidelines](#)
- [Submit Manuscript](#)
- [Author Center](#)
- [Become a Reviewer](#)
- [Additional Information](#)
- [Open Access Publishing Options](#)

Meet the Editor

Editor-in-Chief
Nenad Medvidovic 
University of Southern California
Viterbi School of Engineering,
Department of Computer Science
941 Bloom Walk, SAL 338
Los Angeles, CA 90089-0781
USA

Popular Articles	Latest Published Articles
	Machine Learning Testing: Survey, Landscapes And Horizons Jie M. Zhang; Mark Harman; Lei Ma; Yang Liu
	How Developers Choose Names Dror G. Feitelson; Ayelet Mizrahi; Nofar Noy; Aviad Ben Shabat; Or Eliyahu; Roy Sheffer
	Integrating Provenance Capture And UML With UML2PROV: Principles And Experie Carlos Sáenz-Adán; Beatriz Pérez; Francisco J. García-Izquierdo; Luc Moreau
	Code Reviews With Divergent Review Scores: An Empirical Study Of The OpenStac And Qt Communities Toshiki Hirao; Shane McIntosh; Akinori Ihara; Kenichi Matsumoto
	Just-In-Time Defect Identification And Localization: A Two-Phase Framework Meng Yan; Xin Xia; Yuanrui Fan; Ahmed E. Hassan; David Lo; Shanping Li

S

IEEE TRANSACTIONS ON SOFTWARE ENGINEERING

Desarrollo de Software Científico - Técnico

Escenario

Se han creado y desarrollado multitud de **publicaciones indexadas y revisadas por pares**, orientadas a:

- **Divulgar los desarrollos** concretos de software, su gestión (implementación web, distribución, ...) y aplicaciones.
- Valorar la **contribución científica** de avances en los propios procesos de computación (nuevos algoritmos, estructuras, procesos básicos, etc.).

The screenshot shows the ScienceDirect website for the journal 'Advances in Engineering Software'. The header includes the ScienceDirect logo and a search bar. Below the journal title, it states 'Supports open access'. A table displays the journal's metrics: a CiteScore of 10.1 and an Impact Factor of 4.141. The navigation bar includes a menu icon, a search bar, and links to 'Submit your article' and 'Guide for authors'. The main content area is divided into two sections: 'Latest issue' and 'About the journal'. The 'Latest issue' section highlights 'Volume 166' as 'In progress' for 'April 2022'. The 'About the journal' section describes the journal's objective: to communicate recent and projected advances in computer-based engineering techniques, covering fields like mechanical, aerospace, civil, and environmental engineering, with an emphasis on research and development leading to practical problem-solving. A 'View full aims & scope' link is provided. A 'FEEDBACK' button is located at the bottom right, and a footer bar reads 'SCIEDIRECT ADVANCES IN SOFTWARE ENGINEERING'.

IEEE Computing... Data Scien... Aims & sco... IEEE Trans... Página prin... Advances i...

ScienceDirect

Advances in Engineering Software

Supports open access

10.1	4.141
CiteScore	Impact Factor

Menu Search in this journal Submit your article ↗ Guide for authors ↗

Latest issue

Volume 166

In progress
April 2022

About the journal

The objective of this journal is to communicate recent and projected advances in **computer-based engineering** techniques. The fields covered include **mechanical, aerospace, civil** and **environmental engineering**, with an emphasis on research and development leading to practical problem-solving.

The ...

[View full aims & scope](#)

FEEDBACK

SCIEDIRECT ADVANCES IN SOFTWARE ENGINEERING

Desarrollo de Software Científico - Técnico

Escenario

Se han creado y desarrollado multitud de **publicaciones indexadas y revisadas por pares**, orientadas a:

- **Divulgar los desarrollos** concretos de software, su gestión (implementación web, distribución, ...) y aplicaciones.
- Valorar la **contribución científica** de avances en los propios procesos de computación (nuevos algoritmos, estructuras, procesos básicos, etc.).

Aims & Scope



The objective of this journal is to communicate recent and projected advances in **computer-based engineering** techniques. The fields covered include **mechanical, aerospace, civil and environmental engineering**, with an emphasis on research and development leading to practical problem-solving.

The scope of the journal includes:

- Innovative computational strategies and numerical algorithms for large-scale engineering problems
- Analysis and simulation techniques and systems
- Model and mesh generation
- Control of the accuracy, stability and efficiency of computational process
- Exploitation of new computing environments (eg distributed heterogeneous and collaborative computing)
- Advanced visualization techniques, virtual environments and prototyping
- Applications of AI, knowledge-based systems, computational intelligence, including fuzzy logic, neural networks and evolutionary computations
- Application of object-oriented technology to engineering problems
- Intelligent human computer interfaces
- Design automation, multidisciplinary design and optimization
- CAD, CAE and integrated process and product development systems
- Quality and reliability

(I - 6/6) Formulación del Paradigma de Trabajo

Desarrollo de Software Científico - Técnico

Paradigma

El *Desarrollo de Software Científico - Técnico* es una **actividad necesaria y sistemática**, en la que se emplean importantes recursos de tiempo, esfuerzo y capacidad creativa. Adicionalmente, es una **actividad transversal** a muchos de los trabajos en I+D. Su evolución en términos de exigencia, trascendencia y campos de aplicación, discurre paralela al desarrollo tecnológico.

En consonancia con el enfoque general adoptado para nuestra actividad, resulta acorde la atención a la **demandas de la sociedad**, que tiende hacia la acreditación de la competencia, el desarrollo del trabajo bajo parámetros homologables, la innovación y la mejora continua de la actividad, así como la exploración científica, el retorno de los recursos empleados y la divulgación del conocimiento generado.

Las diferentes tecnologías actuales para el desarrollo y la gestión integral de los proyectos de software, de gran solvencia, robustez, versatilidad y nivel de estandarización, son actualmente herramientas de uso abierto y generalizado, que convierten el *desarrollo profesional de Software Científico - Técnico* en un **trabajo asequible** a nuestra capacidad y recursos, y sus productos en **bienes tangibles**, capaces de aportar valor real en el contexto de una actividad concreta de I+D.

Desarrollo de Software Científico - Técnico

Paradigma

PROPUESTA 1: desarrollo de capacidad tecnológica

Contar con el recurso de **generar software...**

- con un código técnicamente correcto y completo.
- acorde con el estado del arte (incorporar nuevas herramientas)
- extensible (alto nivel de estandarización)
- validado (desarrollo y actualización de módulos test)
- documentado (proyecto documental)
- transmisible (distribuable y publicable)

... para **afrontar** los siguientes retos en el PVLab.

... para dar **soporte** a Procedimientos Técnicos existentes.

... para dar **apoyo** a Proyectos I+D.



**BRECHA
TECNOLÓGICA**



**CUMPLIMIENTO
REQUISITOS I/E**

PROPUESTA 2: afrontar el futuro en información C-T: **Data Science**

Explorar, aprovechar los recursos y **contribuir al avance** de...

- Herramientas existentes para el tratamiento, representación y modelado de datos:
 - **Hadoop** (Big Data), **scikit-learn** (Machine Learning), **keras** (Deep Learning), **Seaborn** (Graphics), etc.
- Big Data: potencia y eficiencia en la gestión de los datos.
- Machine Learning: Esquemas de cumplimiento de tareas (**T**, tasks) y mejora del desempeño (**P**, performance) acorde a métricas, por medio de la experiencia (**E**, experience). *Bases, prácticas modernas e Investigación.*

Bases: álgebra lineal, probabilidad y teoría de la información, computación numérica. **Prácticas modernas:** Deep Feedforward Networks, Regularization, Optimization, Convolutional Networks, Sequence Modeling (Recurrent and Recursive Nets). **Investigación:** Linear Factor Models, Autoencoders, Representation Learning, Structured Probabilistic Models, Monte Carlo methods, Inference, Partition Functions, Deep Generative models.
- Otros enfoques y recursos de la Inteligencia Artificial.

... para obtener y divulgar **nuevos enfoques** y **métodos** en el ámbito de la caracterización, modelado de componentes y simulación en ESFV (EERR).



**AMPLIAR Y FORTALECER
CONOCIMIENTO**



**PREPARAR Y AFRONTAR
EL FUTURO**

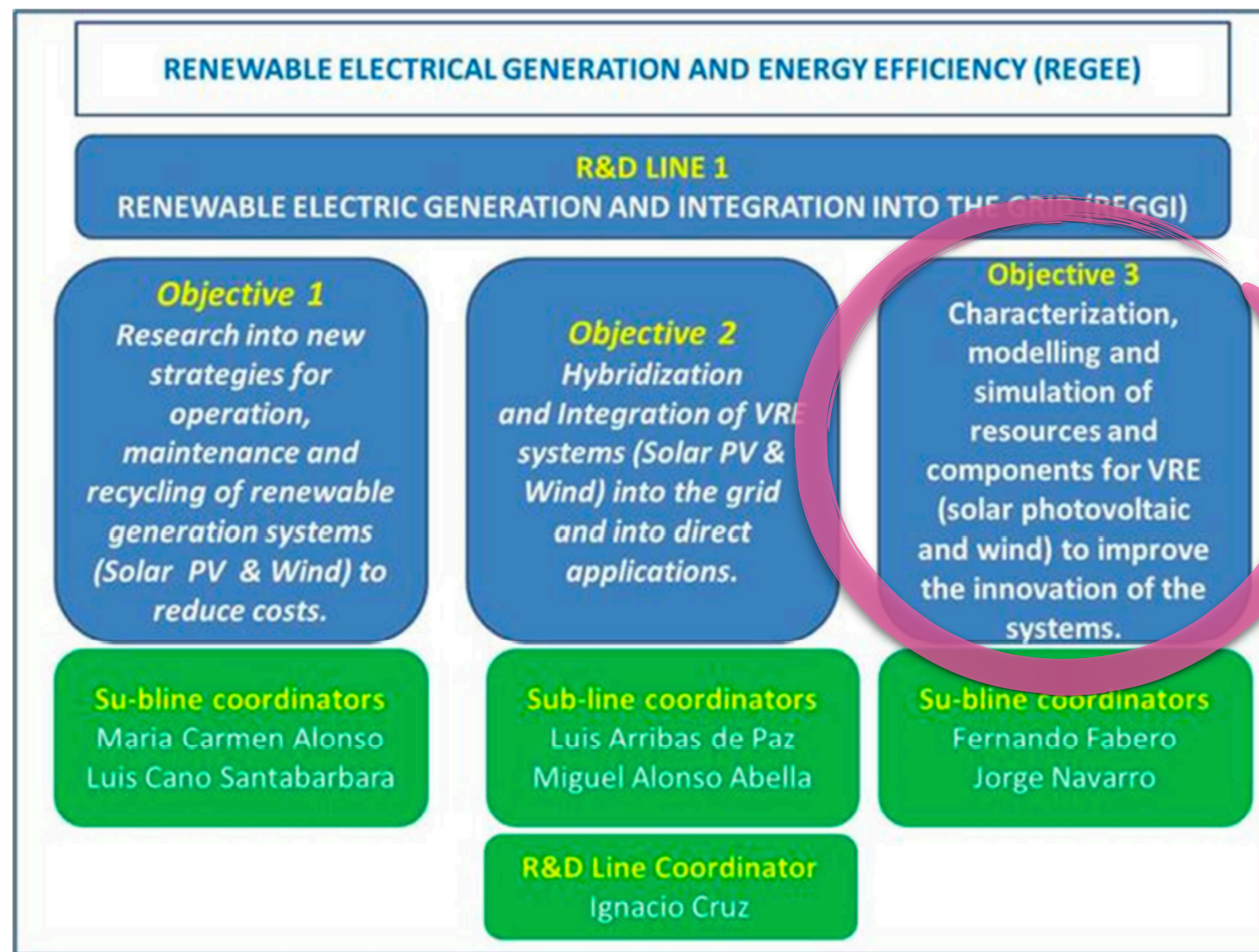
Desarrollo de Software Científico - Técnico

Paradigma

El Punto de partida: buscando **encaje externo**...



(aunque no debe ser la cola quien mueva el perro...)



Línea I+D Ciemat: RENEWABLE ELECTRICAL GENERATION AND ENERGY EFFICIENCY (REGEE)



PLAN ESTRATÉGICO DEL CIEMAT

Desarrollo de Software Científico - Técnico

Paradigma

Coherencia interna: desarrollo de software: ¿*Dónde Estamos?*, ¿*Qué somos?*, ¿*Qué queremos hacer?*



PARTE II: RECURSOS BÁSICOS PARA EL DESARROLLO DE SOFTWARE

Contenido

Parte II - Recursos para el Desarrollo de Software

1. Python
2. Instalación y Entornos
3. Bases de Programación
4. Proyecto Documental
5. Distribución y Publicación

Apéndices

(II - 1/5) El Lenguaje de Programación Python

Recursos para el Desarrollo de Software

Python

¿Qué es Python?

Lenguaje de programación creado por el informático holandés **Guido Van Rossum** en diciembre de 1989, mientras trabajaba en el *Centrum voor Wiskunde & Informatica* (CWI) en Amsterdam, llamado así en honor a la serie *Monty Python's Flying Circus*.

Entre 1982-1986 contribuyó al desarrollo del lenguaje ABC. En 1986 se une al equipo de desarrollo del sistema operativo *Amoeba*, que precisaba un lenguaje de *scripting*. Durante las navidades de 1989 comenzó a implementar en **C** el primer intérprete de Python, utilizado en el CWI para uso interno en 1990. En 1991 se publica la versión 0.9.0. en *alt.sources*. En 1994 se crea el primer grupo de discusión sobre Python, y en 1995 nace el sitio oficial **www.python.org**. Actualmente el sitio lo gestiona la Python Software Foundation (est. 2001), presidida aún por Guido Van Rossum, que ostenta el "título" de Benevolent Dictator for Life (BDFL) en materia del lenguaje.

Fuente: El Gran Libro de Python. Marco Buttu. Editorial Marcombo. Primera Edición, 2016.

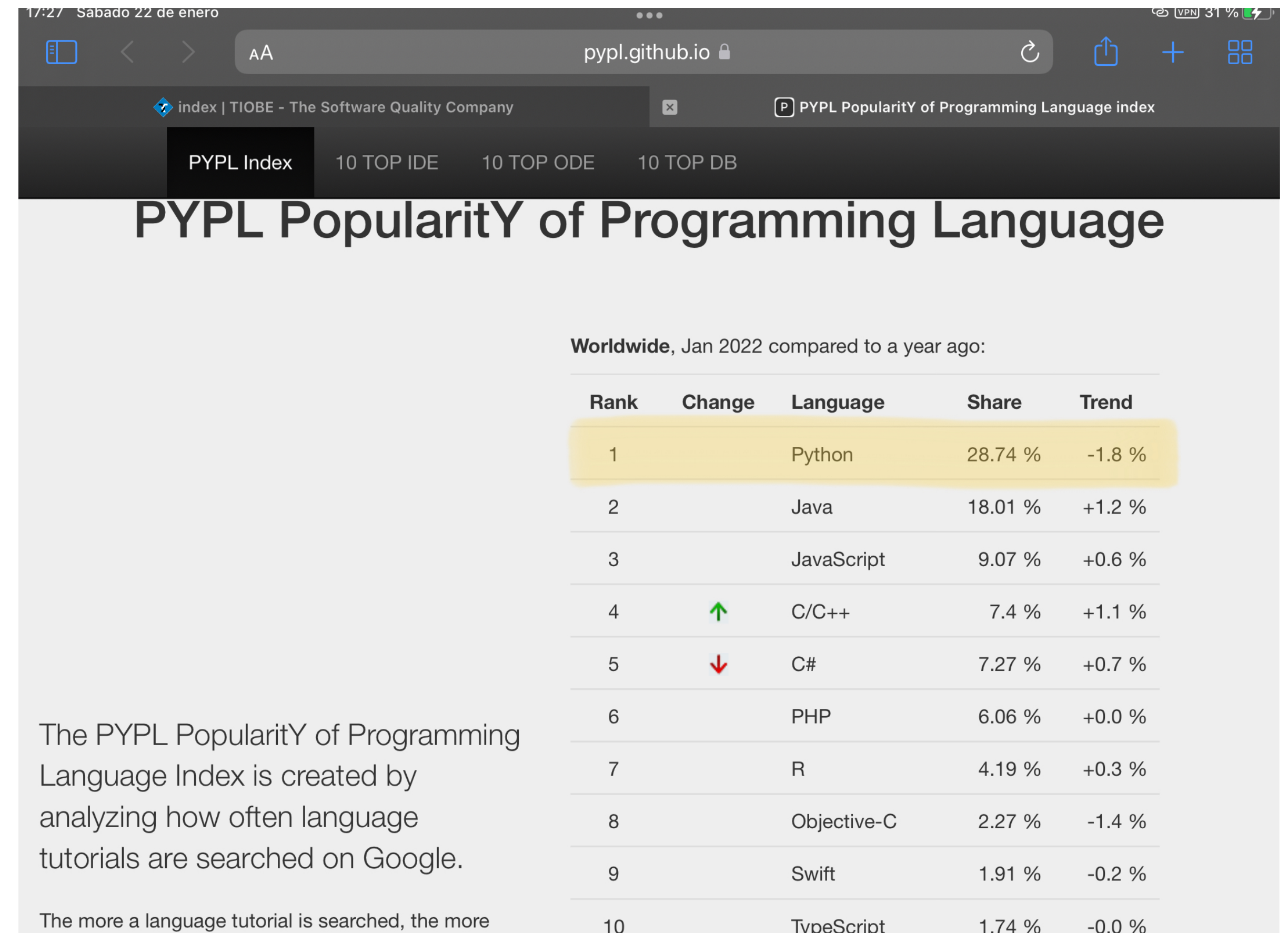
Recursos para el Desarrollo de Software

Python

El uso y popularidad del lenguaje Python lo ha situado a la cabeza de los lenguajes **más buscados** (ref. índice PYPL) y **más utilizados** (ref. índice TIOBE).

Por su potencia, su desarrollo normativo y su uso masivo, es un lenguaje llamado a jugar un importante papel como **lenguaje estándar** en entornos científico-técnicos, sin que otro nuevos lenguajes (R, Julia, ...), presenten competencia más allá de aplicaciones muy concretas.

El uso de Python es por lo tanto una **apuesta lógica de futuro**.



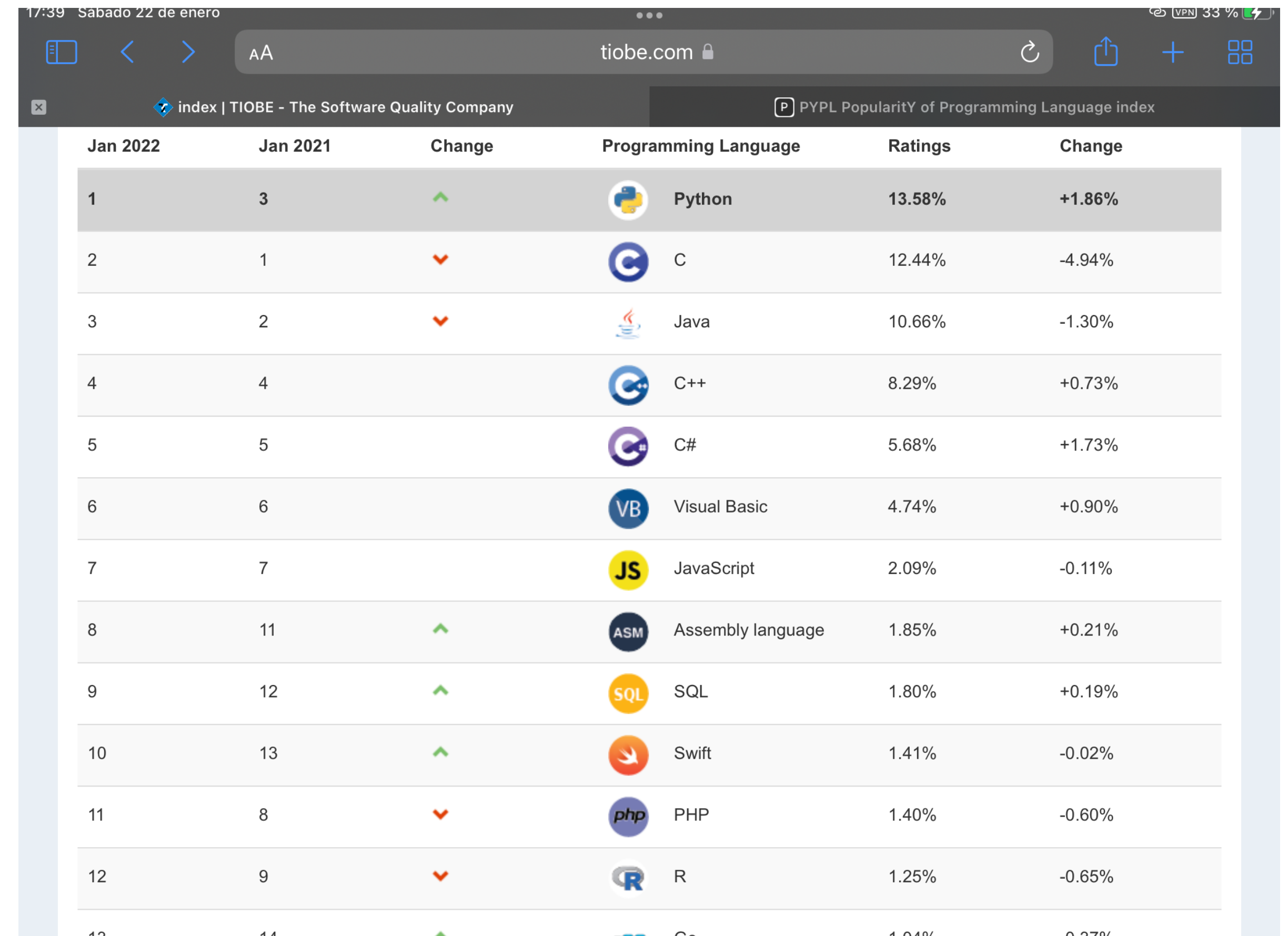
Recursos para el Desarrollo de Software

Python

El uso y popularidad del lenguaje Python lo ha situado a la cabeza de los lenguajes **más buscados** (ref. índice PYPL) y **más utilizados** (ref. índice TIOBE).

Por su potencia, su desarrollo normativo y su uso masivo, es un lenguaje llamado a jugar un importante papel como **lenguaje estándar** en entornos científico-técnicos, sin que otros nuevos lenguajes (R, Julia, ...), presenten competencia más allá de aplicaciones muy concretas.

El uso de Python es por lo tanto una **apuesta lógica de futuro**.



Jan 2022	Jan 2021	Change	Programming Language	Ratings	Change
1	3	▲	 Python	13.58%	+1.86%
2	1	▼	 C	12.44%	-4.94%
3	2	▼	 Java	10.66%	-1.30%
4	4		 C++	8.29%	+0.73%
5	5		 C#	5.68%	+1.73%
6	6		 Visual Basic	4.74%	+0.90%
7	7		 JavaScript	2.09%	-0.11%
8	11	▲	 Assembly language	1.85%	+0.21%
9	12	▲	 SQL	1.80%	+0.19%
10	13	▲	 Swift	1.41%	-0.02%
11	8	▼	 PHP	1.40%	-0.60%
12	9	▼	 R	1.25%	-0.65%
13	14	▲	 Go	1.04%	-0.37%

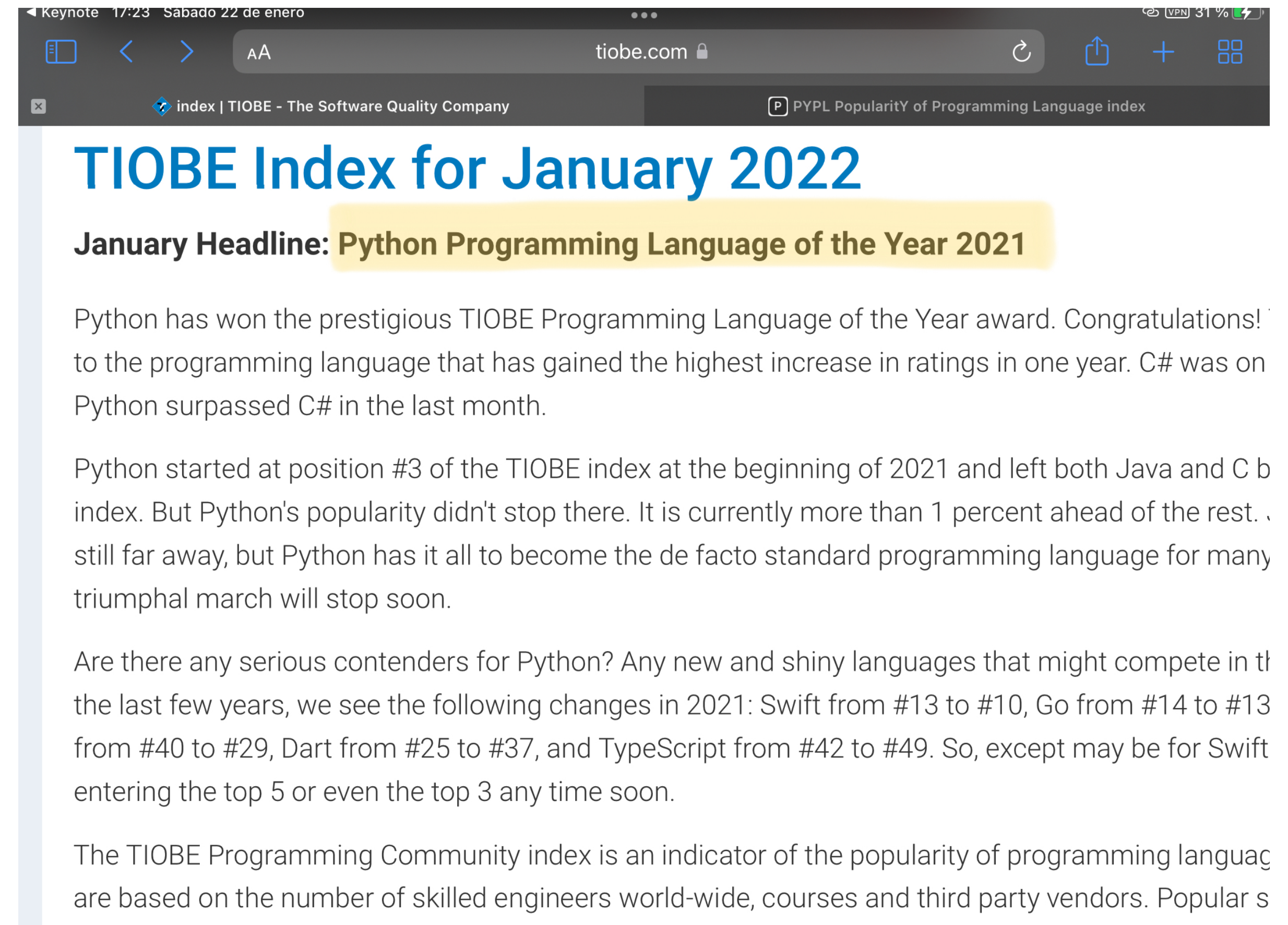
Recursos para el Desarrollo de Software

Python

El uso y popularidad del lenguaje Python lo ha situado a la cabeza de los lenguajes **más buscados** (ref. índice PYPL) y **más utilizados** (ref. índice TIOBE).

Por su potencia, su desarrollo normativo y su uso masivo, es un lenguaje llamado a jugar un importante papel como **lenguaje estándar** en entornos científico-técnicos, sin que otros nuevos lenguajes (R, Julia, ...), presenten competencia más allá de aplicaciones muy concretas.

El uso de Python es por lo tanto una **apuesta lógica de futuro**.



The screenshot shows a mobile browser view of the TIOBE Index website. The address bar shows 'tio.be.com'. The page title is 'TIOBE Index for January 2022'. A yellow highlight is under the 'January Headline: Python Programming Language of the Year 2021'. The main text celebrates Python's win, noting it surpassed C# in the last month. It also mentions Python's position at #3 at the start of 2021 and its current lead. A section discusses other contenders like Swift, Go, Dart, and TypeScript. The footer explains that the index is based on the number of skilled engineers, courses, and third-party vendors.

Keynote 17:23 Sabado 22 de enero

AA tio.be.com

index | TIOBE - The Software Quality Company PYPL Popularity of Programming Language index

TIOBE Index for January 2022

January Headline: Python Programming Language of the Year 2021

Python has won the prestigious TIOBE Programming Language of the Year award. Congratulations! to the programming language that has gained the highest increase in ratings in one year. C# was on Python surpassed C# in the last month.

Python started at position #3 of the TIOBE index at the beginning of 2021 and left both Java and C b index. But Python's popularity didn't stop there. It is currently more than 1 percent ahead of the rest. , still far away, but Python has it all to become the de facto standard programming language for many triumphal march will stop soon.

Are there any serious contenders for Python? Any new and shiny languages that might compete in tl the last few years, we see the following changes in 2021: Swift from #13 to #10, Go from #14 to #13 from #40 to #29, Dart from #25 to #37, and TypeScript from #42 to #49. So, except may be for Swift entering the top 5 or even the top 3 any time soon.

The TIOBE Programming Community index is an indicator of the popularity of programming languag are based on the number of skilled engineers world-wide, courses and third party vendors. Popular s

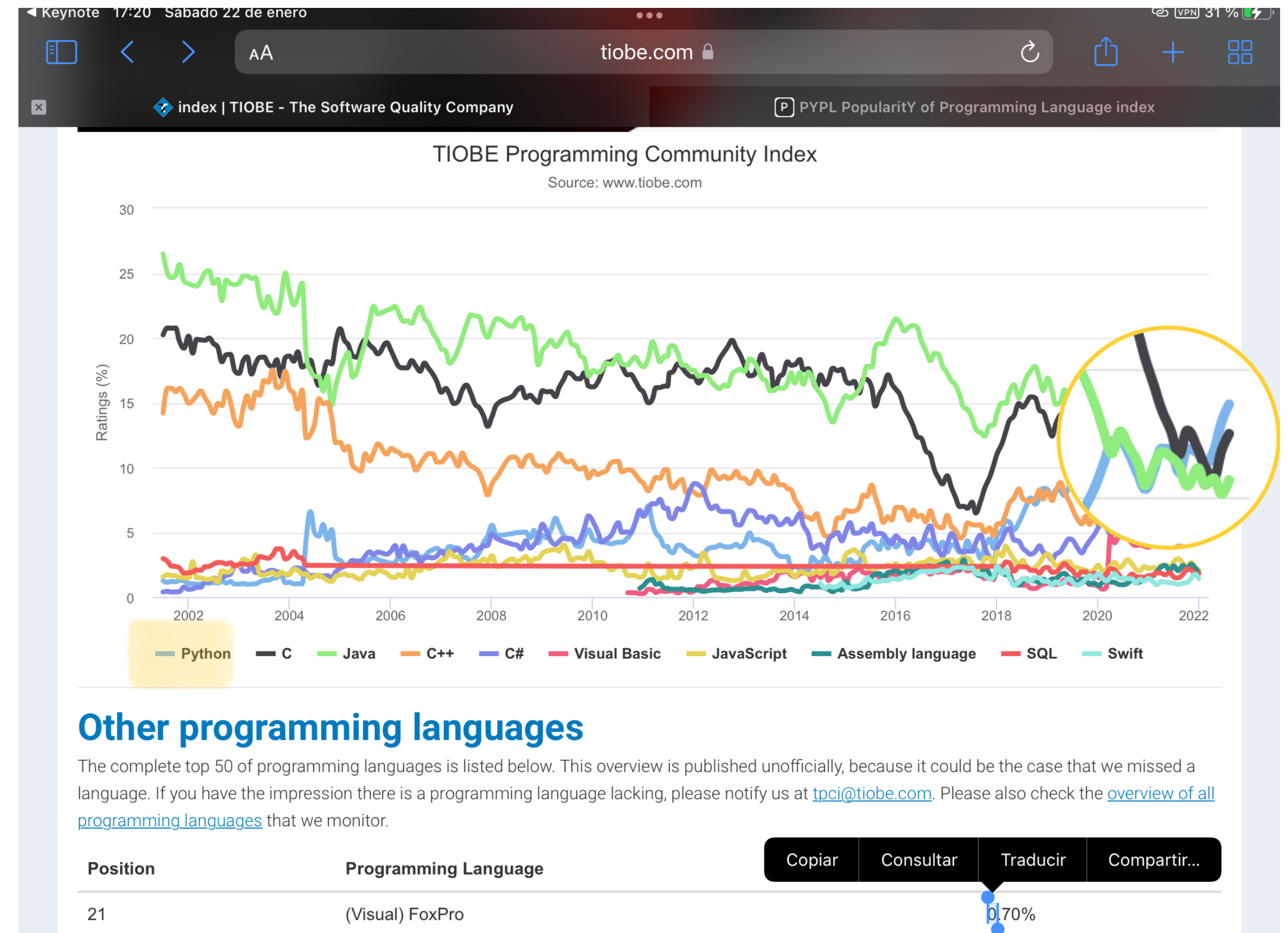
Recursos para el Desarrollo de Software

Python

El uso y popularidad del lenguaje Python lo ha situado a la cabeza de los lenguajes **más buscados** (ref. índice PYPL) y **más utilizados** (ref. índice TIOBE).

Por su potencia, su desarrollo normativo y su uso masivo, es un lenguaje llamado a jugar un importante papel como **lenguaje estándar** en entornos científico-técnicos, sin que otro nuevos lenguajes (R, Julia, ...), presenten competencia más allá de aplicaciones muy concretas.

El uso de Python es por lo tanto una **apuesta lógica de futuro**.



Recursos para el Desarrollo de Software

Python

¿Qué tipo de lenguaje es Python?

Código abierto: Licencia de uso libre Python (PSF). Elementos visibles, no orientado a software propietario u opaco.

Multi-paradigma: imperativo, funcional, orientado a objetos...

Alto nivel: implementado sobre un lenguaje de más bajo nivel (C).

Tipado dinámico: el tipo se define al operar el objeto, no se declara, aunque se puede (y es recomendable) anotar en la firma de la función.

Tipado fuerte: por ejemplo, 1 (**int**) es distinto de '1' (**str**).

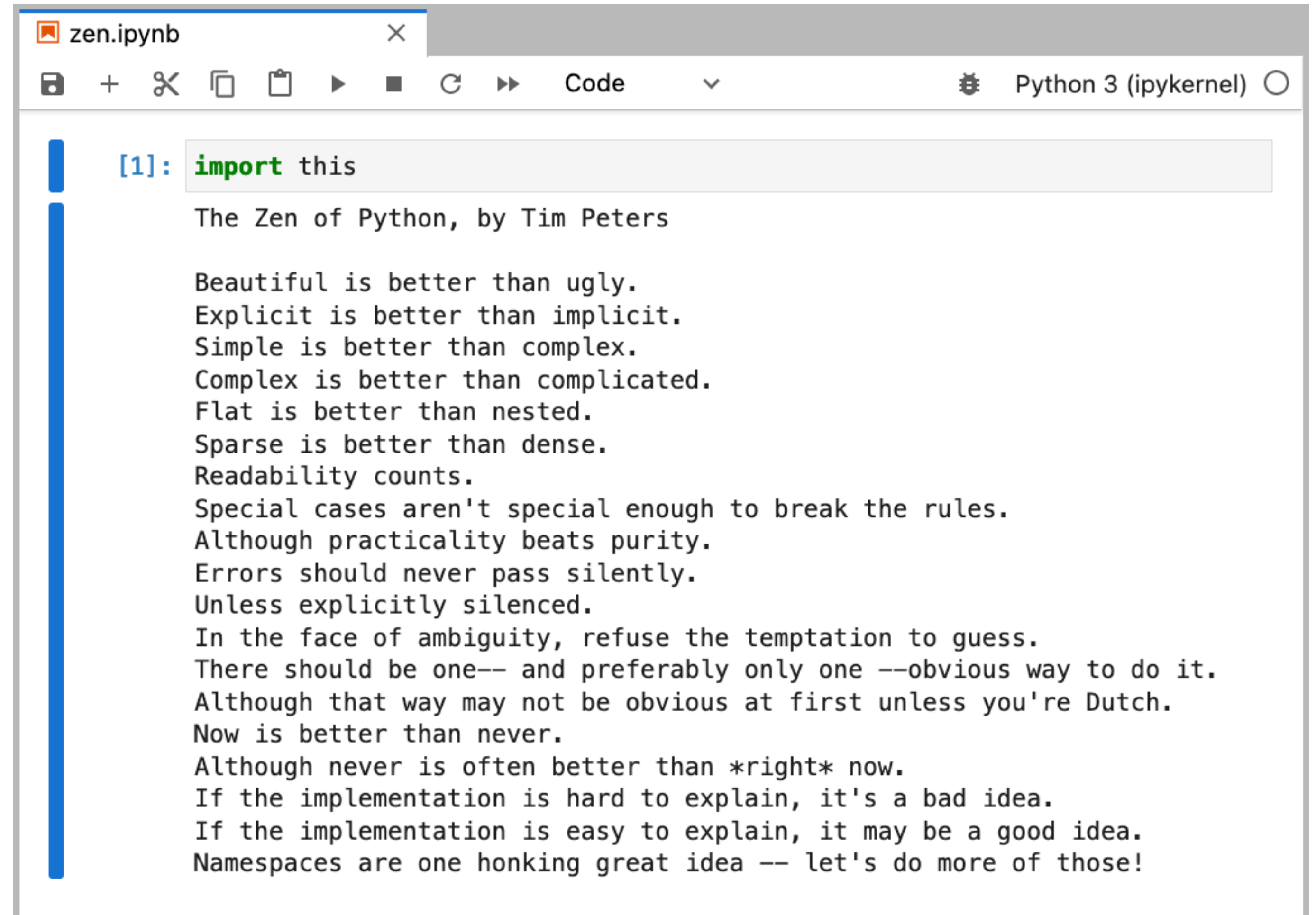
Buena curva de aprendizaje: fácil comienzo y evolución.

Desarrollo comunitario muy activo: del propio lenguaje (PEPs) y librerías.

Compilado: programas compilados (convertidos a *bytecode*, .pyc, usualmente guardados en `__pycache__`).

Multi-plataforma: no restringido a plataformas particulares (Windows, MacOS, Linux, iOS, Android, ...).

Tabulación obligada, recomendable principio "*duck-typing*", recomendables descripciones de módulo y de funciones y clases, con encabezado y ejemplos. Recomendables módulos test. Recomendable propósito unitario de las estructuras (una cosa, bien hecha). Uso específico de mayúsculas, guiones bajos, etc.



```
[1]: import this
```

The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!

Recursos para el Desarrollo de Software Python

Python Enhancement Proposals (PEPs)

Las PEPs y los Procesos relacionados constituyen un sistema reglado por la PSF, trazable y colaborativo, para:

Elevar, discutir, acordar, aceptar, rechazar, modificar o actualizar **normas, información** y **buenas prácticas** en relación al diseño y uso del lenguaje.

Se contemplan tanto **PEPs informativas** (I - Informational PEP) como **PEPs normativas** (S - Standards Track PEP), además de las **PEPs en curso** (P - Process PEP).

Ejemplos:

PEP 8 -- Style Guide for Python Code.

PEP 257 -- Doctring Conventions.

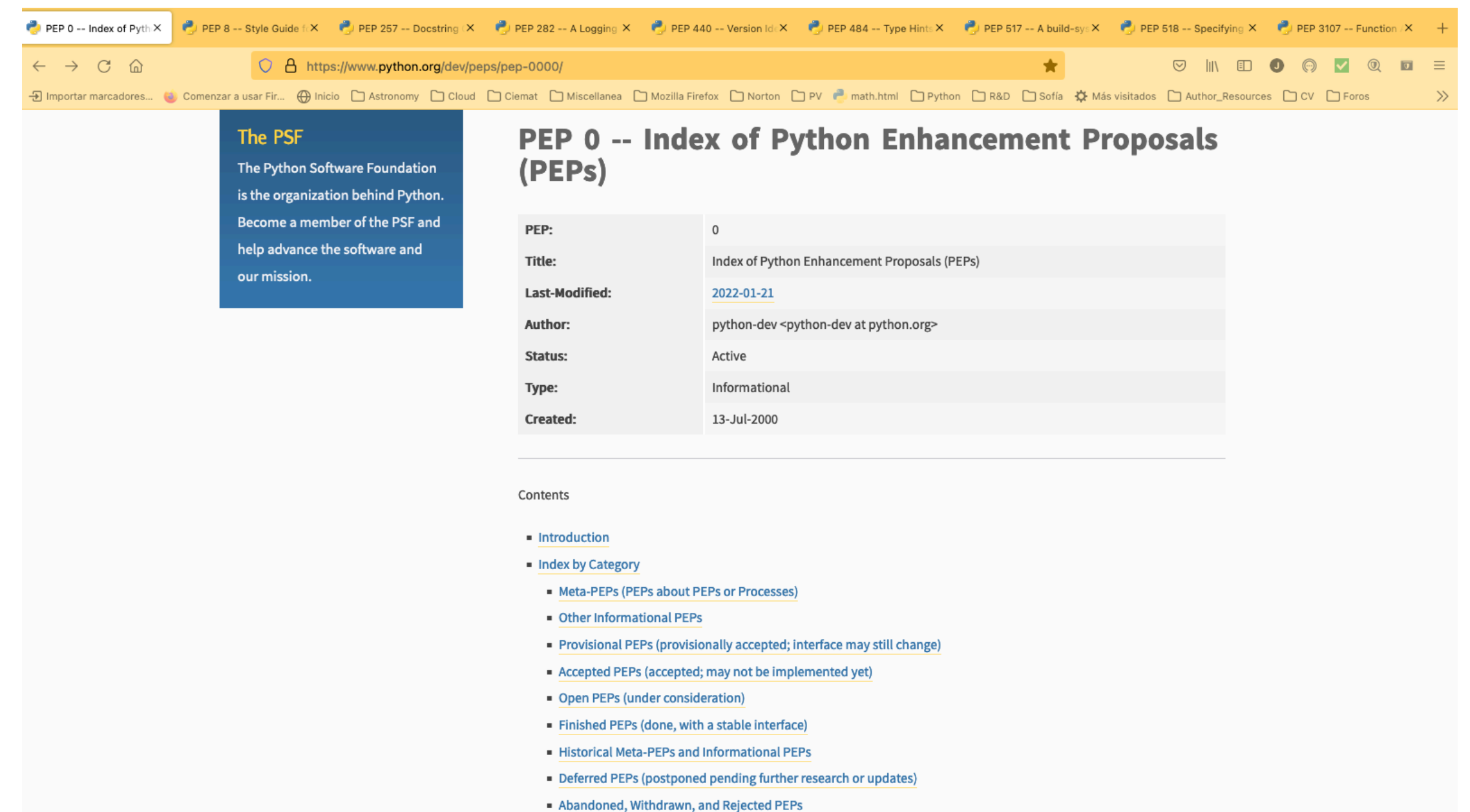
PEP 282 -- A Logging System.

PEP 440 -- Version Identification and Dependency Specification.

PEP 484 -- Type Hints.

PEP 517 -- A build-system independent format for source trees.

PEP 3107 -- Function Annotations.



Recursos para el Desarrollo de Software Python

Python Enhancement Proposals (PEPs)

Las PEPs y los Procesos relacionados constituyen un sistema reglado por la PSF, trazable y colaborativo, para:

Elevar, discutir, acordar, aceptar, rechazar, modificar o actualizar **normas, información** y **buenas prácticas** en relación al diseño y uso del lenguaje.

Se contemplan tanto **PEPs informativas** (I - Informational PEP) como **PEPs normativas** (S - Standards Track PEP), además de las **PEPs en curso** (P - Process PEP).

Ejemplos:

PEP 8 -- Style Guide for Python Code.

PEP 257 -- Doctring Conventions.

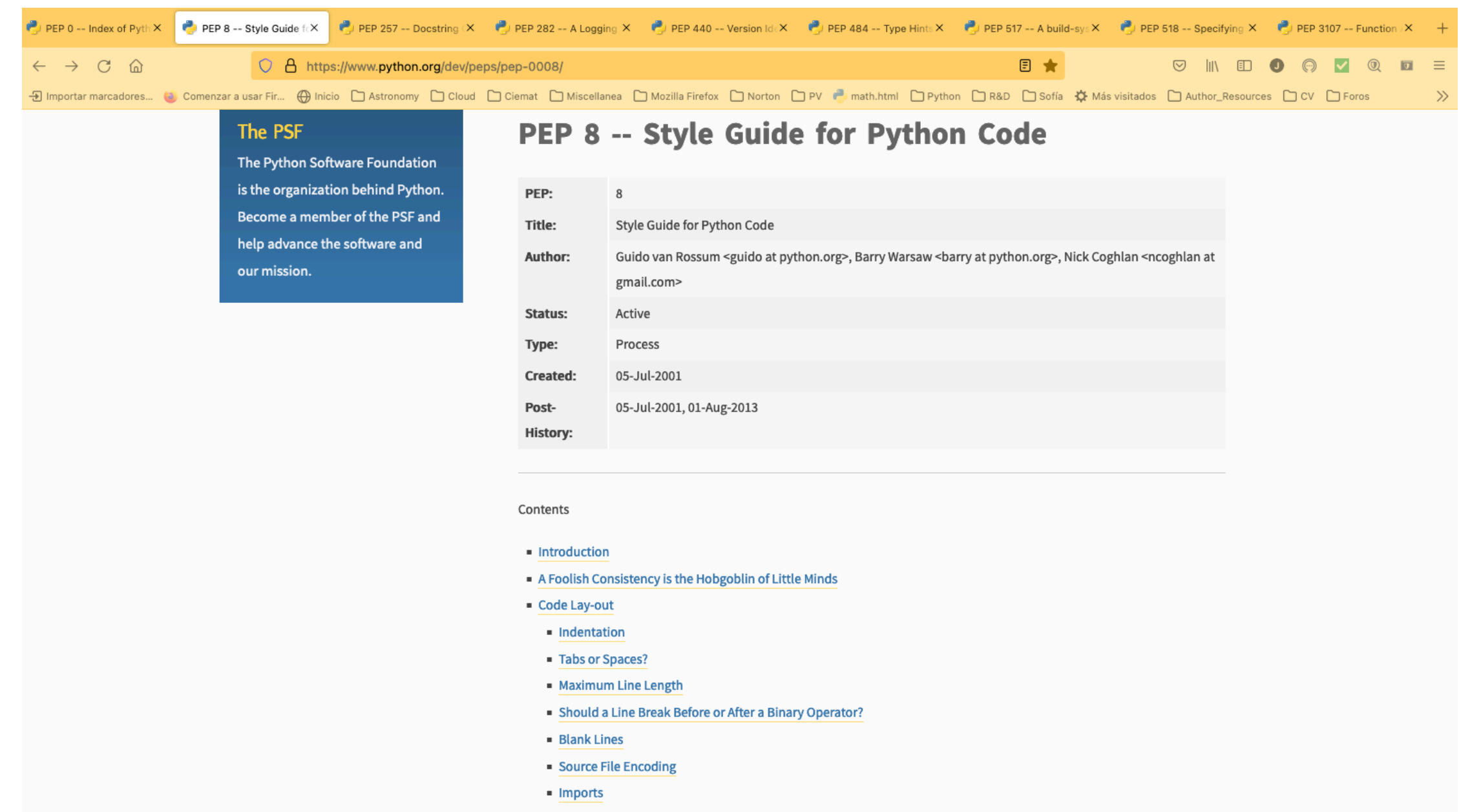
PEP 282 -- A Logging System.

PEP 440 -- Version Identification and Dependency Specification.

PEP 484 -- Type Hints.

PEP 517 -- A build-system independent format for source trees.

PEP 3107 -- Function Annotations.



Recursos para el Desarrollo de Software

Python

Python Enhancement Proposals (PEPs)

Las PEPs y los Procesos relacionados constituyen un sistema reglado por la PSF, trazable y colaborativo, para:

Elevar, discutir, acordar, aceptar, rechazar, modificar o actualizar **normas, información** y **buenas prácticas** en relación al diseño y uso del lenguaje.

Se contemplan tanto **PEPs informativas** (I - Informational PEP) como **PEPs normativas** (S - Standards Track PEP), además de las **PEPs en curso** (P - Process PEP).

Ejemplos:

PEP 8 -- Style Guide for Python Code.

PEP 257 -- Doctring Conventions.

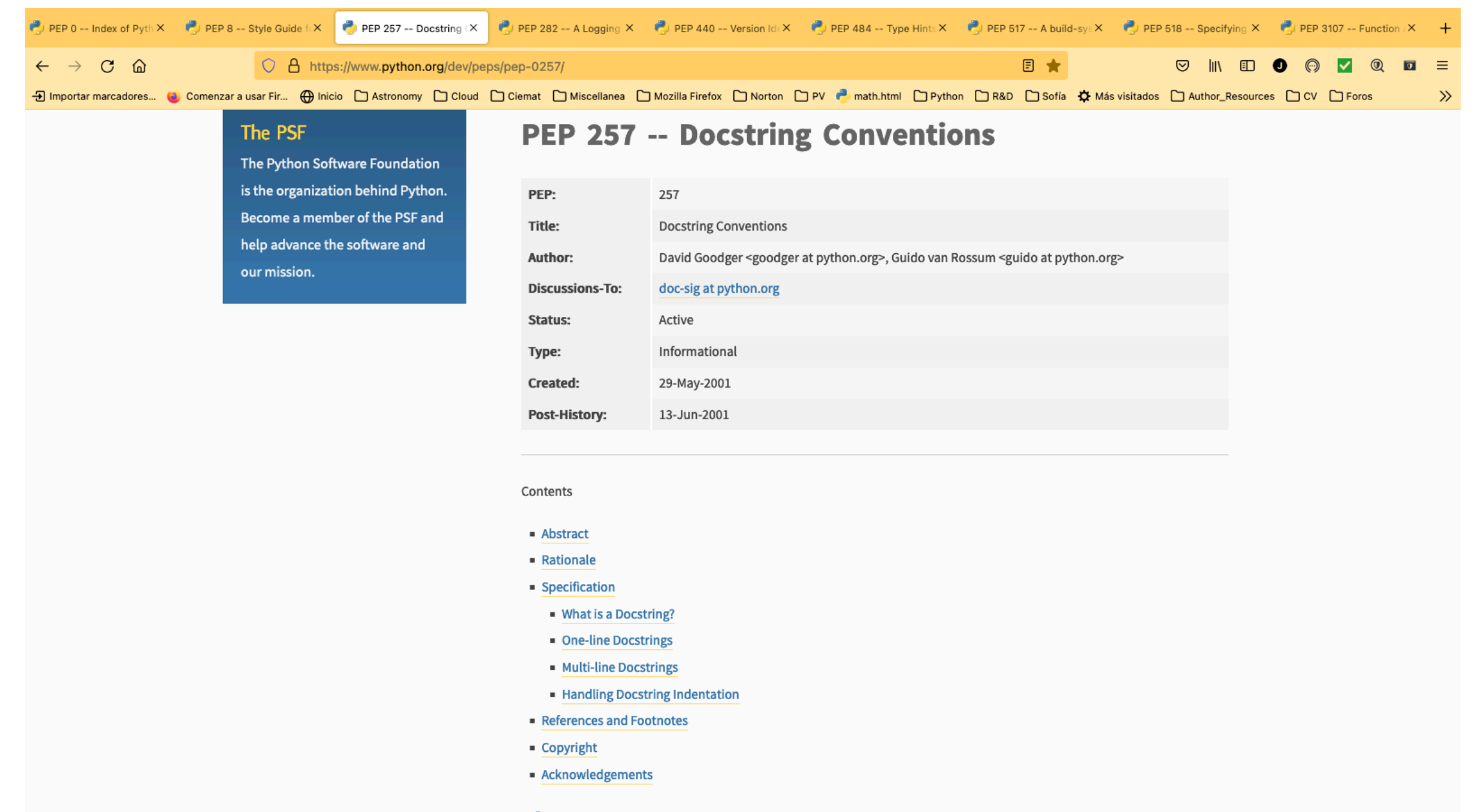
PEP 282 -- A Logging System.

PEP 440 -- Version Identification and Dependency Specification.

PEP 484 -- Type Hints.

PEP 517 -- A build-system independent format for source trees.

PEP 3107 -- Function Annotations.



Recursos para el Desarrollo de Software Python

Python Enhancement Proposals (PEPs)

Las PEPs y los Procesos relacionados constituyen un sistema reglado por la PSF, trazable y colaborativo, para:

Elevar, discutir, acordar, aceptar, rechazar, modificar o actualizar **normas, información** y **buenas prácticas** en relación al diseño y uso del lenguaje.

Se contemplan tanto **PEPs informativas** (I - Informational PEP) como **PEPs normativas** (S - Standards Track PEP), además de las **PEPs en curso** (P - Process PEP).

Ejemplos:

PEP 8 -- Style Guide for Python Code.

PEP 257 -- Docstring Conventions.

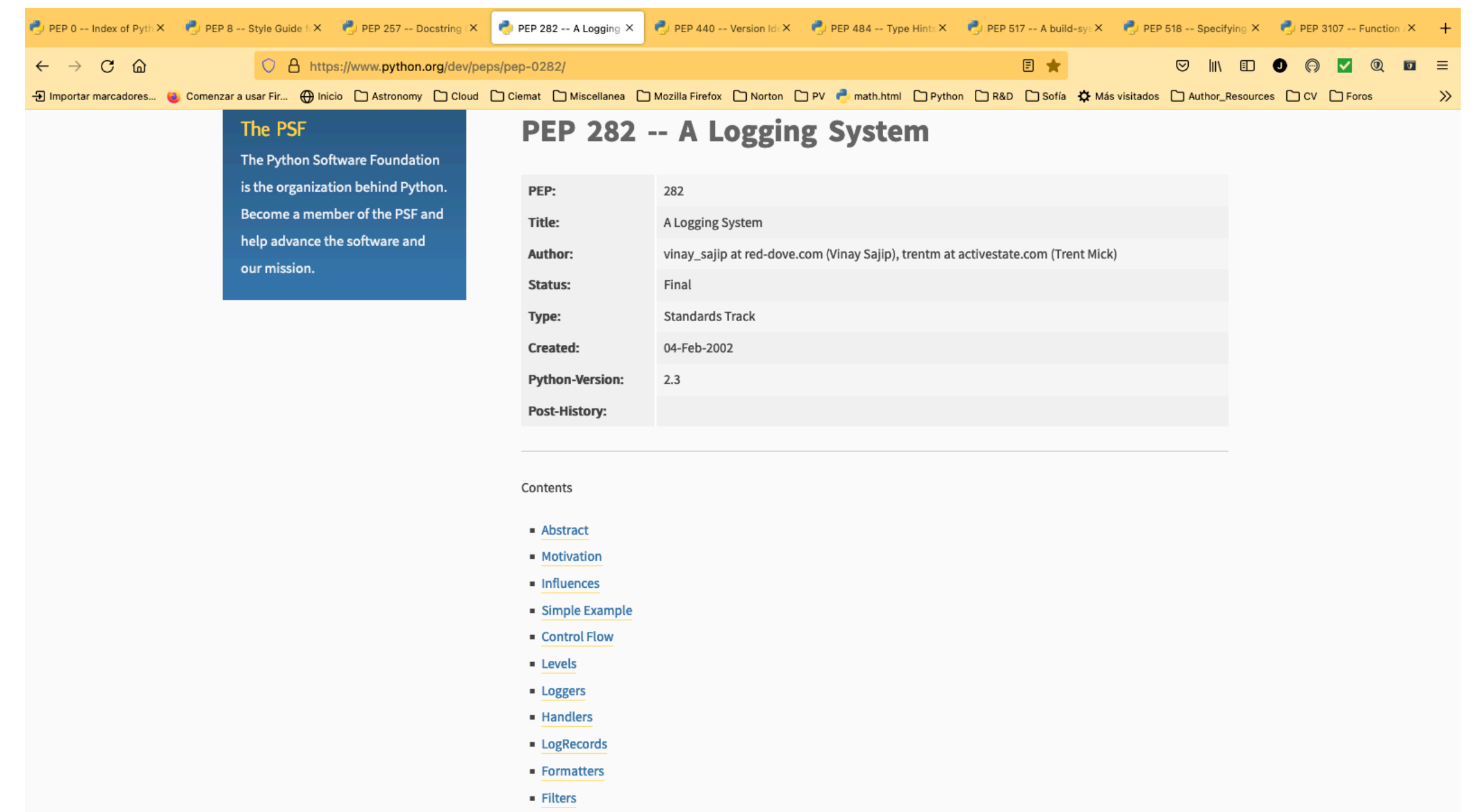
PEP 282 -- A Logging System.

PEP 440 -- Version Identification and Dependency Specification.

PEP 484 -- Type Hints.

PEP 517 -- A build-system independent format for source trees.

PEP 3107 -- Function Annotations.



Recursos para el Desarrollo de Software Python

Python Enhancement Proposals (PEPs)

Las PEPs y los Procesos relacionados constituyen un sistema reglado por la PSF, trazable y colaborativo, para:

Elevar, discutir, acordar, aceptar, rechazar, modificar o actualizar **normas, información y buenas prácticas** en relación al diseño y uso del lenguaje.

Se contemplan tanto **PEPs informativas** (I - Informational PEP) como **PEPs normativas** (S - Standards Track PEP), además de las **PEPs en curso** (P - Process PEP).

Ejemplos:

PEP 8 -- Style Guide for Python Code.

PEP 257 -- Docstring Conventions.

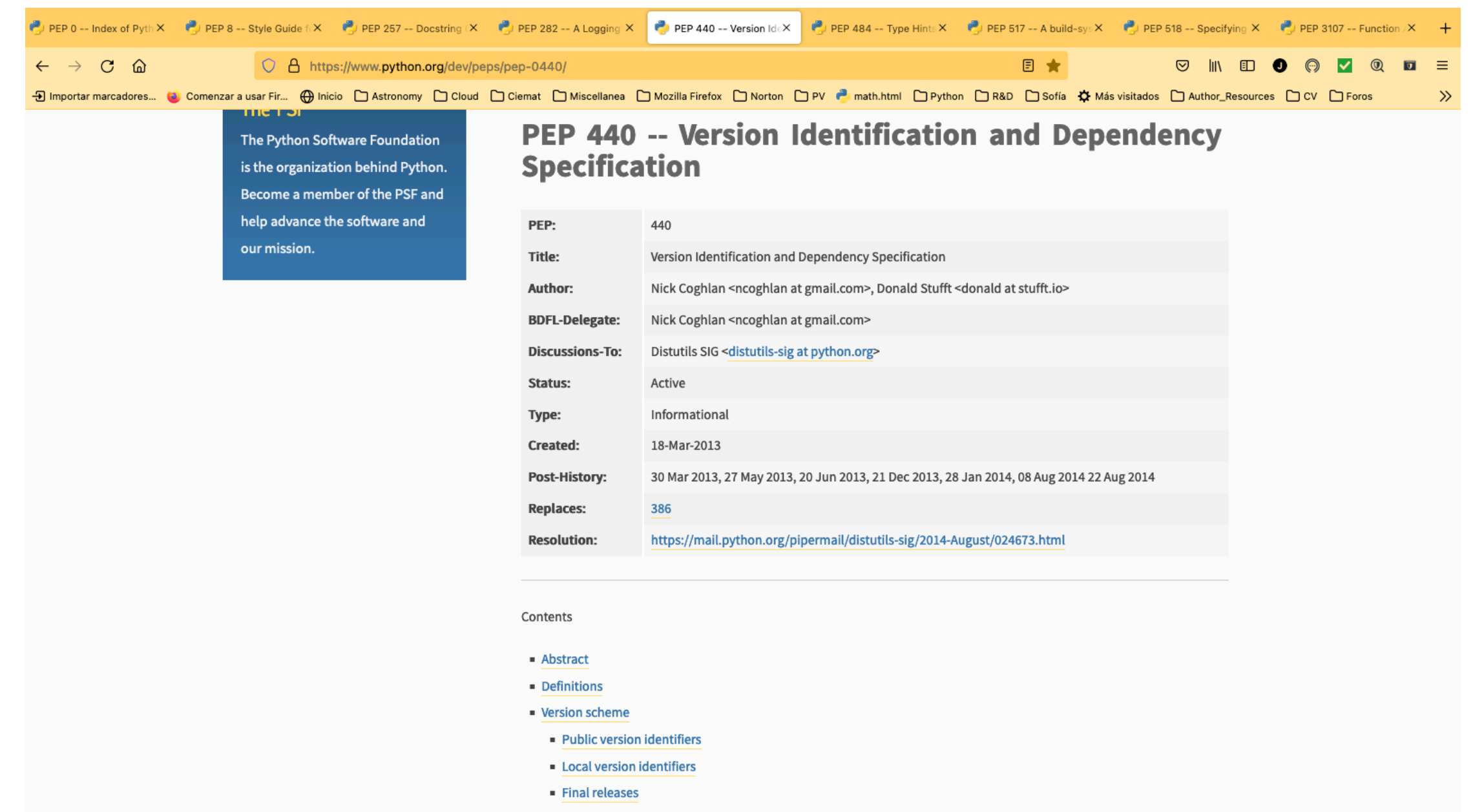
PEP 282 -- A Logging System.

PEP 440 -- Version Identification and Dependency Specification.

PEP 484 -- Type Hints.

PEP 517 -- A build-system independent format for source trees.

PEP 3107 -- Function Annotations.



The screenshot shows a web browser with multiple tabs open, including 'PEP 0 -- Index of Pyth...', 'PEP 8 -- Style Guide', 'PEP 257 -- Docstring', 'PEP 282 -- A Logging', and 'PEP 440 -- Version Id'. The active tab is 'PEP 440 -- Version Id'. The browser address bar shows 'https://www.python.org/dev/peps/pep-0440/'. The page content includes a sidebar with the Python Software Foundation logo and a main section titled 'PEP 440 -- Version Identification and Dependency Specification'. Below the title is a table with the following information:

PEP:	440
Title:	Version Identification and Dependency Specification
Author:	Nick Coghlan <ncoghlan at gmail.com>, Donald Stufft <donald at stufft.io>
BDFL-Delegate:	Nick Coghlan <ncoghlan at gmail.com>
Discussions-To:	Distutils SIG <distutils-sig at python.org>
Status:	Active
Type:	Informational
Created:	18-Mar-2013
Post-History:	30 Mar 2013, 27 May 2013, 20 Jun 2013, 21 Dec 2013, 28 Jan 2014, 08 Aug 2014 22 Aug 2014
Replaces:	386
Resolution:	https://mail.python.org/pipermail/distutils-sig/2014-August/024673.html

Below the table is a 'Contents' section with a list of links: Abstract, Definitions, Version scheme, Public version identifiers, Local version identifiers, Final releases, and Back releases.

Recursos para el Desarrollo de Software Python

Python Enhancement Proposals (PEPs)

Las PEPs y los Procesos relacionados constituyen un sistema reglado por la PSF, trazable y colaborativo, para:

Elevar, discutir, acordar, aceptar, rechazar, modificar o actualizar **normas, información** y **buenas prácticas** en relación al diseño y uso del lenguaje.

Se contemplan tanto **PEPs informativas** (I - Informational PEP) como **PEPs normativas** (S - Standards Track PEP), además de las **PEPs en curso** (P - Process PEP).

Ejemplos:

PEP 8 -- Style Guide for Python Code.

PEP 257 -- Docstring Conventions.

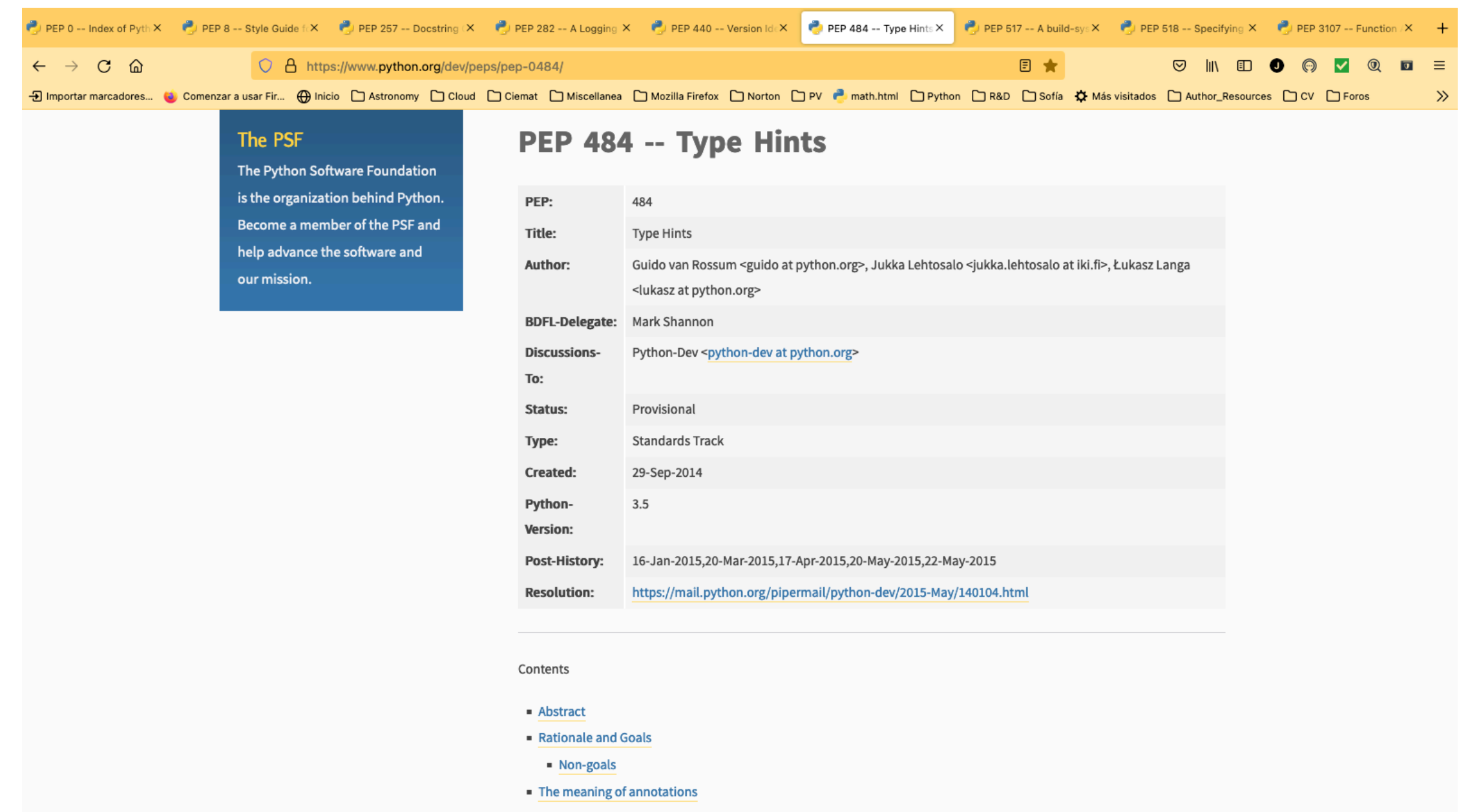
PEP 282 -- A Logging System.

PEP 440 -- Version Identification and Dependency Specification.

PEP 484 -- Type Hints.

PEP 517 -- A build-system independent format for source trees.

PEP 3107 -- Function Annotations.



The screenshot shows a web browser with multiple tabs open, including various PEPs. The active tab is 'PEP 484 -- Type Hints'. The browser address bar shows the URL 'https://www.python.org/dev/peps/pep-0484/'. The page content includes a sidebar for 'The PSF' (Python Software Foundation) and a main section for 'PEP 484 -- Type Hints'. The main section contains a table with the following information:

PEP:	484
Title:	Type Hints
Author:	Guido van Rossum <guido at python.org>, Jukka Lehtosalo <jukka.lehtosalo at iki.fi>, Łukasz Langa <lukasz at python.org>
BDFL-Delegate:	Mark Shannon
Discussions-To:	Python-Dev < python-dev at python.org >
Status:	Provisional
Type:	Standards Track
Created:	29-Sep-2014
Python-Version:	3.5
Post-History:	16-Jan-2015,20-Mar-2015,17-Apr-2015,20-May-2015,22-May-2015
Resolution:	https://mail.python.org/pipermail/python-dev/2015-May/140104.html

Below the table, there is a 'Contents' section with a list of links: Abstract, Rationale and Goals (with a sub-link for Non-goals), The meaning of annotations, and Type Definition Syntax.

Recursos para el Desarrollo de Software Python

Python Enhancement Proposals (PEPs)

Las PEPs y los Procesos relacionados constituyen un sistema reglado por la PSF, trazable y colaborativo, para:

Elevar, discutir, acordar, aceptar, rechazar, modificar o actualizar **normas, información y buenas prácticas** en relación al diseño y uso del lenguaje.

Se contemplan tanto **PEPs informativas** (I - Informational PEP) como **PEPs normativas** (S - Standards Track PEP), además de las **PEPs en curso** (P - Process PEP).

Ejemplos:

PEP 8 -- Style Guide for Python Code.

PEP 257 -- Docstring Conventions.

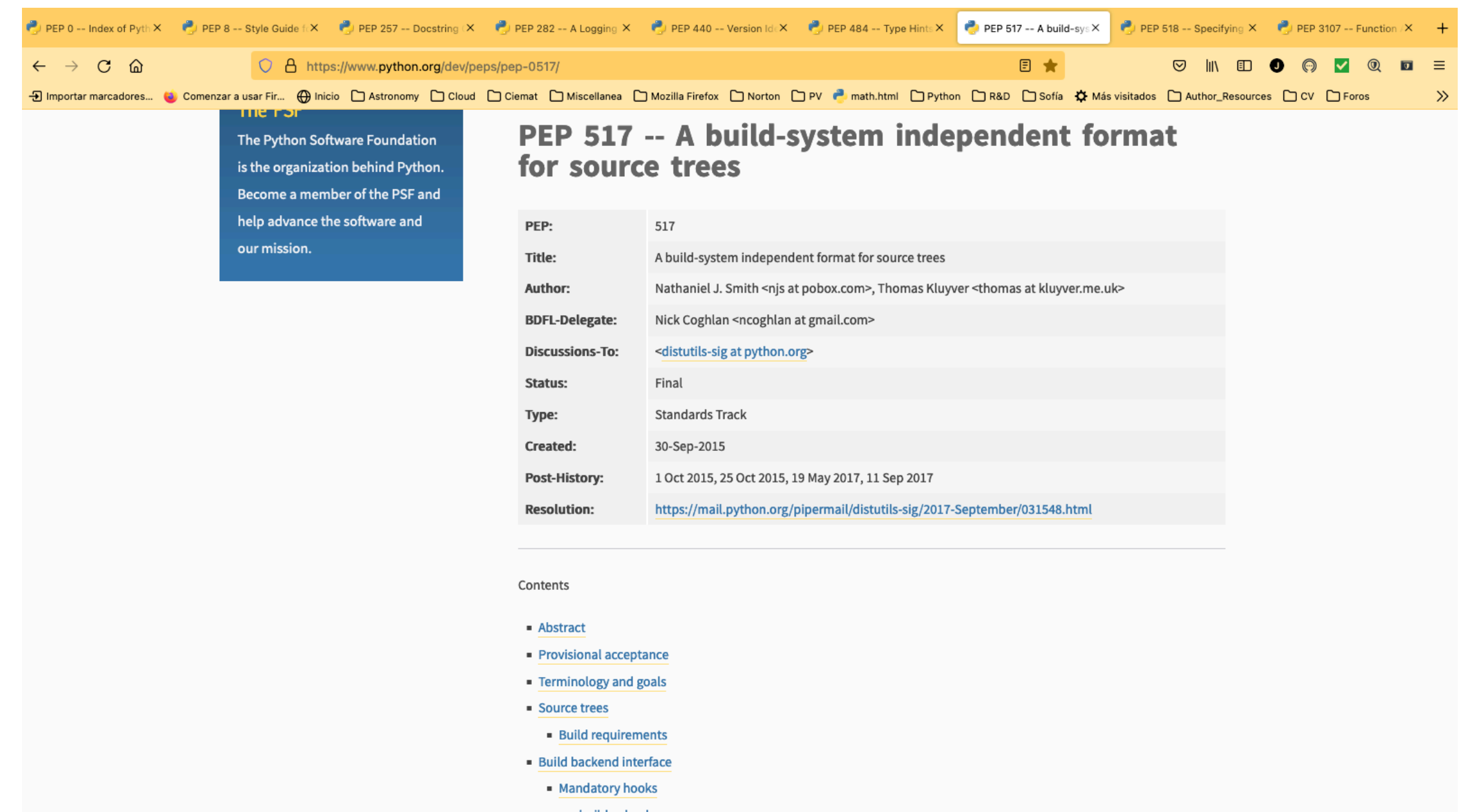
PEP 282 -- A Logging System.

PEP 440 -- Version Identification and Dependency Specification.

PEP 484 -- Type Hints.

PEP 517 -- A build-system independent format for source trees.

PEP 3107 -- Function Annotations.



The screenshot shows a web browser with multiple tabs open, including 'PEP 0 -- Index of Pyth...', 'PEP 8 -- Style Guide', 'PEP 257 -- Docstring', 'PEP 282 -- A Logging', 'PEP 440 -- Version Id...', 'PEP 484 -- Type Hint...', 'PEP 517 -- A build-sys...', 'PEP 518 -- Specifying', and 'PEP 3107 -- Function'. The active tab is 'PEP 517 -- A build-system independent format for source trees'. The page content includes a sidebar with the Python Software Foundation logo and a main area with the title 'PEP 517 -- A build-system independent format for source trees'. Below the title is a table with the following information:

PEP:	517
Title:	A build-system independent format for source trees
Author:	Nathaniel J. Smith <njs at pobox.com>, Thomas Kluyver <thomas at kluyver.me.uk>
BDFL-Delegate:	Nick Coghlan <ncoghlan at gmail.com>
Discussions-To:	<distutils-sig at python.org>
Status:	Final
Type:	Standards Track
Created:	30-Sep-2015
Post-History:	1 Oct 2015, 25 Oct 2015, 19 May 2017, 11 Sep 2017
Resolution:	https://mail.python.org/pipermail/distutils-sig/2017-September/031548.html

Below the table is a 'Contents' section with a list of links: Abstract, Provisional acceptance, Terminology and goals, Source trees, Build requirements, Build backend interface, Mandatory hooks, and build wheel.

Recursos para el Desarrollo de Software Python

Python Enhancement Proposals (PEPs)

Las PEPs y los Procesos relacionados constituyen un sistema reglado por la PSF, trazable y colaborativo, para:

Elevar, discutir, acordar, aceptar, rechazar, modificar o actualizar **normas, información** y **buenas prácticas** en relación al diseño y uso del lenguaje.

Se contemplan tanto **PEPs informativas** (I - Informational PEP) como **PEPs normativas** (S - Standards Track PEP), además de las **PEPs en curso** (P - Process PEP).

Ejemplos:

PEP 8 -- Style Guide for Python Code.

PEP 257 -- Doctring Conventions.

PEP 282 -- A Logging System.

PEP 440 -- Version Identification and Dependency Specification.

PEP 484 -- Type Hints.

PEP 517 -- A build-system independent format for source trees.

PEP 3107 -- Function Annotations.

The screenshot shows a web browser with multiple tabs open, including 'PEP 0 -- Index of Pyth...', 'PEP 8 -- Style Guide', 'PEP 257 -- Doctring', 'PEP 282 -- A Logging', 'PEP 440 -- Version Id...', 'PEP 484 -- Type Hint...', 'PEP 517 -- A build-sy...', and 'PEP 518 -- Specifying'. The active tab is 'PEP 518 -- Specifying'. The browser address bar shows 'https://www.python.org/dev/peps/pep-0518/'. The page content includes a blue sidebar with the Python Software Foundation logo and a main content area titled 'PEP 518 -- Specifying Minimum Build System Requirements for Python Projects'. The main content area contains a table with the following information:

PEP:	518
Title:	Specifying Minimum Build System Requirements for Python Projects
Author:	Brett Cannon <brett at python.org>, Nathaniel Smith <njs at pobox.com>, Donald Stufft <dona at stufft.io>
BDFL-Delegate:	Nick Coghlan
Discussions-To:	distutils-sig <distutils-sig at python.org>
Status:	Final
Type:	Standards Track
Created:	10-May-2016
Post-History:	10-May-2016, 11-May-2016, 13-May-2016
Resolution:	https://mail.python.org/pipermail/distutils-sig/2016-May/028969.html

Below the table, there is a 'Contents' section with a list of links: Abstract, Rationale, Specification, File Format, build-system table, and tool table.

Recursos para el Desarrollo de Software

Python

Python Enhancement Proposals (PEPs)

Las PEPs y los Procesos relacionados constituyen un sistema reglado por la PSF, trazable y colaborativo, para:

Elevar, discutir, acordar, aceptar, rechazar, modificar o actualizar **normas, información** y **buenas prácticas** en relación al diseño y uso del lenguaje.

Se contemplan tanto **PEPs informativas** (I - Informational PEP) como **PEPs normativas** (S - Standards Track PEP), además de las **PEPs en curso** (P - Process PEP).

Ejemplos:

PEP 8 -- Style Guide for Python Code.

PEP 257 -- Docstring Conventions.

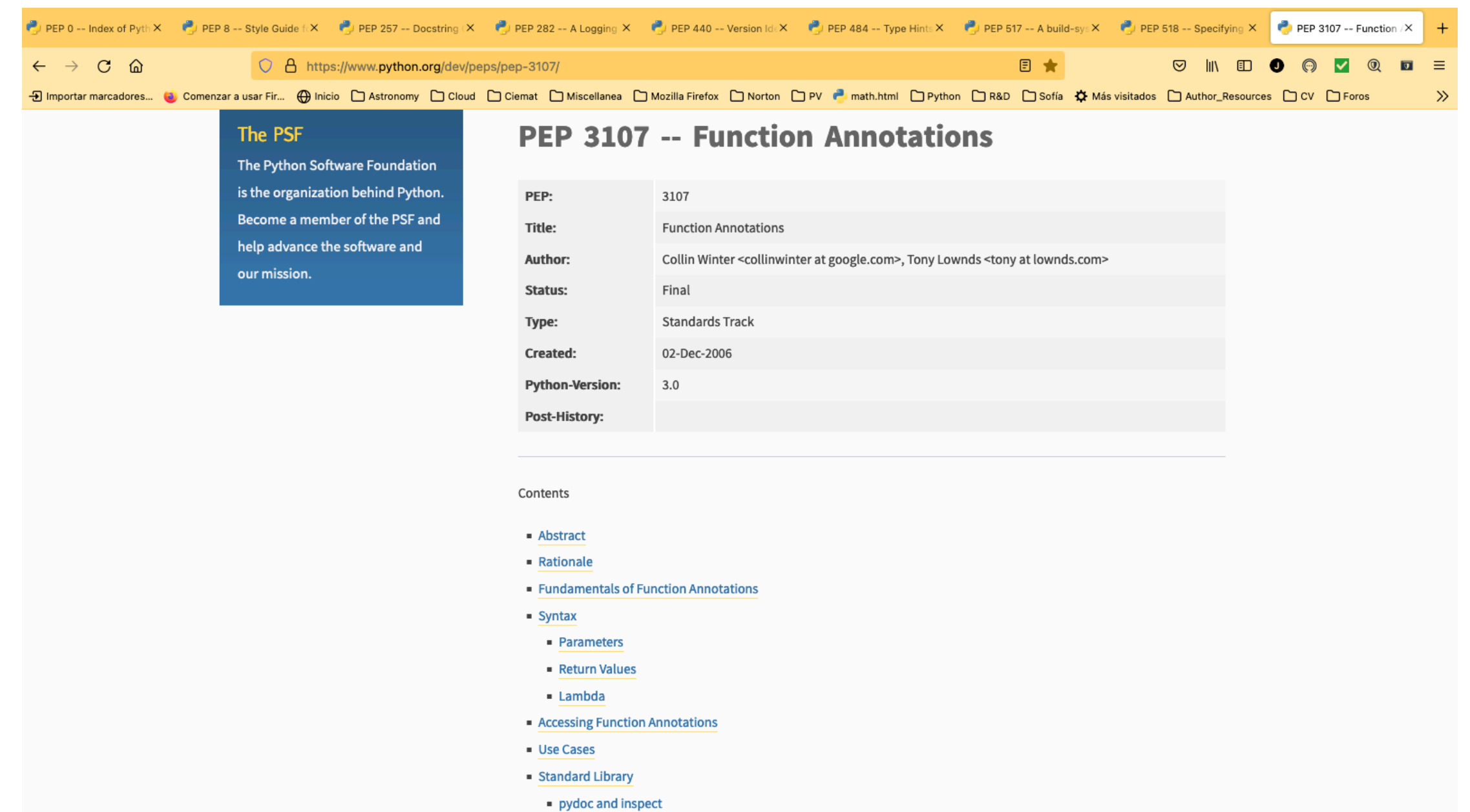
PEP 282 -- A Logging System.

PEP 440 -- Version Identification and Dependency Specification.

PEP 484 -- Type Hints.

PEP 517 -- A build-system independent format for source trees.

PEP 3107 -- Function Annotations.



The screenshot shows a web browser with multiple tabs open, including various PEPs. The active tab is 'PEP 3107 -- Function Annotations'. The page header shows the URL 'https://www.python.org/dev/peps/pep-3107/'. On the left, there is a blue box with the text 'The PSF' and 'The Python Software Foundation is the organization behind Python. Become a member of the PSF and help advance the software and our mission.' The main content area is titled 'PEP 3107 -- Function Annotations' and contains a table with the following information:

PEP:	3107
Title:	Function Annotations
Author:	Collin Winter <collinwinter at google.com>, Tony Lownds <tony at lownds.com>
Status:	Final
Type:	Standards Track
Created:	02-Dec-2006
Python-Version:	3.0
Post-History:	

Below the table, there is a 'Contents' section with a list of links: Abstract, Rationale, Fundamentals of Function Annotations, Syntax (with sub-links for Parameters, Return Values, and Lambda), Accessing Function Annotations, Use Cases, and Standard Library (with a sub-link for pydoc and inspect).

Recursos para el Desarrollo de Software

Entorno Python: Aplicaciones, Librerías y Plataformas actuales y en estudio

Fuentes: [Python Developers Survey 2021](#) and Paul & Harvey Deitel "[Python for Programmers](#)", 2019):

Cloud Platforms: [PythonAnywhere](#), Microsoft Azure, [AWS](#), Google Cloud Platform, OpenStack, RockSpace, Linode, OpenShift, DigitalOcean, Heroku.

Unit Testing Frameworks: pytest, nose, [unittest](#), [doctest](#), tox, Hypothesis, mock.

ORM (DB dev.): Peewee, PonyORM, SQLAlchemy, TortoiseORM, Dejavu, SQLAlchemy, [DjangoORM](#), Raw SQL.

Database: MongoDB, MySQL, Redis, Neo4j, [PostgreSQL](#), Cassandra, Oracle DB, Amazon Redshift, MS SQL Server, h2, CouchBase, DB2, HBase, SQLite.

Data Science Libraries: [Numpy](#), [Pandas](#), [SciPy](#), NLTK, TextBlob, Tweepy, [Matplotlib](#), [Seaborn](#), Folium.

Big Data Tools: Apache Spark, Tez, Kafka, [Hadoop](#), MapReduce, Samza, Beam, Flink, Kive, ClickHouse, Dask.

Artificial Intelligence: Natural Language Processing (NLP), Data Mining Twitter, IBM Watson, [scikit-learn](#) (Machine Learning), [Keras](#) (Deep Learning), Internet of Things (IoT).

Continuous Integration Systems: Jenkins, Hudson, BitBucket, Pipelines, CruiseControl, TeamCity, [GitHub Actions](#), Circle CI, Travis CI, Gitlab CI, AppVeyor, Bamboo.

Configuration Management: Chef, Custom Solution, Ansible, Salt, [Puppet](#).

Documentation: [Sphinx](#), MKDocs, Doxygen.

Editors: [Atom](#), Eclipse + PyDev, Emacs, IDLE, [JupyterLab](#), [Jupyter Notebook](#), NotePad++, PyCharm Community Edition, PyCharm Professional Edition, Python tools for Visual Studio (PTVS), [Spyder](#), Sublime Text, Vim, VS Code.

Tools to isolate environments: virtualenv, [venv](#), hatch, [Poetry](#), pipenv, [Conda](#), [Docker \(container\)](#).

Packaging Tools: [pip](#), [Conda](#), pipenv, Poetry, [venv](#), virtualenv, [flit](#), tox, PDM, [twine](#), Docker (container), virtual machines, word place specific proprietary solutions.

Automated Version Managing Systems: Dependabot, PyUp, custom tools (cron job, CI task ...).

Tools for installing packages: [pip](#), easy_install, [Conda](#), [Poetry](#), pip-sync, pipx.

Package sources: [PyPI](#), [local source](#), internal mirror of PyPI, private PyPI, [Anaconda](#), Default Conda channel, conda-forge Conda channel, Art-factory, [GitLab](#), Linux distribution.

Inteligencia Artificial

(II - 2/5) Instalación y Entornos Virtuales

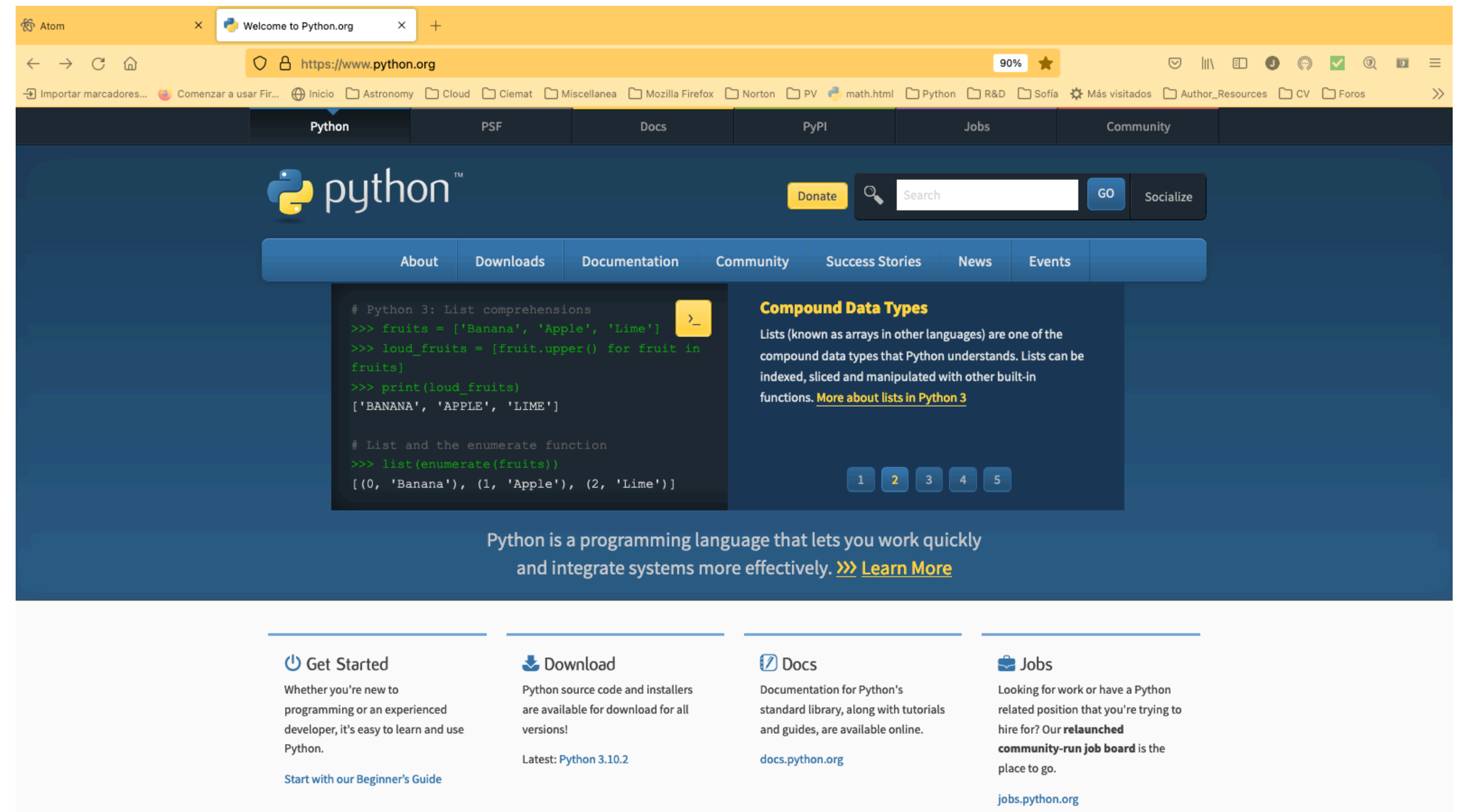
Recursos para el Desarrollo de Software

Entornos de Trabajo

Instalación y Modos de Trabajo

Descarga del lenguaje desde la web institucional de [Python](https://www.python.org)

- ✓ El método más ligero.
- ✓ Permite la creación de **Virtual Environments** (esencial para desarrolladores).
- ↕ Precisa la instalación de un **Editor** de código (por ejemplo, Atom).
- ↕ Ejecución desde **Terminal**, **Command Prompt** u otra *Shell* de terceros.
- ↕ Precisa la **instalación** manual de las dependencias en cada EV.



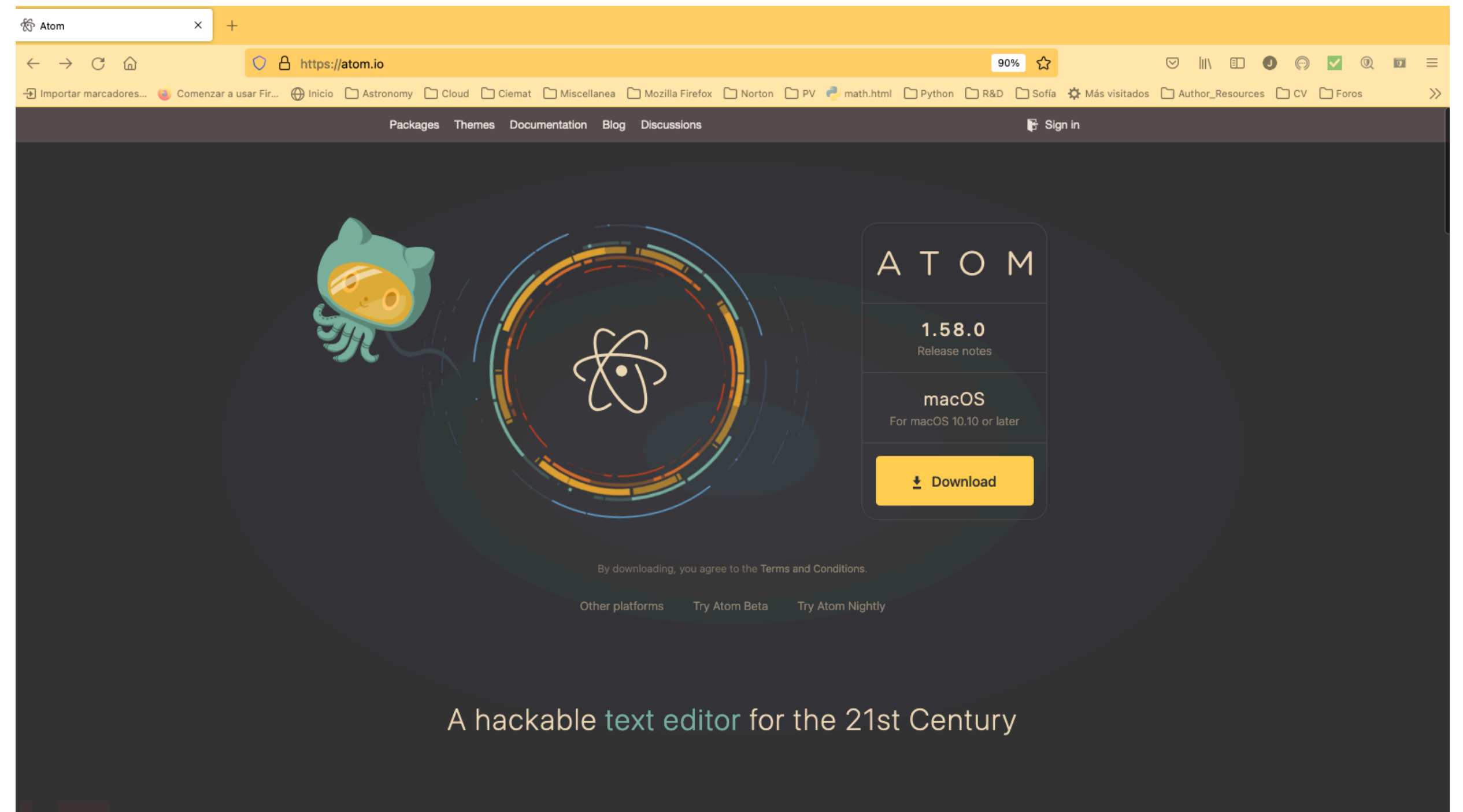
Recursos para el Desarrollo de Software

Entornos de Trabajo

Instalación y Modos de Trabajo

Descarga del lenguaje desde la web institucional de [Python](#)

- ✓ El método más ligero.
- ✓ Permite la creación de **Virtual Environments** (esencial para desarrolladores).
- ↕ Precisa la instalación de un **Editor** de código (por ejemplo, Atom).
- ↕ Ejecución desde **Terminal**, **Command Prompt** u otra *Shell* de terceros.
- ↕ Precisa la **instalación** manual de las dependencias en cada EV.



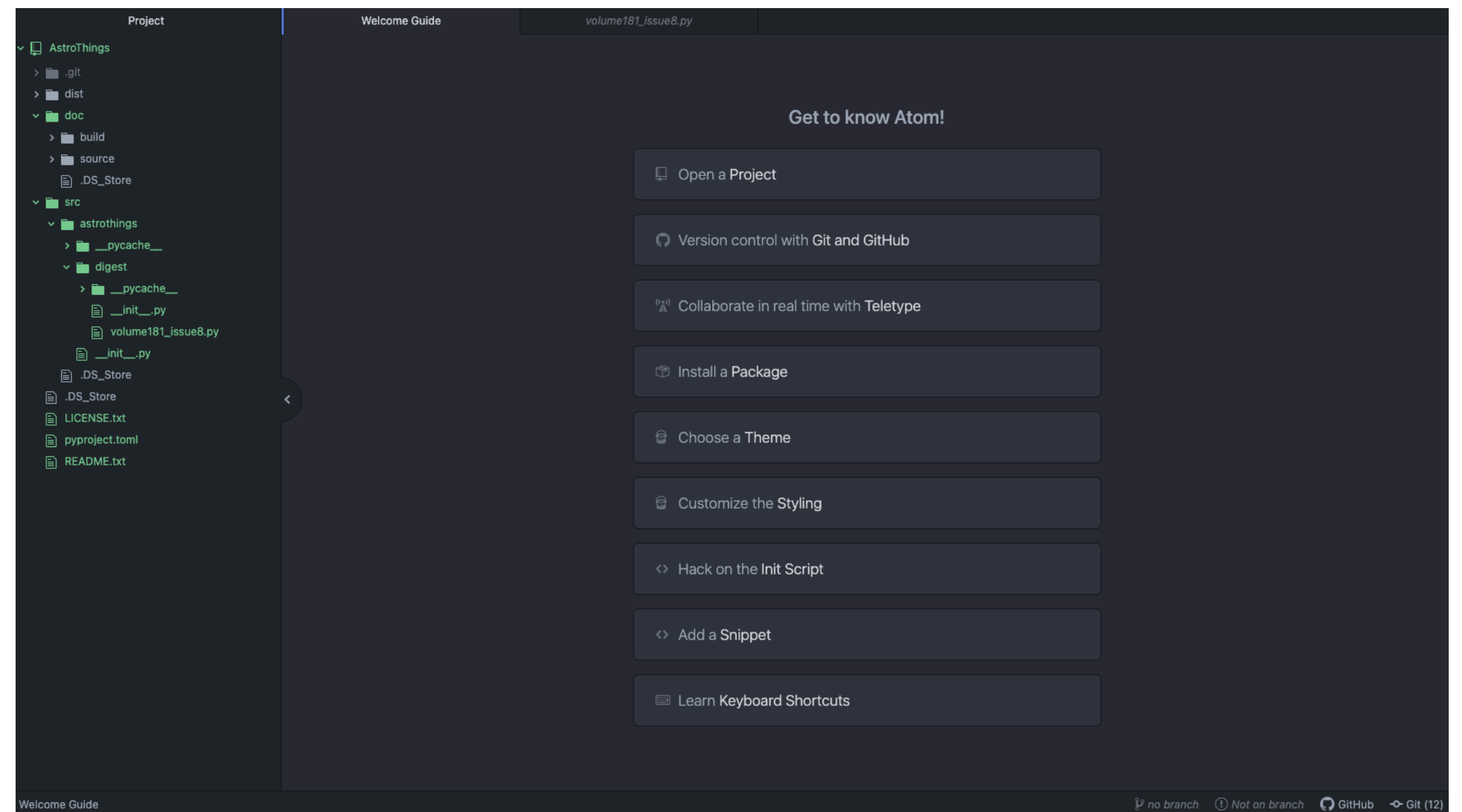
Recursos para el Desarrollo de Software

Entornos de Trabajo

Instalación y Modos de Trabajo

Descarga del lenguaje desde la web institucional de [Python](#)

- ✓ El método más ligero.
- ✓ Permite la creación de **Virtual Environments** (esencial para desarrolladores).
- ↕ Precisa la instalación de un **Editor** de código (por ejemplo, Atom).
- ↕ Ejecución desde **Terminal, Command Prompt** u otra *Shell* de terceros.
- ↕ Precisa la **instalación** manual de las dependencias en cada EV.



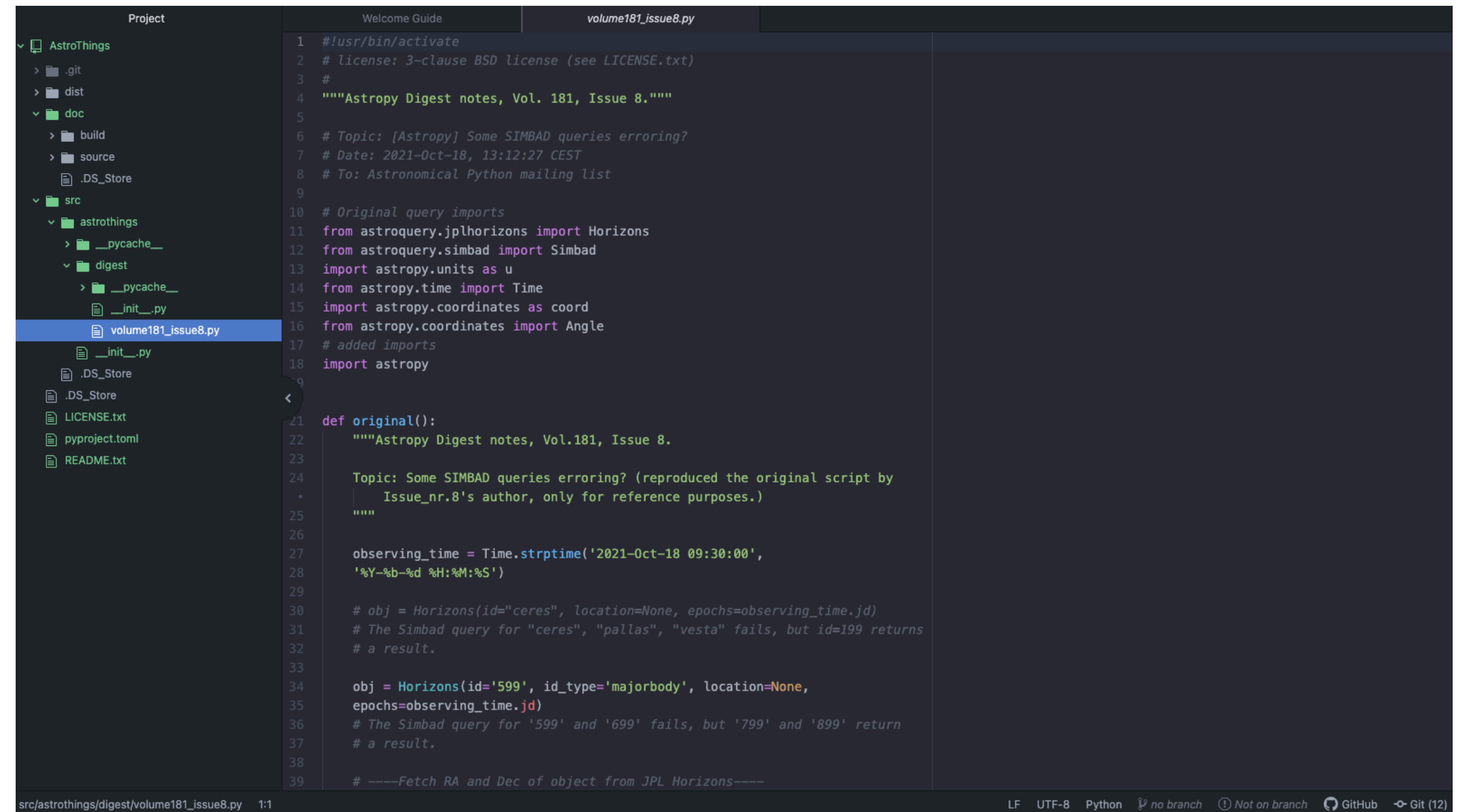
Recursos para el Desarrollo de Software

Entornos de Trabajo

Instalación y Modos de Trabajo

Descarga del lenguaje desde la web institucional de [Python](#)

- ✓ El método más ligero.
- ✓ Permite la creación de **Virtual Environments** (esencial para desarrolladores).
- ↕ Precisa la instalación de un **Editor** de código (por ejemplo, Atom).
- ↕ Ejecución desde **Terminal**, **Command Prompt** u otra *Shell* de terceros.
- ↕ Precisa la **instalación** manual de las dependencias en cada EV.



The screenshot shows a code editor with a dark theme. On the left, a file explorer shows a project structure with folders like '.git', 'doc', 'src', and files like 'LICENSE.txt', 'pyproject.toml', and 'README.txt'. The main editor area displays a Python script named 'volume181_issue8.py'. The script includes a shebang line, a license notice, a docstring, and a function 'original()' that uses 'astroquery' and 'astropy' to fetch astronomical data. The status bar at the bottom indicates the file path 'src/astrothings/digest/volume181_issue8.py' and various settings like 'LF', 'UTF-8', 'Python', and 'no branch'.

```
1 #!/usr/bin/activate
2 # license: 3-clause BSD license (see LICENSE.txt)
3 #
4 """Astropy Digest notes, Vol. 181, Issue 8."""
5
6 # Topic: [Astropy] Some SIMBAD queries erroring?
7 # Date: 2021-Oct-18, 13:12:27 CEST
8 # To: Astronomical Python mailing list
9
10 # Original query imports
11 from astroquery.jplhorizons import Horizons
12 from astroquery.simbad import Simbad
13 import astropy.units as u
14 from astropy.time import Time
15 import astropy.coordinates as coord
16 from astropy.coordinates import Angle
17 # added imports
18 import astropy
19
20
21 def original():
22     """Astropy Digest notes, Vol.181, Issue 8.
23
24     Topic: Some SIMBAD queries erroring? (reproduced the original script by
25     Issue_nr.8's author, only for reference purposes.)
26     """
27
28     observing_time = Time.strptime('2021-Oct-18 09:30:00',
29                                   '%Y-%b-%d %H:%M:%S')
30
31     # obj = Horizons(id="ceres", location=None, epochs=observing_time.jd)
32     # The Simbad query for "ceres", "pallas", "vesta" fails, but id=199 returns
33     # a result.
34
35     obj = Horizons(id='599', id_type='majorbody', location=None,
36                   epochs=observing_time.jd)
37     # The Simbad query for '599' and '699' fails, but '799' and '899' return
38     # a result.
39
40     # ----Fetch RA and Dec of object from JPL Horizons----
```

Recursos para el Desarrollo de Software

Entornos de Trabajo

Instalación y Modos de Trabajo

Descarga del lenguaje desde la web institucional de [Python](#)

- ✓ El método más ligero.
- ✓ Permite la creación de **Virtual Environments** (esencial para desarrolladores).
- ↕ Precisa la instalación de un **Editor** de código (por ejemplo, Atom).
- ↕ Ejecución desde **Terminal**, **Command Prompt** u otra *Shell* de terceros.
- ↕ Precisa la **instalación** manual de las dependencias en cada EV.

```
INSTALL  UNINSTALL  EXPORT  BE AWARE OF VENV  VIEW DEPENDENCIES  ZEN  TERMINAL  ~ -- Python  +
josepedro@iMac-de-Jose ~ % source pvlabvenv/bin/activate
(pvlabvenv) josepedro@iMac-de-Jose ~ % python
Python 3.9.7 (v3.9.7:1016ef3790, Aug 30 2021, 16:39:15)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> index = [2*x+1 for x in range(0, 3)]
>>> index
[1, 3, 5]
>>> import numpy as np
>>> data = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
>>> columns = ['primera', 'segunda', 'tercera']
>>> import pandas as pd
>>> df = pd.DataFrame(index = index, data = data, columns = columns)
>>> df
   primera  segunda  tercera
1         1         2         3
3         4         5         6
5         7         8         9
>>>
```

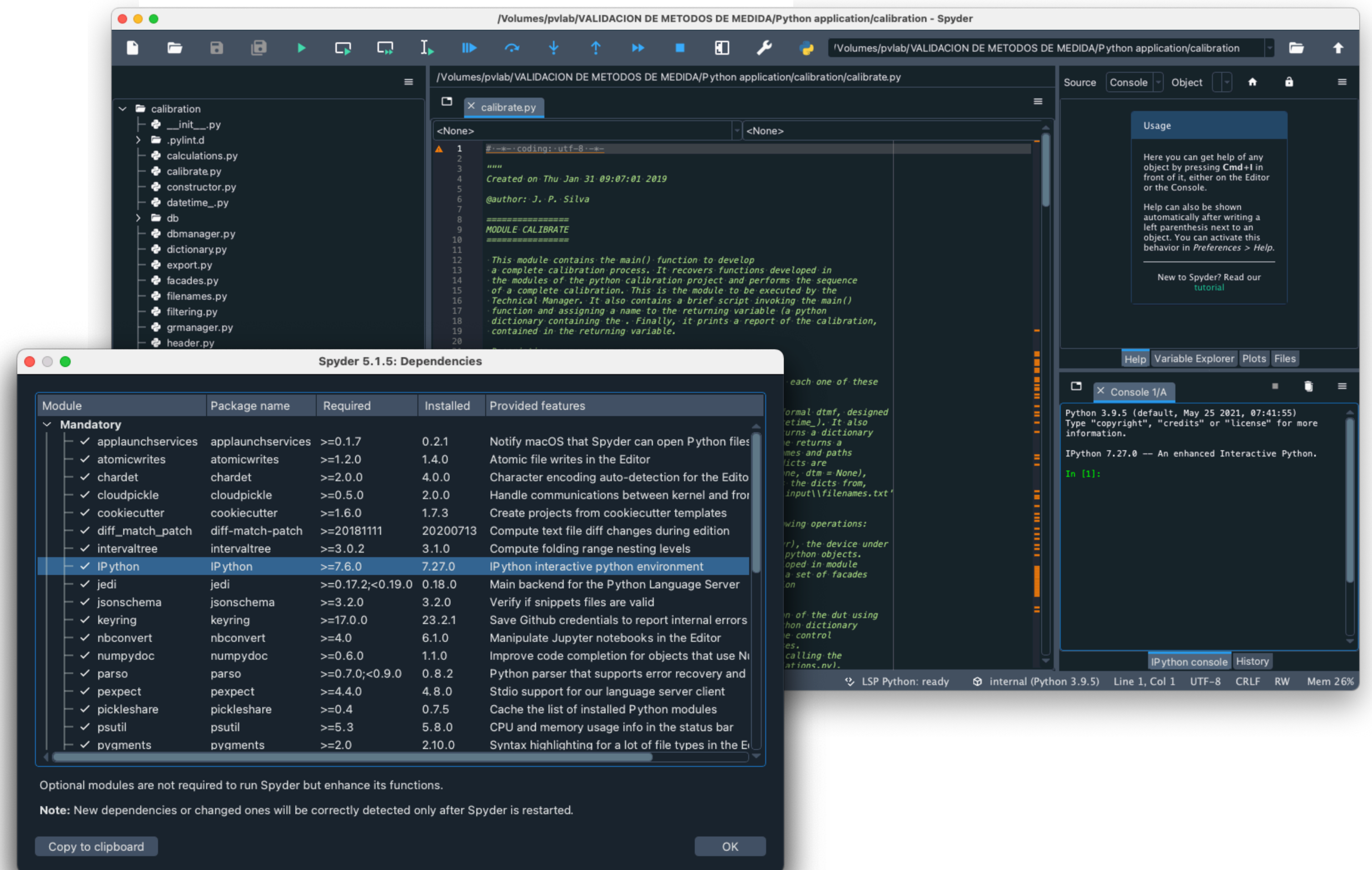

Recursos para el Desarrollo de Software

Entornos de Trabajo

Instalación y Modos de Trabajo

Descarga de un **Entorno Integrado de Desarrollo (IDE)**

- ✓ Provee un **editor**, un **terminal** con funciones avanzadas (*IPython*), **visor de proyectos**, **visor gráfico**, **ayuda**, **explorador de variables**, **panel de gráficos**, etc. en un único entorno.
- ✓ Contiene un listado de **Dependencias** (librerías pre-instaladas) amplio, orientado al cálculo científico - técnico.
- ✗ **No admite la instalación** de nuevos módulos (precisa Anaconda).
- ✗ **No admite el aislamiento** de entornos de trabajo (precisa Anaconda).
- ✗ **Requiere** que el proyecto haya sido creado mediante el mismo IDE.
- ✗ En general **no contiene la última versión de Python** o de otros paquetes.



Recursos para el Desarrollo de Software

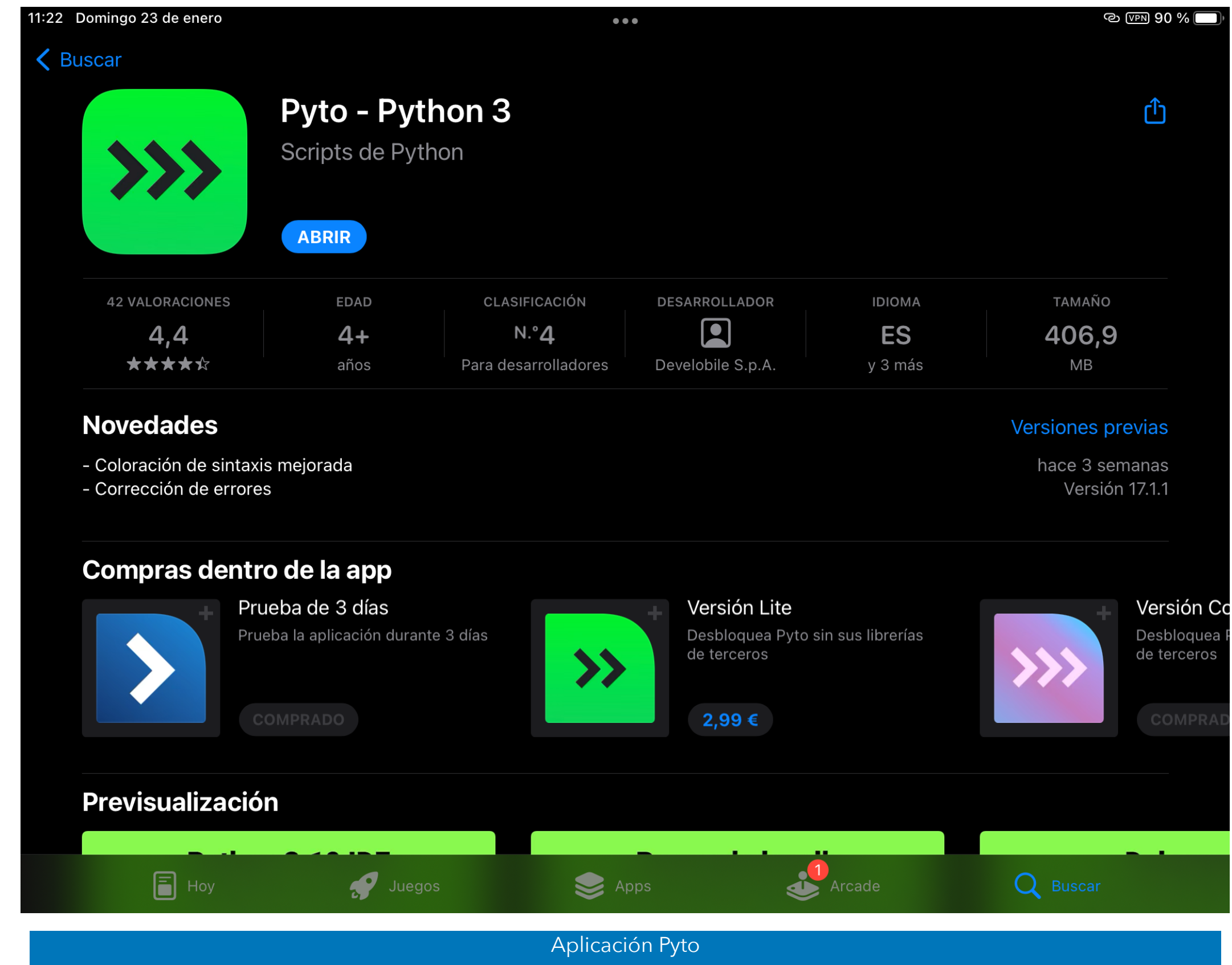
Entornos de Trabajo

Instalación y Modos de Trabajo

Entorno Integrado de Desarrollo (IDE) para móviles y tablets (e.g. *Pyto*).

Capacidades (aplicación *Pyto*):

- ✓ Terminal (shell) y Consola de Python.
- ✓ Editor de código.
- ✓ Descarga de paquetes de *Python Package Index*, con opción a seleccionar la versión a descargar.
- ✓ Visor de Proyectos.
- ✗ No admite la creación (aislamiento) de entornos virtuales.



Recursos para el Desarrollo de Software

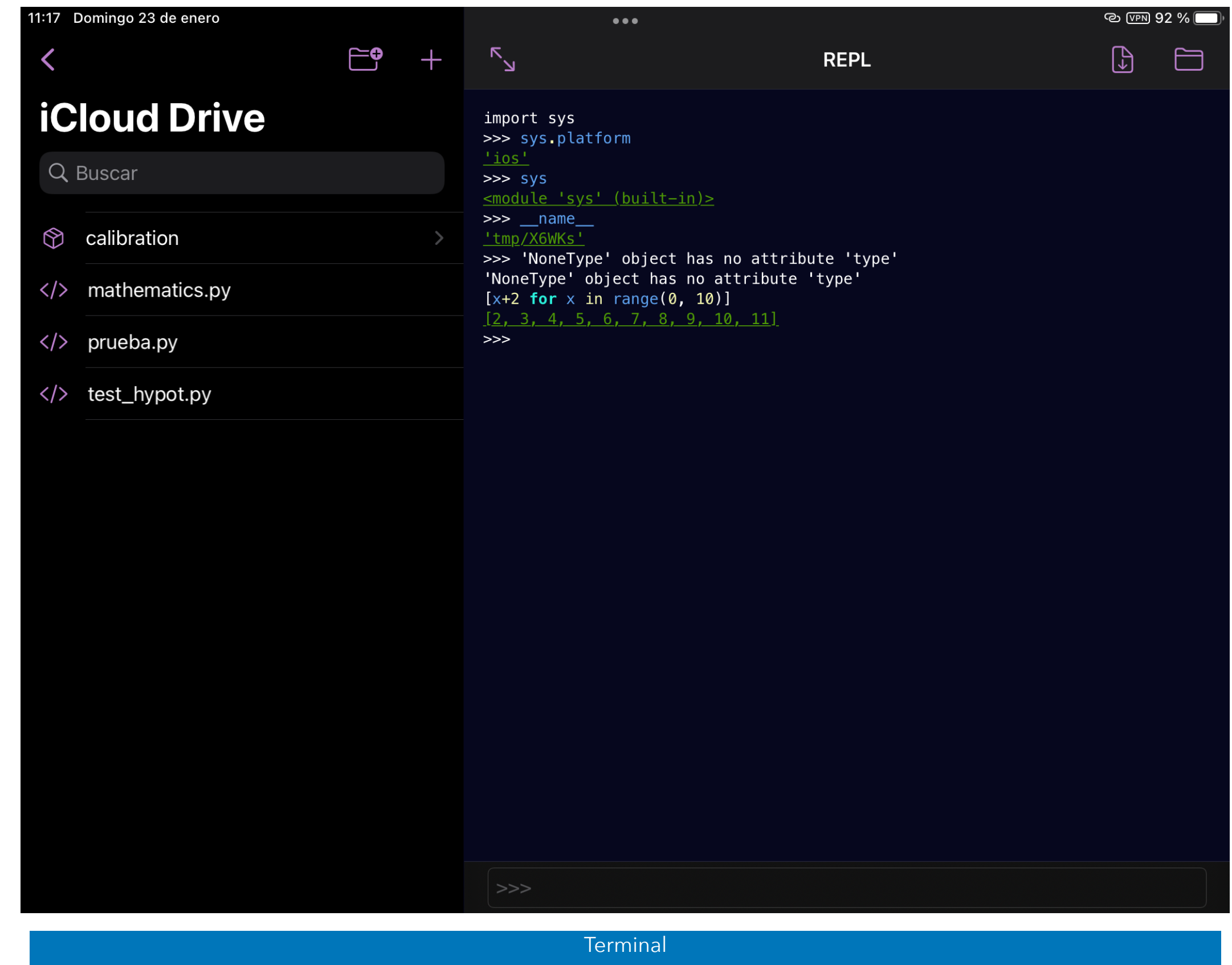
Entornos de Trabajo

Instalación y Modos de Trabajo

Entorno Integrado de Desarrollo (IDE) para móviles y tablets (e.g. *Pyto*).

Capacidades (aplicación *Pyto*):

- ✓ Terminal (shell) y Consola de Python.
- ✓ Editor de código.
- ✓ Descarga de paquetes de *Python Package Index*, con opción a seleccionar la versión a descargar.
- ✓ Visor de Proyectos.
- ✗ No admite la creación (aislamiento) de entornos virtuales.



Recursos para el Desarrollo de Software

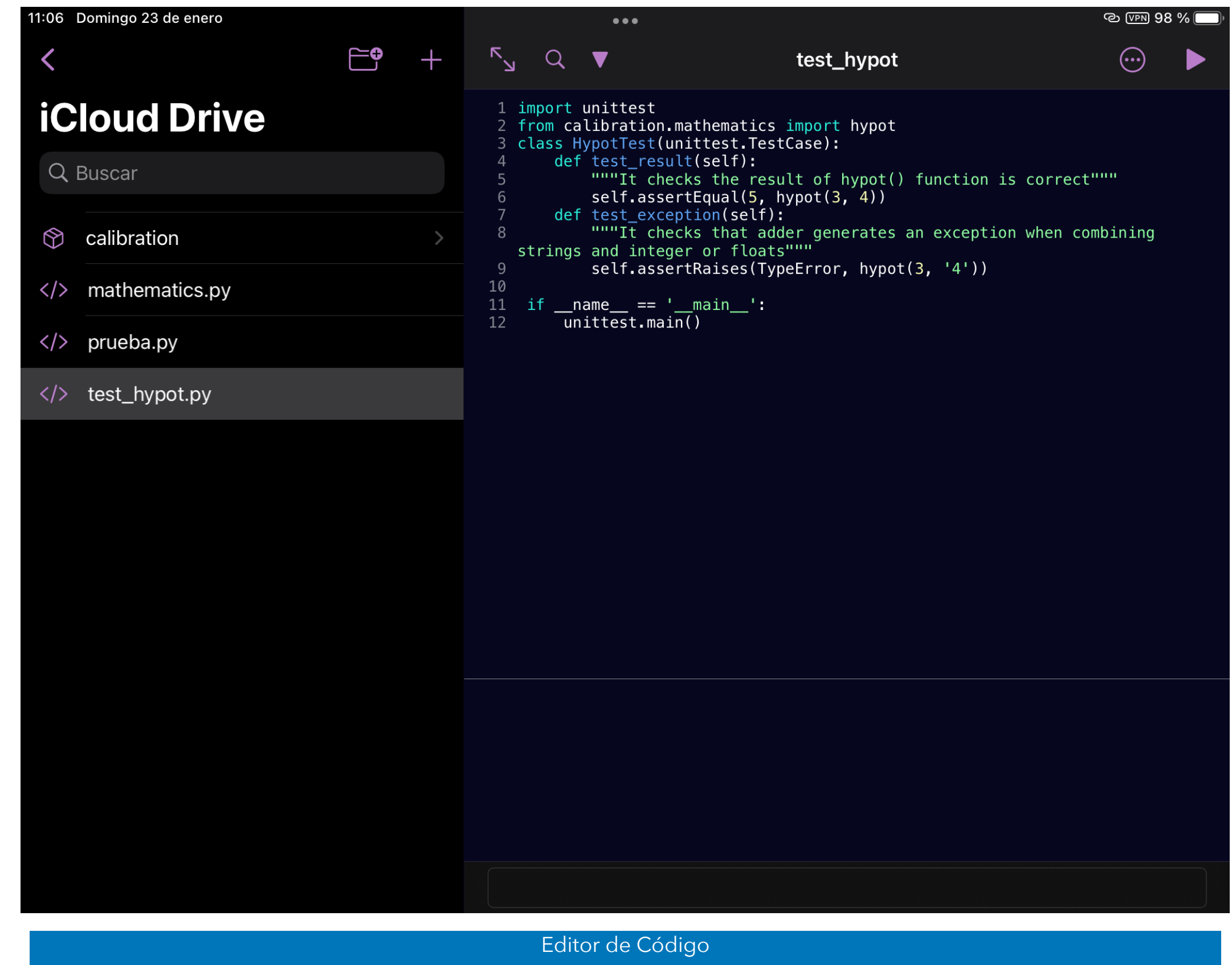
Entornos de Trabajo

Instalación y Modos de Trabajo

Entorno Integrado de Desarrollo (IDE) para móviles y tablets (e.g. *Pyto*).

Capacidades (aplicación *Pyto*):

- ✓ Terminal (shell) y Consola de Python.
- ✓ Editor de código.
- ✓ Descarga de paquetes de *Python Package Index*, con opción a seleccionar la versión a descargar.
- ✓ Visor de Proyectos.
- ✗ No admite la creación (aislamiento) de entornos virtuales.



Recursos para el Desarrollo de Software

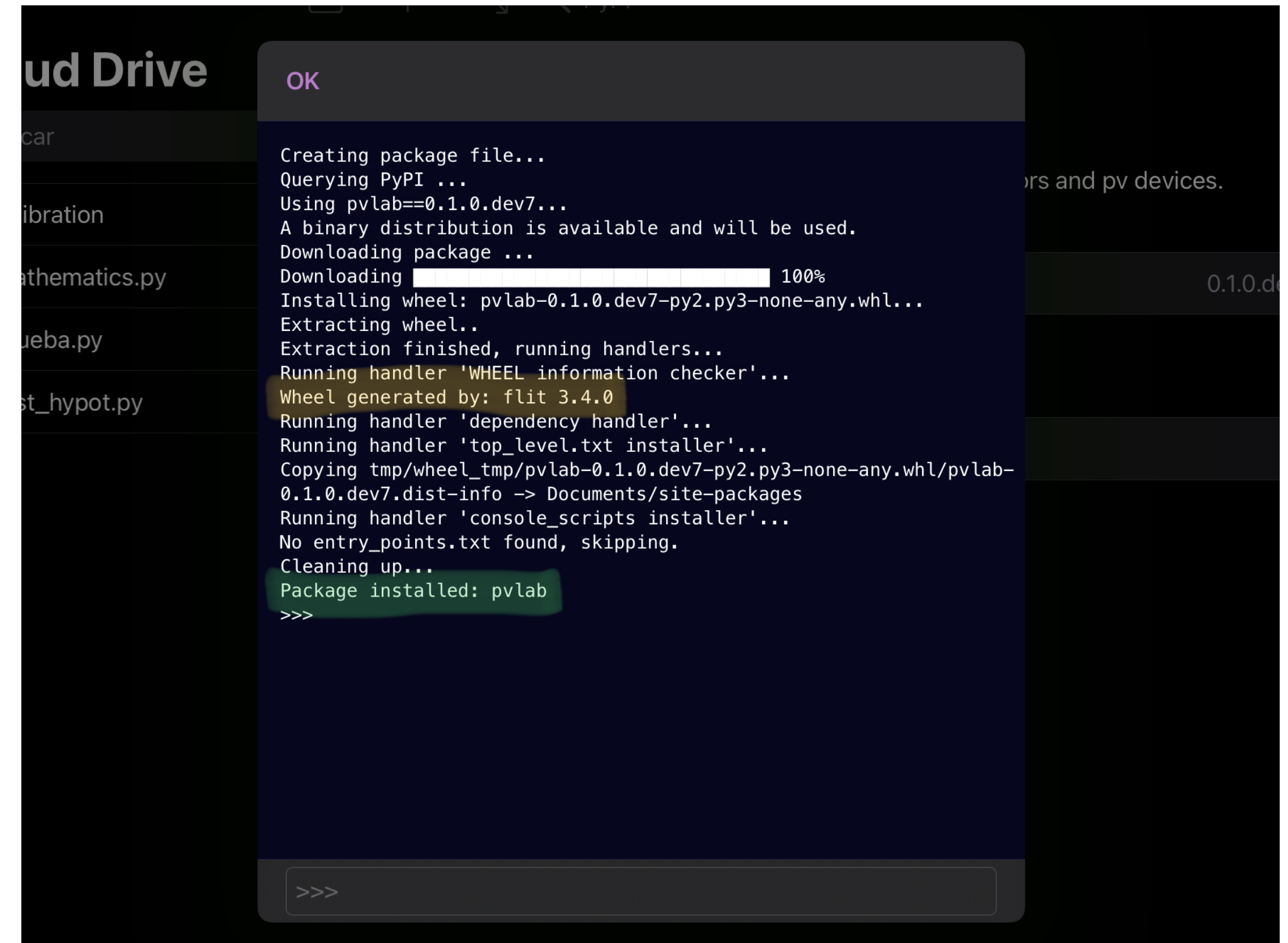
Entornos de Trabajo

Instalación y Modos de Trabajo

Entorno Integrado de Desarrollo (IDE) para móviles y tablets (e.g. *Pyto*).

Capacidades (aplicación *Pyto*):

- ✓ Terminal (shell) y Consola de Python.
- ✓ Editor de código.
- ✓ Descarga de paquetes de *Python Package Index*, con opción a seleccionar la versión a descargar.
- ✓ Visor de Proyectos.
- ✗ No admite la creación (aislamiento) de entornos virtuales.



```
ud Drive
OK
Creating package file...
Querying PyPI ...
Using pvlab==0.1.0.dev7...
A binary distribution is available and will be used.
Downloading package ...
Downloading 100%
Installing wheel: pvlab-0.1.0.dev7-py2.py3-none-any.whl...
Extracting wheel..
Extraction finished, running handlers...
Running handler 'WHEEL information checker'...
Wheel generated by: flit 3.4.0
Running handler 'dependency handler'...
Running handler 'top_level.txt installer'...
Copying tmp/wheel_tmp/pvlab-0.1.0.dev7-py2.py3-none-any.whl/pvlab-
0.1.0.dev7.dist-info -> Documents/site-packages
Running handler 'console_scripts installer'...
No entry_points.txt found, skipping.
Cleaning up...
Package installed: pvlab
>>>
```

Descarga de paquetes Python desde PyPI

Recursos para el Desarrollo de Software

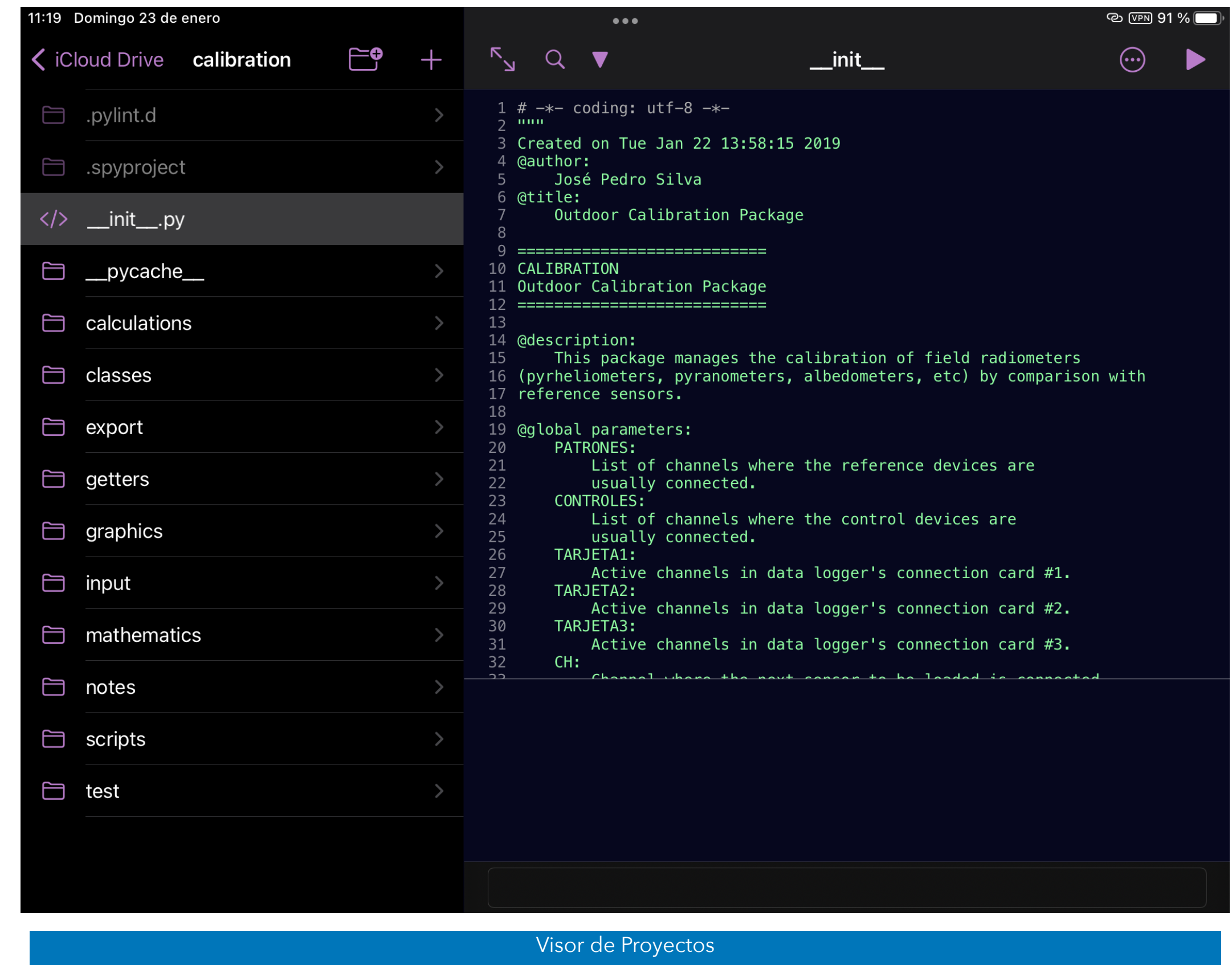
Entornos de Trabajo

Instalación y Modos de Trabajo

Entorno Integrado de Desarrollo (IDE) para móviles y tablets (e.g. *Pyto*).

Capacidades (aplicación *Pyto*):

- ✓ Terminal (shell) y Consola de Python.
- ✓ Editor de código.
- ✓ Descarga de paquetes de *Python Package Index*, con opción a seleccionar la versión a descargar.
- ✓ Visor de Proyectos.
- ✗ No admite la creación (aislamiento) de entornos virtuales.



Recursos para el Desarrollo de Software

Entornos de Trabajo

Instalación y Modos de Trabajo

Entorno Integrado de Desarrollo (IDE) para móviles y tablets (e.g. *Pyto*).

Capacidades (aplicación *Pyto*):

- ✓ Terminal (shell) y Consola de Python.
- ✓ Editor de código.
- ✓ Descarga de paquetes de *Python Package Index*, con opción a seleccionar la versión a descargar.
- ✓ Visor de Proyectos.
- ✗ No admite la creación (aislamiento) de entornos virtuales.

```
<module 'venv' from 'Pyto.app/site-packages/python3.10/venv/__init__.py'>
```

```
>>> quit()
```

```
Documents $ pythom -m venv pvlabvenv
```

```
ImportError: No module named pythom
```

```
Documents $ venv pvlabvenv
```

```
Error: Subprocesses are not supported on this platform.
```

```
Documents $
```



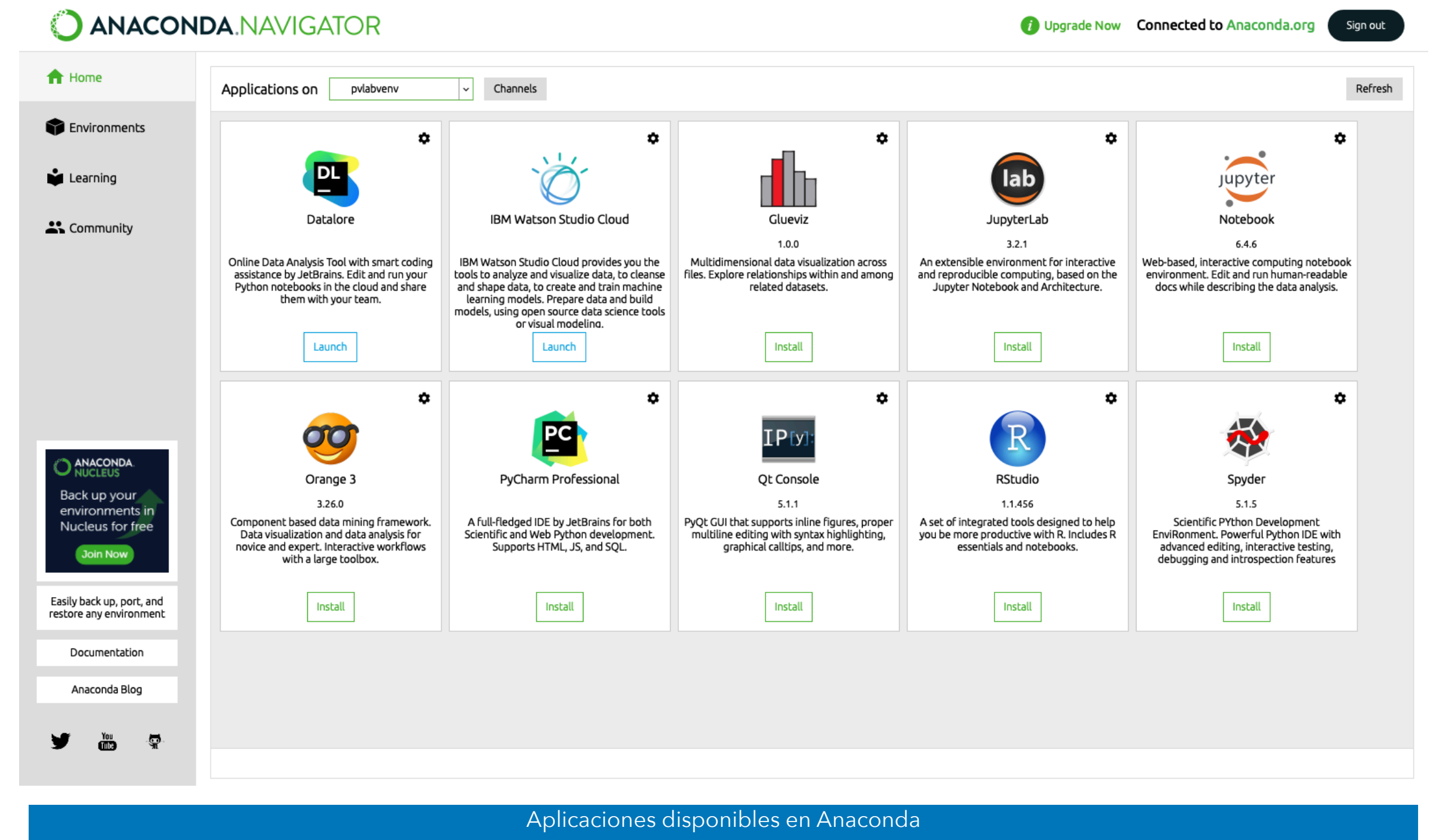
Recursos para el Desarrollo de Software

Entornos de Trabajo

Instalación y Modos de Trabajo

Descarga de un **Kit de Desarrollo de Software** (SDK) completo

- ✓ Permite generar entornos de trabajo aislados (Environments).
- ✓ Además de *Spyder*, contiene un conjunto de **Aplicaciones**, para el análisis y visualización de datos.
- ↕ Administra su propio listado de **Dependencias**, ampliando el listado nativo de *Spyder* (*PyPI mirror*). La instalación se lleva a cabo desde una consola propia.
- ✗ Se trata de un entorno muy pesado, a veces poco operativo.



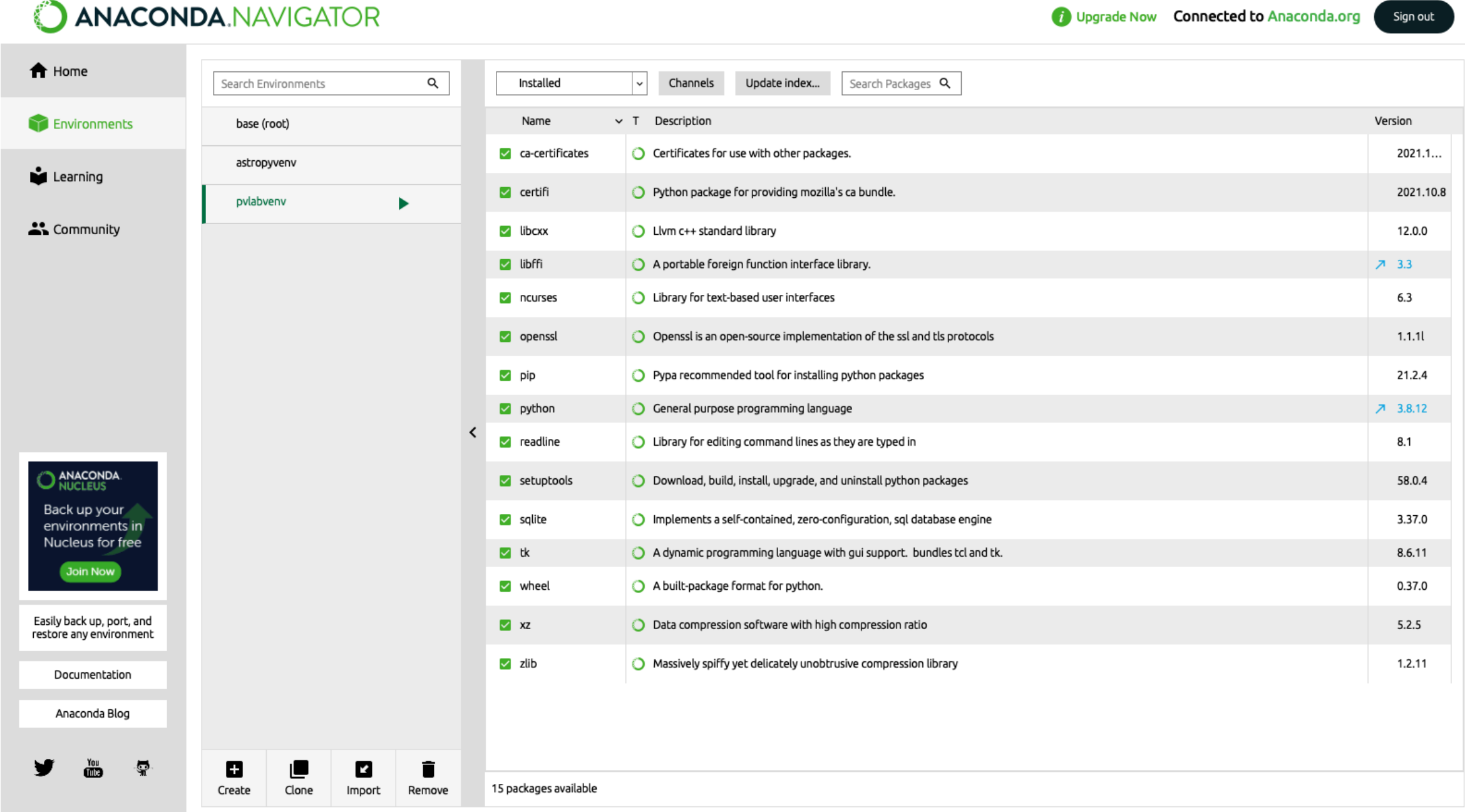
Recursos para el Desarrollo de Software

Entornos de Trabajo

Instalación y Modos de Trabajo

Descarga de un **Kit de Desarrollo de Software (SDK)** completo

- ✓ Permite generar entornos de trabajo aislados (Environments).
- ✓ Además de *Spyder*, contiene un conjunto de **Aplicaciones**, para el análisis y visualización de datos.
- ↕ Administra su propio listado de **Dependencias**, ampliando el listado nativo de *Spyder* (*PyPI mirror*). La instalación se lleva a cabo desde una consola propia.
- ✗ Se trata de un entorno muy pesado, a veces poco operativo.



The screenshot shows the Anaconda Navigator application. On the left sidebar, the 'Environments' tab is selected, showing a list of environments: 'base (root)', 'astropyenv', and 'pvlabenv'. The 'pvlabenv' environment is selected. The main panel displays a list of installed packages for this environment. The table below represents the data shown in the screenshot.

Name	Description	Version
ca-certificates	Certificates for use with other packages.	2021.1...
certifi	Python package for providing mozilla's ca bundle.	2021.10.8
libcxx	Llvm c++ standard library	12.0.0
libffi	A portable foreign function interface library.	3.3
ncurses	Library for text-based user interfaces	6.3
openssl	Openssl is an open-source implementation of the ssl and tls protocols	1.1.1i
pip	Pypa recommended tool for installing python packages	21.2.4
python	General purpose programming language	3.8.12
readline	Library for editing command lines as they are typed in	8.1
setuptools	Download, build, install, upgrade, and uninstall python packages	58.0.4
sqlite	Implements a self-contained, zero-configuration, sql database engine	3.37.0
tk	A dynamic programming language with gui support. bundles tcl and tk.	8.6.11
wheel	A built-package format for python.	0.37.0
xz	Data compression software with high compression ratio	5.2.5
zlib	Massively spiffy yet delicately unobtrusive compression library	1.2.11

15 packages available

Listado de dependencias (paquetes de software) en el EV seleccionado.

Recursos para el Desarrollo de Software

Entornos de Trabajo

Entorno Virtual (Virtual Environment)

Genera y aísla un entorno de trabajo virtual

Puede crearse con la herramienta integrada **venv**.

```
[user] ~ %: python -m venv venvname
```

Se genera **un directorio** que alberga los scripts y paquetes instalados en dicho entorno de trabajo.

Aísla la versión de **python**, así como la de otras **aplicaciones y librerías**:

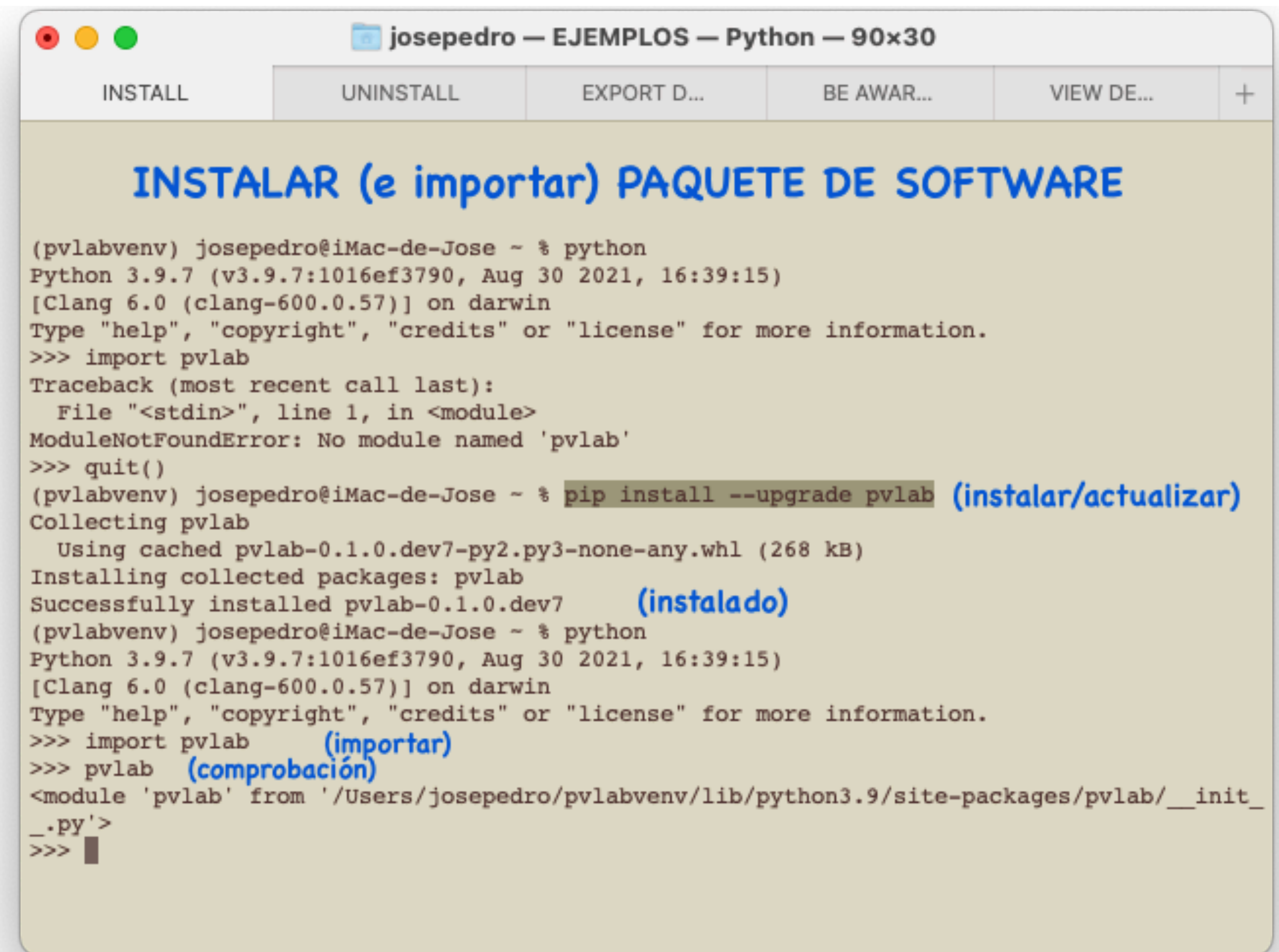
- Asegura que el entorno futuro de ejecución del proyecto es el mismo que el entorno en el que se desarrolló.
- Previene frente a actualizaciones posteriores no compatibles o que generen errores de ejecución o cálculo (p.ej. "floor division" v2.Vs.v3).

Es recomendable crear un entorno para cada proyecto.

El listado de paquetes puede exportarse, guardarse y cargarse:

```
(venvname) ~%: python -m pip freeze > requirements.txt
```

```
(venvname) ~%: python -m pip install -r requirements.txt
```



```
josepedro — EJEMPLOS — Python — 90x30
INSTALL  UNINSTALL  EXPORT D...  BE AWAR...  VIEW DE...  +

INSTALAR (e importar) PAQUETE DE SOFTWARE

(pvlabvenv) josepedro@iMac-de-Jose ~ % python
Python 3.9.7 (v3.9.7:1016ef3790, Aug 30 2021, 16:39:15)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import pvlab
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'pvlab'
>>> quit()
(pvlabvenv) josepedro@iMac-de-Jose ~ % pip install --upgrade pvlab (instalar/actualizar)
Collecting pvlab
  Using cached pvlab-0.1.0.dev7-py2.py3-none-any.whl (268 kB)
Installing collected packages: pvlab
Successfully installed pvlab-0.1.0.dev7 (instalado)
(pvlabvenv) josepedro@iMac-de-Jose ~ % python
Python 3.9.7 (v3.9.7:1016ef3790, Aug 30 2021, 16:39:15)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import pvlab (importar)
>>> pvlab (comprobación)
<module 'pvlab' from '/Users/josepedro/pvlabvenv/lib/python3.9/site-packages/pvlab/__init_
_.py'>
>>>
```

Recursos para el Desarrollo de Software

Entornos de Trabajo

Entorno Virtual (Virtual Environment)

Genera y aísla un entorno de trabajo virtual

Puede crearse con la herramienta integrada **venv**.

```
[user] ~ %: python -m venv venvname
```

Se genera **un directorio** que alberga los scripts y paquetes instalados en dicho entorno de trabajo.

Aísla la versión de **python**, así como la de otras **aplicaciones y librerías**:

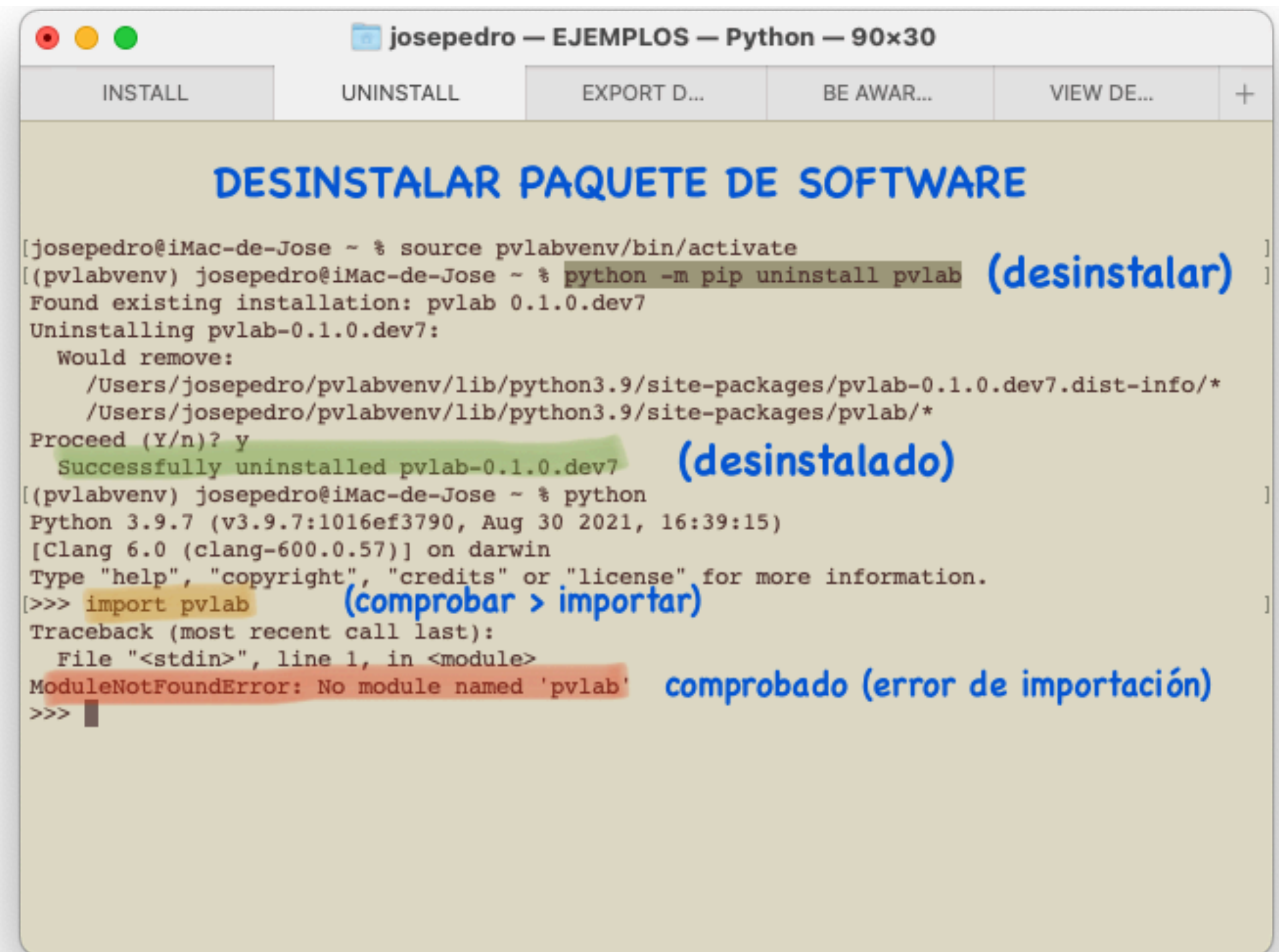
- Asegura que el entorno futuro de ejecución del proyecto es el mismo que el entorno en el que se desarrolló.
- Previene frente a actualizaciones posteriores no compatibles o que generen errores de ejecución o cálculo (p.ej. "floor division" v2.Vs.v3).

Es recomendable crear un entorno para cada proyecto.

El listado de paquetes puede exportarse, guardarse y cargarse:

```
(venvname) ~%: python -m pip freeze > requirements.txt
```

```
(venvname) ~%: python -m pip install -r requirements.txt
```



```
josepedro — EJEMPLOS — Python — 90x30
INSTALL  UNINSTALL  EXPORT D...  BE AWAR...  VIEW DE...  +

DESINSTALAR PAQUETE DE SOFTWARE

[josepedro@iMac-de-Jose ~ % source pvlabvenv/bin/activate
(pvlabvenv) josepedro@iMac-de-Jose ~ % python -m pip uninstall pvlab (desinstalar)
Found existing installation: pvlab 0.1.0.dev7
Uninstalling pvlab-0.1.0.dev7:
  Would remove:
    /Users/josepedro/pvlabvenv/lib/python3.9/site-packages/pvlab-0.1.0.dev7.dist-info/*
    /Users/josepedro/pvlabvenv/lib/python3.9/site-packages/pvlab/*
Proceed (Y/n)? y
Successfully uninstalled pvlab-0.1.0.dev7 (desinstalado)
(pvlabvenv) josepedro@iMac-de-Jose ~ % python
Python 3.9.7 (v3.9.7:1016ef3790, Aug 30 2021, 16:39:15)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
[>>> import pvlab (comprobar > importar)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'pvlab' comprobado (error de importación)
>>>
```


Recursos para el Desarrollo de Software

Entornos de Trabajo

Entorno Virtual (Virtual Environment)

Genera y aísla un entorno de trabajo virtual

Puede crearse con la herramienta integrada **venv**.

```
[user] ~ %: python -m venv venvname
```

Se genera **un directorio** que alberga los scripts y paquetes instalados en dicho entorno de trabajo.

Aísla la versión de **python**, así como la de otras **aplicaciones y librerías**:

- Asegura que el entorno futuro de ejecución del proyecto es el mismo que el entorno en el que se desarrolló.
- Previene frente a actualizaciones posteriores no compatibles o que generen errores de ejecución o cálculo (p.ej. "floor division" v2.Vs.v3).

Es recomendable crear un entorno para cada proyecto.

El listado de paquetes puede exportarse, guardarse y cargarse:

```
(venvname) ~%: python -m pip freeze > requirements.txt
```

```
(venvname) ~%: python -m pip install -r requirements.txt
```

```
josepedro - zsh - 90x30
INSTALL UNINSTALL EXPORT BE AWAR... VIEW DE... +

EXPORTAR DEPENDENCIAS

[josepedro@iMac-de-Jose ~ % source pvlabvenv/bin/activate
(pvlabvenv) josepedro@iMac-de-Jose ~ % python -m pip freeze > requirements.txt
(pvlabvenv) josepedro@iMac-de-Jose ~ % cat requirements.txt
alabaster==0.7.12
anyio==3.4.0
appnope==0.1.2
argon2-cffi==21.3.0
argon2-cffi-bindings==21.2.0
attrs==21.2.0
Babel==2.9.1
backcall==0.2.0
bleach==3.3.0
blessings==1.7
bpython==0.21
build==0.7.0
certifi==2021.5.30
cffi==1.15.0
chardet==4.0.0
colorama==0.4.4
coverage==6.0.1
curtsies==0.3.5
cwcwidth==0.1.4
cyclor==0.10.0
DateTime==4.3
debugpy==1.5.1
decorator==5.1.0
```


Recursos para el Desarrollo de Software

Entornos de Trabajo

Entorno Virtual (Virtual Environment)

Genera y aísla un entorno de trabajo virtual

Puede crearse con la herramienta integrada **venv**.

```
[user] ~ %: python -m venv venvname
```

Se genera **un directorio** que alberga los scripts y paquetes instalados en dicho entorno de trabajo.

Aísla la versión de **python**, así como la de otras **aplicaciones y librerías**:

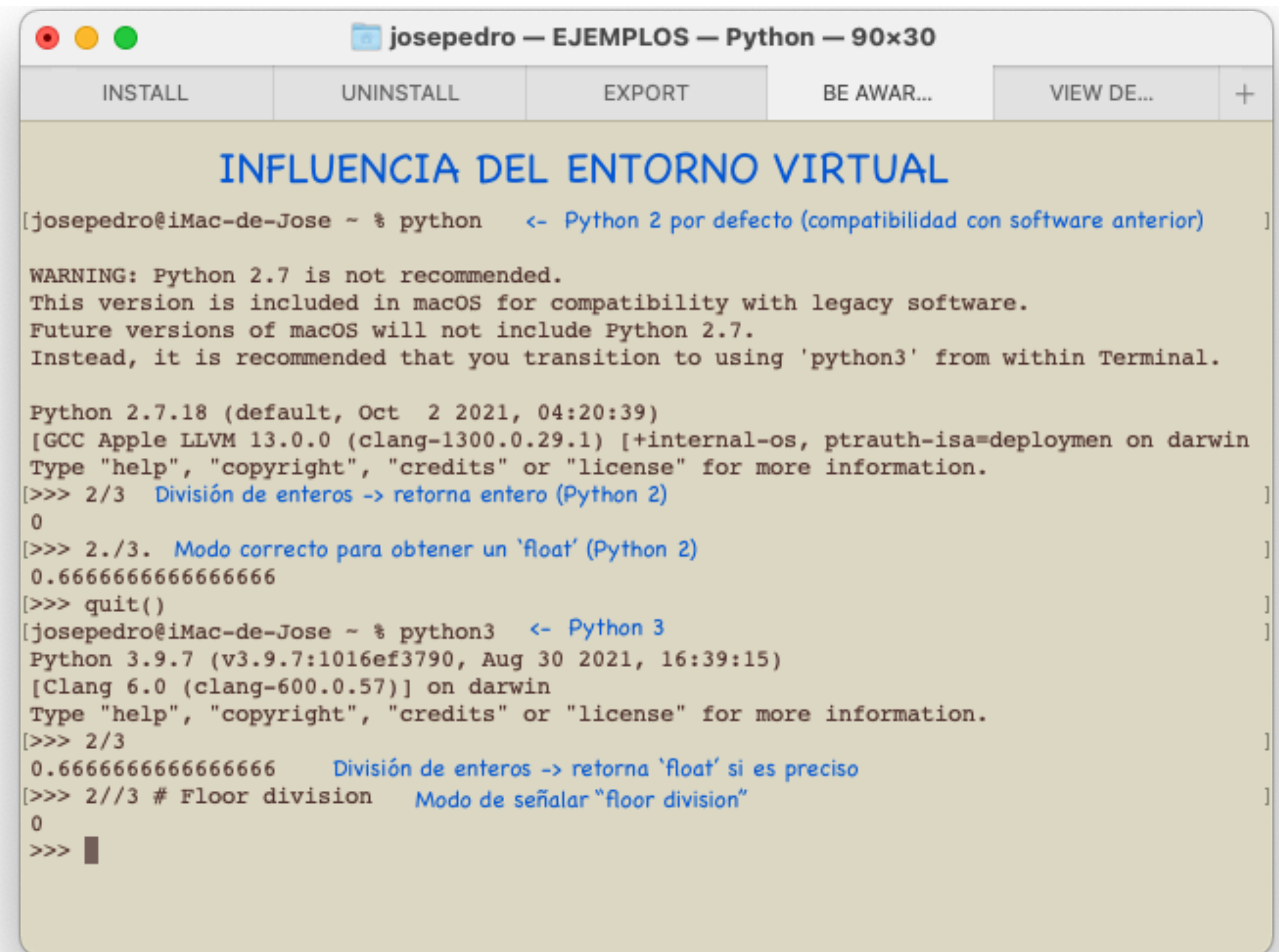
- Asegura que el entorno futuro de ejecución del proyecto es el mismo que el entorno en el que se desarrolló.
- Previene frente a actualizaciones posteriores no compatibles o que generen errores de ejecución o cálculo (p.ej. "floor division" v2.Vs.v3).

Es recomendable crear un entorno para cada proyecto.

El listado de paquetes puede exportarse, guardarse y cargarse:

```
(venvname) ~%: python -m pip freeze > requirements.txt
```

```
(venvname) ~%: python -m pip install -r requirements.txt
```



```
josepedro — EJEMPLOS — Python — 90x30
INSTALL  UNINSTALL  EXPORT  BE AWAR...  VIEW DE...  +

INFLUENCIA DEL ENTORNO VIRTUAL

[josepedro@iMac-de-Jose ~ % python  <- Python 2 por defecto (compatibilidad con software anterior)]

WARNING: Python 2.7 is not recommended.
This version is included in macOS for compatibility with legacy software.
Future versions of macOS will not include Python 2.7.
Instead, it is recommended that you transition to using 'python3' from within Terminal.

Python 2.7.18 (default, Oct  2 2021, 04:20:39)
[GCC Apple LLVM 13.0.0 (clang-1300.0.29.1) [+internal-os, ptrauth-isa=deploymen on darwin
Type "help", "copyright", "credits" or "license" for more information.
[>>> 2/3  División de enteros -> retorna entero (Python 2)
0
[>>> 2./3.  Modo correcto para obtener un 'float' (Python 2)
0.6666666666666666
[>>> quit()
[josepedro@iMac-de-Jose ~ % python3  <- Python 3
Python 3.9.7 (v3.9.7:1016ef3790, Aug 30 2021, 16:39:15)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
[>>> 2/3
0.6666666666666666  División de enteros -> retorna 'float' si es preciso
[>>> 2//3 # Floor division  Modo de señalar "floor division"
0
>>> █
```

Recursos para el Desarrollo de Software

Entornos de Trabajo

Entorno Virtual (Virtual Environment)

Genera y aísla un entorno de trabajo virtual

Puede crearse con la herramienta integrada **venv**.

```
[user] ~ %: python -m venv venvname
```

Se genera **un directorio** que alberga los scripts y paquetes instalados en dicho entorno de trabajo.

Aísla la versión de **python**, así como la de otras **aplicaciones y librerías**:

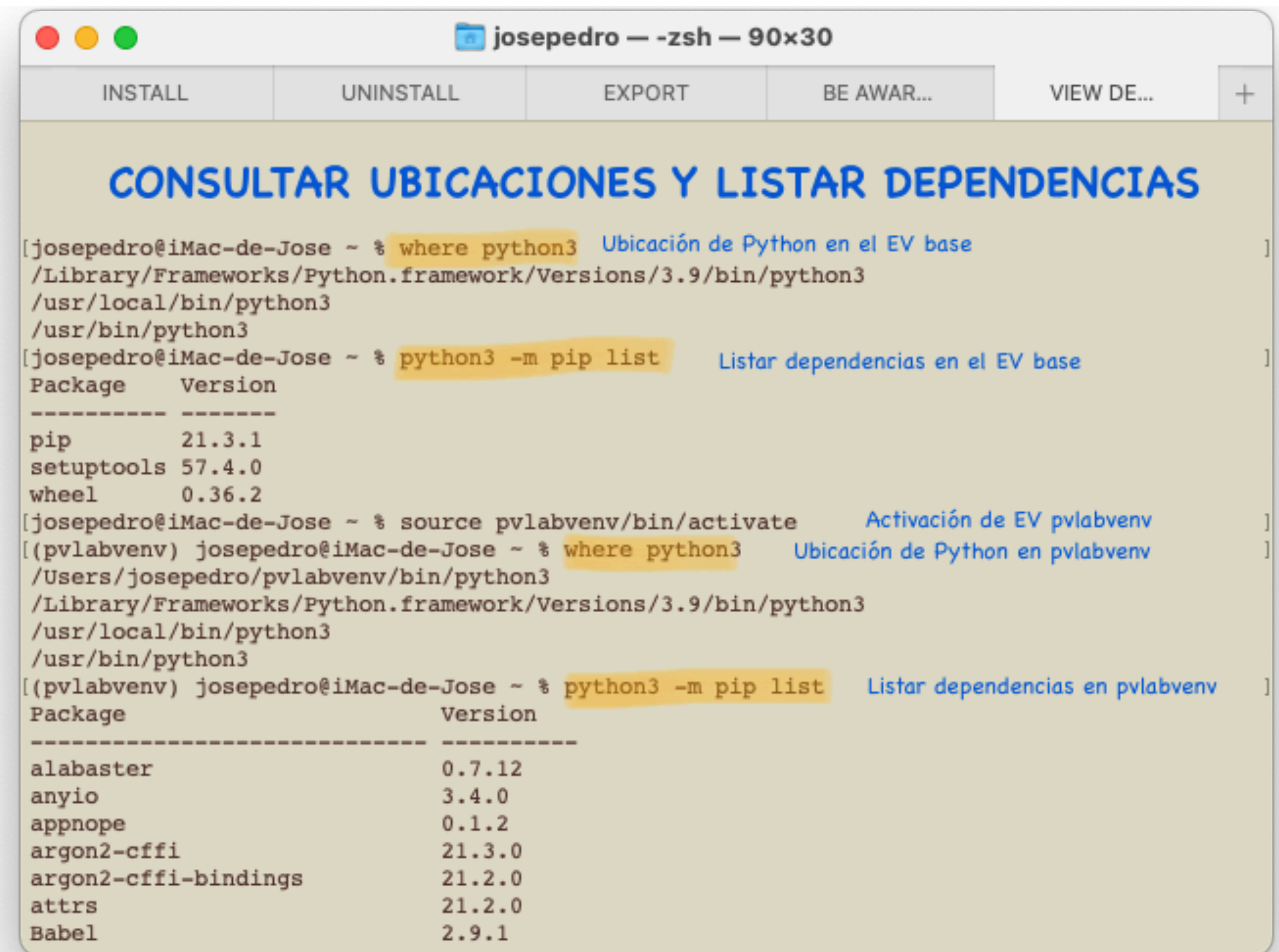
- Asegura que el entorno futuro de ejecución del proyecto es el mismo que el entorno en el que se desarrolló.
- Previene frente a actualizaciones posteriores no compatibles o que generen errores de ejecución o cálculo (p.ej. "floor division" v2.Vs.v3).

Es recomendable crear un entorno para cada proyecto.

El listado de paquetes puede exportarse, guardarse y cargarse:

```
(venvname) ~%: python -m pip freeze > requirements.txt
```

```
(venvname) ~%: python -m pip install -r requirements.txt
```



The screenshot shows a terminal window titled "josepedro — -zsh — 90x30". At the top, there are buttons for "INSTALL", "UNINSTALL", "EXPORT", "BE AWA...", and "VIEW DE...". The main content is titled "CONSULTAR UBICACIONES Y LISTAR DEPENDENCIAS". It shows three terminal sessions:

```
[josepedro@iMac-de-Jose ~ % where python3 Ubicación de Python en el EV base
/Library/Frameworks/Python.framework/Versions/3.9/bin/python3
/usr/local/bin/python3
/usr/bin/python3
[josepedro@iMac-de-Jose ~ % python3 -m pip list Listar dependencias en el EV base
Package      Version
-----
pip          21.3.1
setuptools   57.4.0
wheel        0.36.2
[josepedro@iMac-de-Jose ~ % source pvlabvenv/bin/activate Activación de EV pvlabvenv
(pvlabvenv) josepedro@iMac-de-Jose ~ % where python3 Ubicación de Python en pvlabvenv
/Users/josepedro/pvlabvenv/bin/python3
/Library/Frameworks/Python.framework/Versions/3.9/bin/python3
/usr/local/bin/python3
/usr/bin/python3
(pvlabvenv) josepedro@iMac-de-Jose ~ % python3 -m pip list Listar dependencias en pvlabvenv
Package      Version
-----
alabaster     0.7.12
anyio         3.4.0
appnope       0.1.2
argon2-cffi   21.3.0
argon2-cffi-bindings 21.2.0
attrs         21.2.0
Babel         2.9.1
```

(II - 3/5) Bases de la Programación en Python

Bases de la Programación en Python

Contenido

1. Paquetes, Proyectos y Módulos
2. Objetos y Estructuras
3. Funciones y Decoradores
4. Programación Orientada a Objetos
5. Documentación del código
6. Validación del código

Bases de la Programación en Python

1. Diseño de Paquetes, Proyectos y Módulos

Diseño de Paquetes, Proyectos y Módulos

 **pvlab-0.1.0.dev7**

 pyproject.toml

 .gitignore

 LICENSE.txt

 README.txt

 dist

 src

 **pvlab** 

 **pvlab**

 **__init__.py**

 doc

 __pycache__

 [dataframes]

 [io]

 [math]

  test

Diseño de Paquetes, Proyectos y Módulos

Módulos

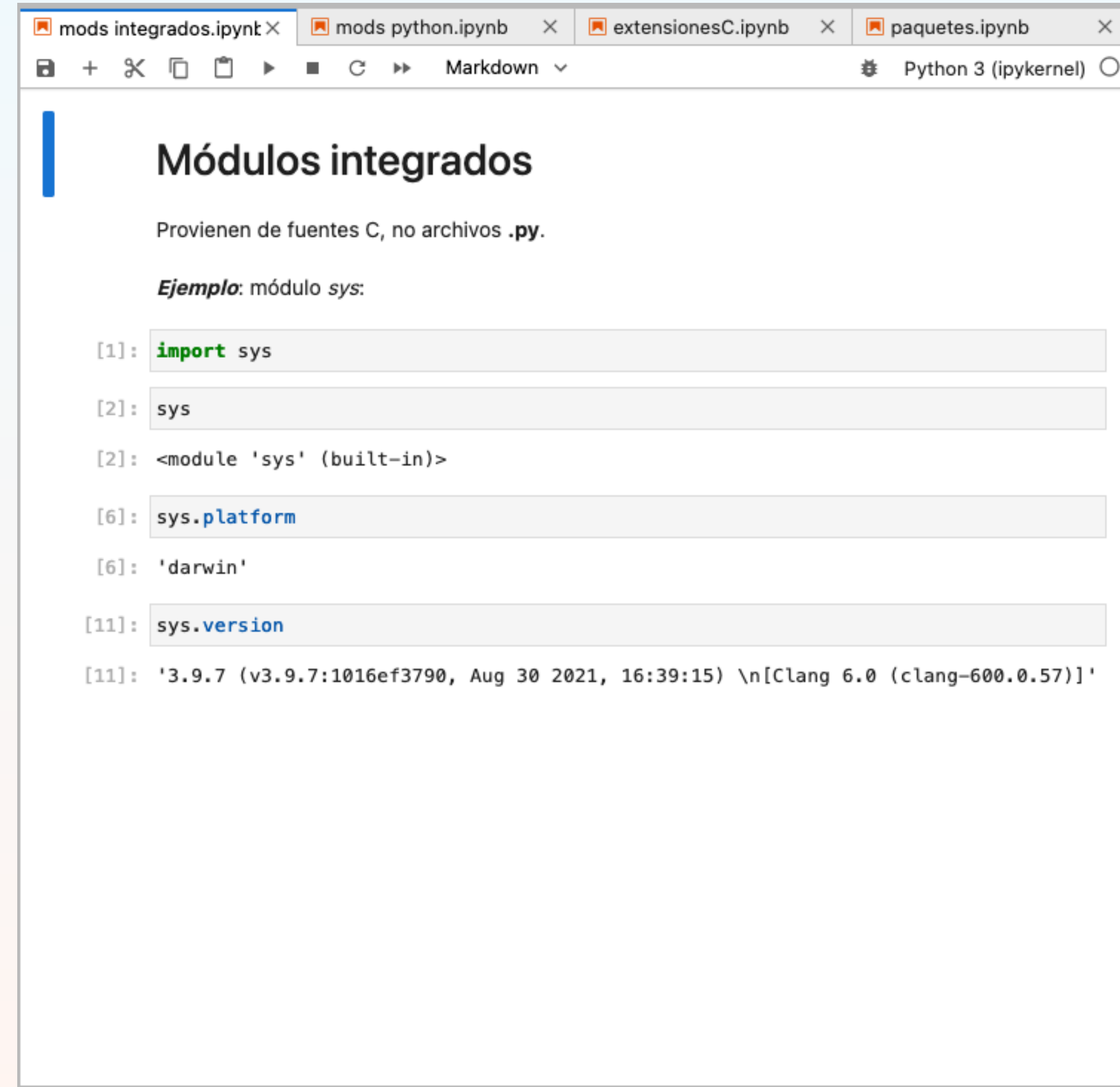
Los módulos son las unidades de software a **importar**, que contienen código Python (scripts, funciones, clases,...). Pueden ser de diversos tipos:

1. Python: archivo **.py** con (cero o más) instrucciones Python (**scripts**, **híbridos**, **contenedores** de funciones y/o clases).
2. Compilados: archivos **.pyc** (bytecode), en la carpeta `__pycache__`.
3. Integrados (built-in): fuentes C vinculadas estáticamente al intérprete.
4. Extensiones: extensiones C compiladas como librerías dinámicas (**.dll** o **.so**).

Paquetes

Contienen sub-paquetes y/o módulos:

1. Contienen un archivo llamado **__init__.py** (aunque esté vacío).
2. Todos los paquetes son también módulos en sí mismos, pero no al contrario.
3. Al importar un paquete, se importa su archivo "de inicio" **__init__.py**. Usualmente, éste contiene la descripción del módulo y su atributo "mágico" **__version__** esenciales para la distribución del proyecto.



The screenshot shows a Jupyter Notebook with four tabs: 'mods integrados.ipynk', 'mods python.ipynb', 'extensionesC.ipynb', and 'paquetes.ipynb'. The active tab is 'mods integrados.ipynk'. The notebook content is titled 'Módulos integrados' and includes the text 'Proviene de fuentes C, no archivos .py.' and 'Ejemplo: módulo sys:'. The code cells show the following interactions:

```
[1]: import sys
[2]: sys
[2]: <module 'sys' (built-in)>
[6]: sys.platform
[6]: 'darwin'
[11]: sys.version
[11]: '3.9.7 (v3.9.7:1016ef3790, Aug 30 2021, 16:39:15) \n[Clang 6.0 (clang-600.0.57)]'
```

Diseño de Paquetes, Proyectos y Módulos

Módulos

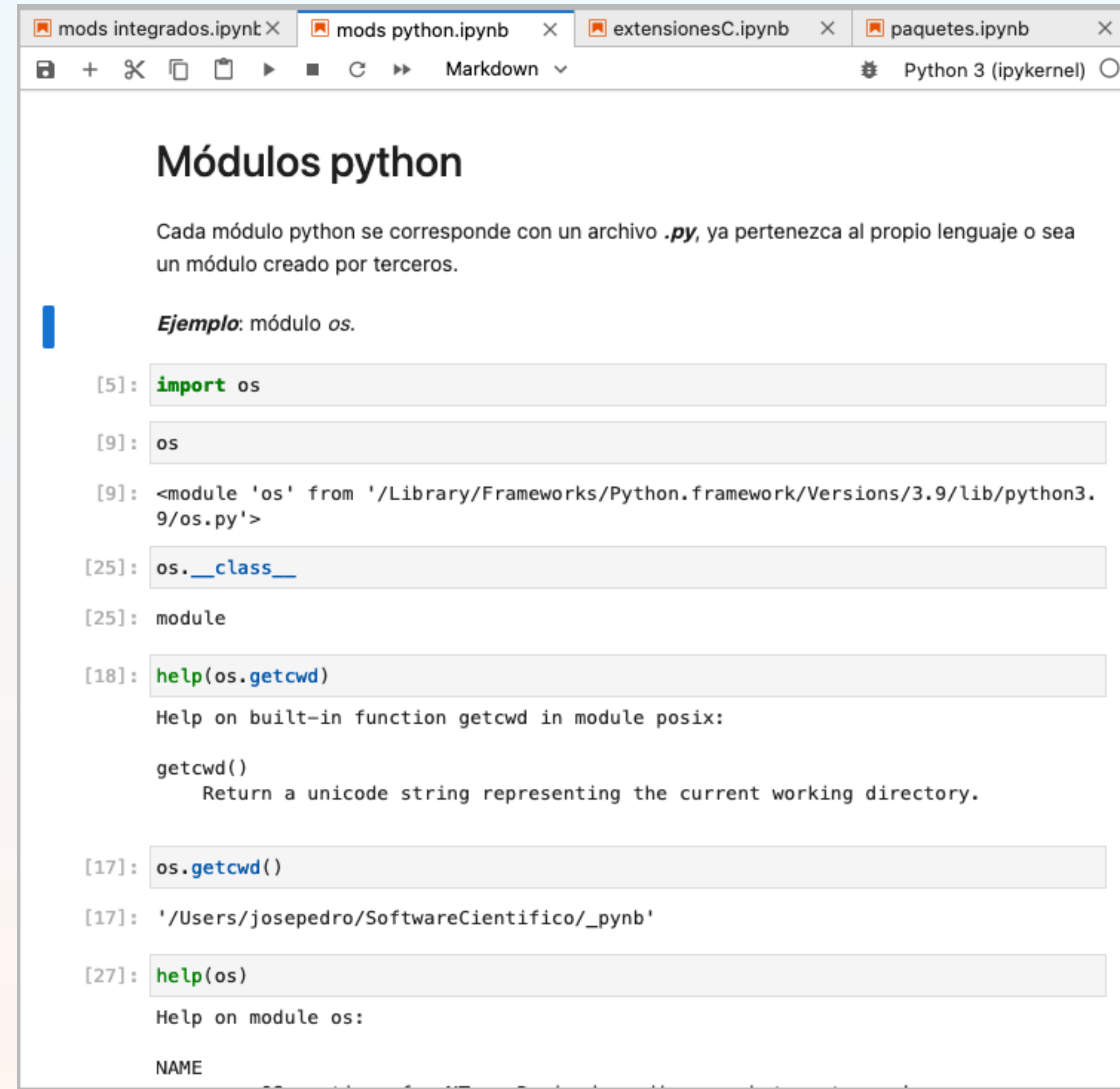
Los módulos son las unidades de software a **importar**, que contienen código Python (*scripts, funciones, clases,...*). Pueden ser de diversos tipos:

1. Python: archivo **.py** con (cero o más) instrucciones Python (**scripts, híbridos, contenedores** de funciones y/o clases).
2. Compilados: archivos **.pyc** (bytecode), en la carpeta `__pycache__`.
3. Integrados (built-in): fuentes C vinculadas estáticamente al intérprete.
4. Extensiones: extensiones C compiladas como librerías dinámicas (**.dll** o **.so**).

Paquetes

Contienen sub-paquetes y/o módulos:

1. Contienen un archivo llamado **__init__.py** (aunque esté vacío).
2. Todos los paquetes son también módulos en sí mismos, pero no al contrario.
3. Al importar un paquete, se importa su archivo "de inicio" **__init__.py**. Usualmente, éste contiene la descripción del módulo y su atributo "mágico" **__version__** esenciales para la distribución del proyecto.



The screenshot shows a Jupyter Notebook interface with four tabs: 'mods integrados.ipynb', 'mods python.ipynb', 'extensionesC.ipynb', and 'paquetes.ipynb'. The 'mods python.ipynb' tab is active, displaying a series of code cells and their outputs. The first cell imports the 'os' module. The second cell prints the module object. The third cell prints the module's class. The fourth cell prints the module object again. The fifth cell uses 'help(os.getcwd())' to show the documentation for the 'getcwd' function. The sixth cell prints the result of 'os.getcwd()'. The seventh cell uses 'help(os)' to show the documentation for the 'os' module.

```
[5]: import os

[9]: os

[9]: <module 'os' from '/Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/os.py'>

[25]: os.__class__

[25]: module

[18]: help(os.getcwd)

Help on built-in function getcwd in module posix:

getcwd()
    Return a unicode string representing the current working directory.

[17]: os.getcwd()

[17]: '/Users/josepedro/SoftwareCientifico/_pynb'

[27]: help(os)

Help on module os:

NAME
```

Diseño de Paquetes, Proyectos y Módulos

Módulos

Los módulos son las unidades de software a **importar**, que contienen código Python (*scripts, funciones, clases,...*). Pueden ser de diversos tipos:

1. Python: archivo **.py** con (cero o más) instrucciones Python (**scripts, híbridos, contenedores** de funciones y/o clases).
2. Compilados: archivos **.pyc** (bytecode), en la carpeta `__pycache__`.
3. Integrados (built-in): fuentes C vinculadas estáticamente al intérprete.
4. Extensiones: extensiones C compiladas como librerías dinámicas (**.dll** o **.so**).

Paquetes

Contienen sub-paquetes y/o módulos:

1. Contienen un archivo llamado **__init__.py** (aunque esté vacío).
2. Todos los paquetes son también módulos en sí mismos, pero no al contrario.
3. Al importar un paquete, se importa su archivo "de inicio" **__init__.py**. Usualmente, éste contiene la descripción del módulo y su atributo "mágico" **__version__** esenciales para la distribución del proyecto.

```
mods integrados.ipynb X mods python.ipynb X extensionesC.ipynb X paquetes.ipynb X
Python 3 (ipykernel)

Extensiones C

Extensiones C compiladas como librerías dinámicas. Cada módulo python se corresponde con
un archivo .dll (Windows) o .so (Unix).

Ejemplo: módulo math.

[10]: import math

[11]: math

[11]: <module 'math' from '/Library/Frameworks/Python.framework/Versions/3.9/lib/python
3.9/lib-dynload/math.cpython-39-darwin.so'>

[12]: math.sin(math.pi/2)

[12]: 1.0

[13]: math.e

[13]: 2.718281828459045
```


Diseño de Paquetes, Proyectos y Módulos

Módulos

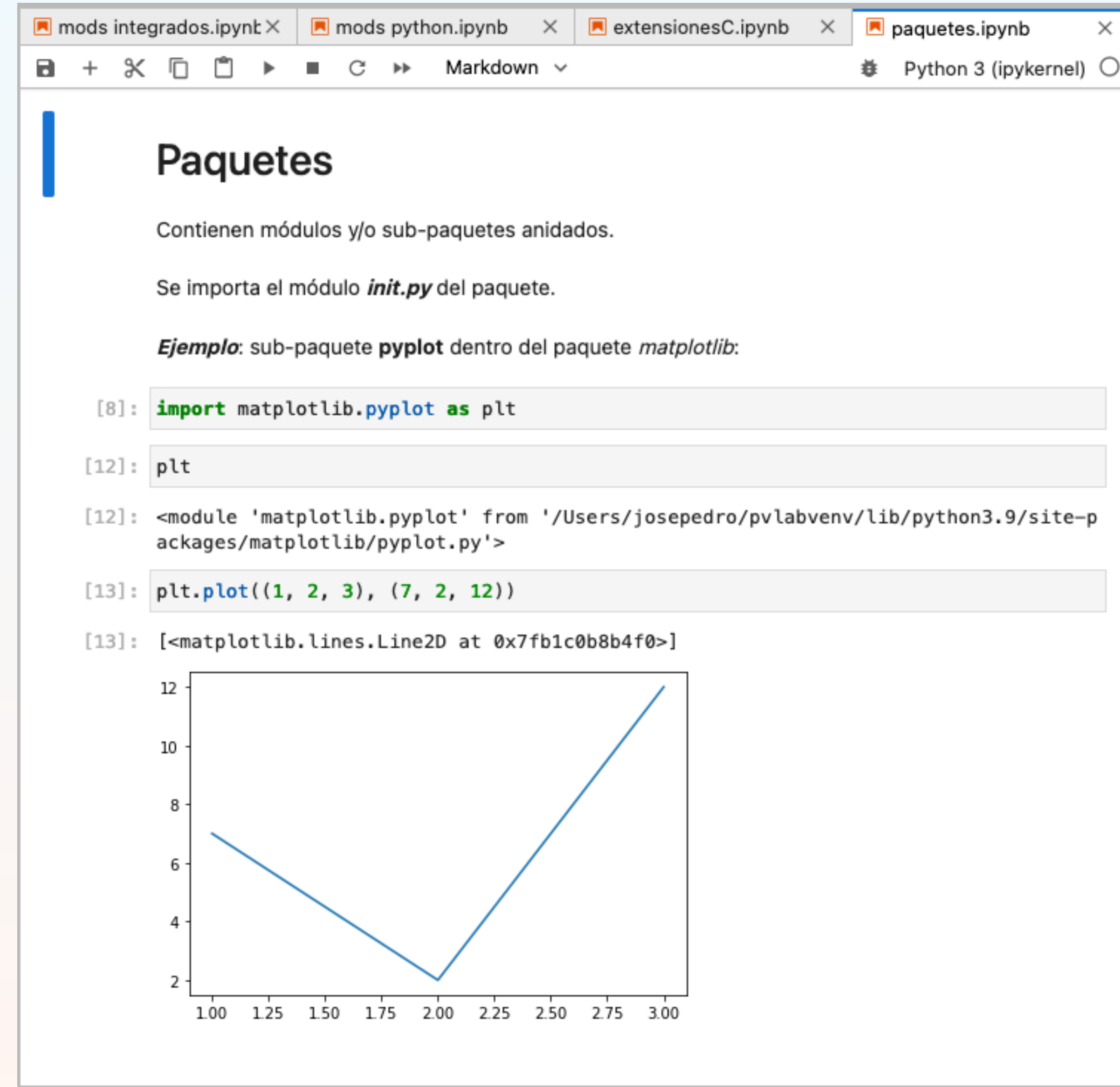
Los módulos son las unidades de software a **importar**, que contienen código Python (*scripts, funciones, clases,...*). Pueden ser de diversos tipos:

1. Python: archivo **.py** con (cero o más) instrucciones Python (**scripts, híbridos, contenedores** de funciones y/o clases).
2. Compilados: archivos **.pyc** (bytecode), en la carpeta `__pycache__`.
3. Integrados (built-in): fuentes C vinculadas estáticamente al intérprete.
4. Extensiones: extensiones C compiladas como librerías dinámicas (**.dll** o **.so**).

Paquetes

Contienen sub-paquetes y/o módulos:

1. Contienen un archivo llamado **__init__.py** (aunque esté vacío).
2. Todos los paquetes son también módulos en sí mismos, pero no al contrario.
3. Al importar un paquete, se importa su archivo "de inicio" **__init__.py**. Usualmente, éste contiene la descripción del módulo y su atributo "mágico" **__version__** esenciales para la distribución del proyecto.



The screenshot shows a Jupyter Notebook interface with the title 'paquetes.ipynb'. The notebook content is titled 'Paquetes' and includes the following text:

Contienen módulos y/o sub-paquetes anidados.

Se importa el módulo **init.py** del paquete.

Ejemplo: sub-paquete **pyplot** dentro del paquete **matplotlib**:

```
[8]: import matplotlib.pyplot as plt
```

```
[12]: plt
```

```
[12]: <module 'matplotlib.pyplot' from '/Users/josepedro/pvlabvenv/lib/python3.9/site-packages/matplotlib/pyplot.py'>
```

```
[13]: plt.plot((1, 2, 3), (7, 2, 12))
```

```
[13]: [<matplotlib.lines.Line2D at 0x7fb1c0b8b4f0>]
```

Below the code cells, a plot is displayed. The x-axis ranges from 1.00 to 3.00 with major ticks every 0.25. The y-axis ranges from 2 to 12 with major ticks every 2. The plot shows a blue line connecting the points (1, 7), (2, 2), and (3, 12), forming a V-shape.

Bases de la Programación en Python

2. Objetos y Estructuras Integradas

Objetos y Estructuras Integradas

Principales objetos Integrados en la Standard Python Library

Booleanos (bool): True (0), False (False)

Enteros (int): 0, 1, 2, ...

Valores en coma flotante (float): 0.2434, 7.2135e3, ...

Caracteres/cadenas: 'uno', 'hola', 'h', '1', "", ...

Conjuntos (set): {1, 5, 3}, {'hola', 'mundo'}

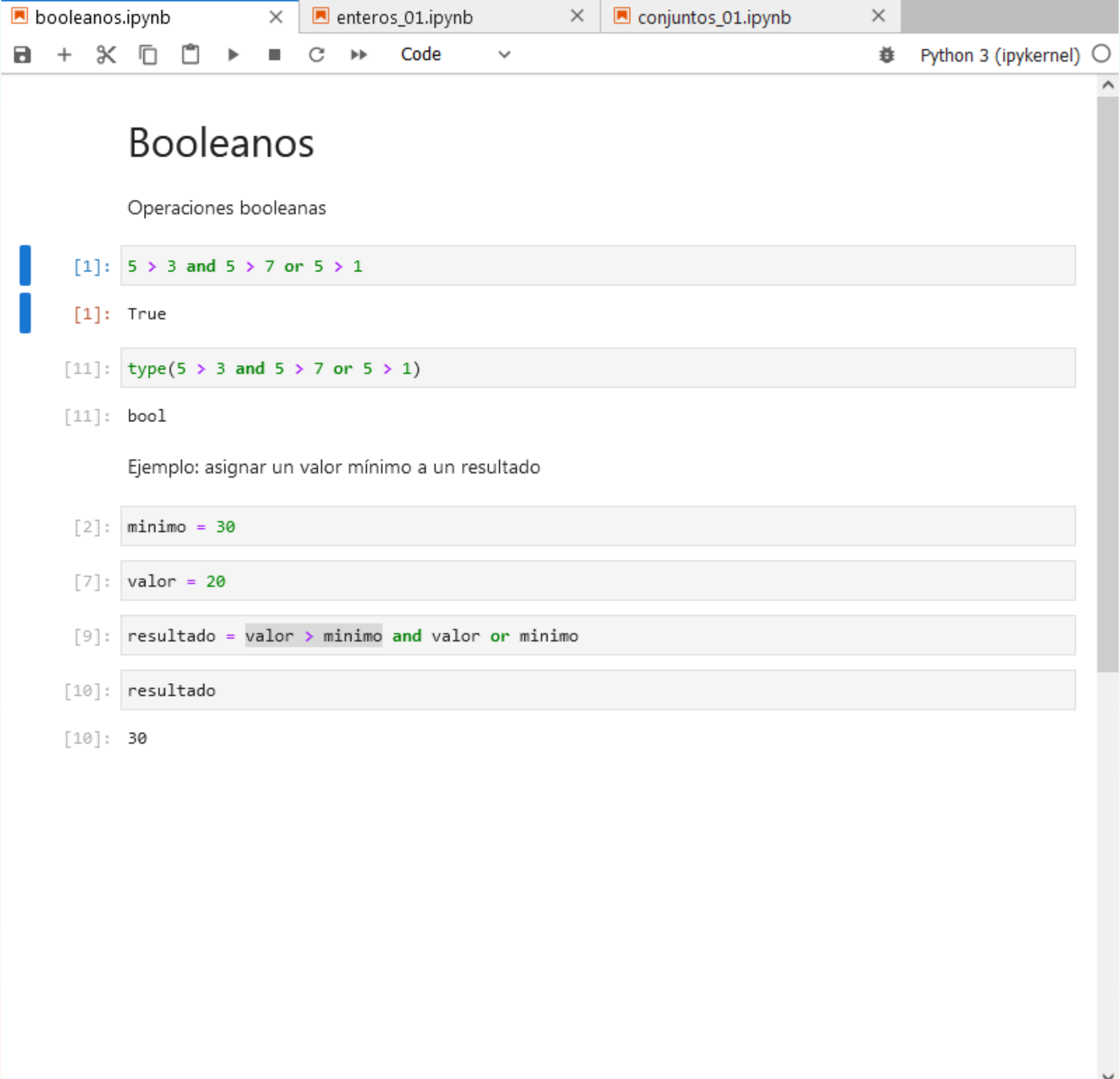
Lista (list): [3, 4, 5, 6], ['letraA', 'letraB', 'letraC'], ...

Diccionarios (dict): {'key1': 'value1', 'key2': 'value2'}

Funciones (function): hasattr()

Métodos (method...): list.append, set.add, dict.update, ...

clases (type): int, float, function, ...



The screenshot shows a Jupyter Notebook interface with three tabs: 'booleanos.ipynb', 'enteros_01.ipynb', and 'conjuntos_01.ipynb'. The 'booleanos.ipynb' tab is active, displaying the title 'Booleanos' and the subtitle 'Operaciones booleanas'. The notebook contains several code cells:

- Cell [1]: `5 > 3 and 5 > 7 or 5 > 1` (input) and `True` (output).
- Cell [11]: `type(5 > 3 and 5 > 7 or 5 > 1)` (input) and `bool` (output).
- Text: 'Ejemplo: asignar un valor mínimo a un resultado'.
- Cell [2]: `minimo = 30` (input) and `30` (output).
- Cell [7]: `valor = 20` (input) and `20` (output).
- Cell [9]: `resultado = valor > minimo and valor or minimo` (input) and `30` (output).
- Cell [10]: `resultado` (input) and `30` (output).

Objetos y Estructuras Integradas

Principales objetos Integrados en la Standard Python Library

Booleanos (bool): True (0), False (False)

Enteros (int): 0, 1, 2, ...

Valores en coma flotante (float): 0.2434, 7.2135e3, ...

Caracteres/cadenas: 'uno', 'hola', 'h', '1', "", ...

Conjuntos (set): {1, 5, 3}, {'hola', 'mundo'}

Lista (list): [3, 4, 5, 6], ['letraA', 'letraB', 'letraC'], ...

Diccionarios (dict): {'key1': 'value1', 'key2': 'value2'}

Funciones (function): hasattr()

Métodos (method...): list.append, set.add, dict.update, ...

clases (type): int, float, function, ...

The screenshot shows a Jupyter Notebook interface with three tabs: 'booleanos.ipynb', 'enteros_01.ipynb' (active), and 'conjuntos_01.ipynb'. The active notebook is titled 'Enteros' and contains the following content:

Operaciones con enteros:

```
[10]: 3 + 5 - 2
[10]: 6
```

• [13]: `round(2/3, 2)` # División estandar

```
[13]: 0.67
```

[14]: `round(2//3, 2)` # División truncada (floor division)

```
[14]: 0
```

Modelo de objetos (enteros):

```
[16]: int('32') # Conversión a entero
[16]: 32
```

[17]: `int('32').__add__` # Método __add__ del tipo int (asignado al operador '+')

```
[17]: <method-wrapper '__add__' of int object at 0x000007FEE9B28020>
```

[18]: `int('32').__add__(3)` # Llamada al método __add__

```
[18]: 35
```

Objetos y Estructuras Integradas

Principales objetos Integrados en la Standard Python Library

Booleanos (bool): True (0), False (False)

Enteros (int): 0, 1, 2, ...

Valores en coma flotante (float): 0.2434, 7.2135e3, ...

Caracteres/cadenas: 'uno', 'hola', 'h', '1', "", ...

Conjuntos (set): {1, 5, 3}, {'hola', 'mundo'}

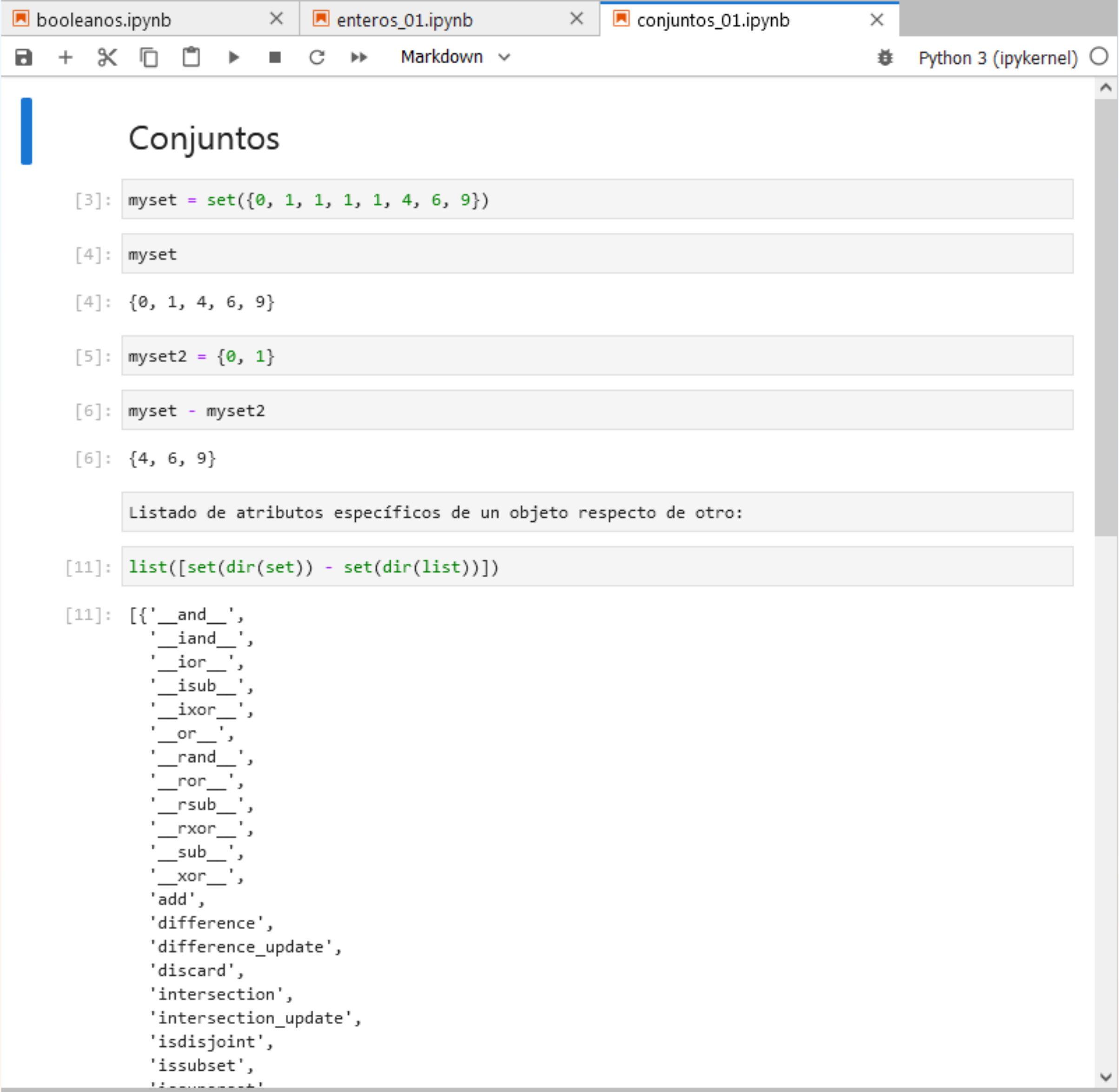
Lista (list): [3, 4, 5, 6], ['letraA', 'letraB', 'letraC'], ...

Diccionarios (dict): {'key1': 'value1', 'key2': 'value2'}

Funciones (function): hasattr()

Métodos (method...): list.append, set.add, dict.update, ...

clases (type): int, float, function, ...



The screenshot shows a Jupyter Notebook interface with three tabs: 'booleanos.ipynb', 'enteros_01.ipynb', and 'conjuntos_01.ipynb'. The active tab is 'conjuntos_01.ipynb'. The notebook content is as follows:

```
Conjuntos

[3]: myset = set({0, 1, 1, 1, 1, 4, 6, 9})

[4]: myset

[4]: {0, 1, 4, 6, 9}

[5]: myset2 = {0, 1}

[6]: myset - myset2

[6]: {4, 6, 9}

Listado de atributos específicos de un objeto respecto de otro:

[11]: list([set(dir(set)) - set(dir(list))])

[11]: [['_and_',
['_iand_',
['_ior_',
['_isub_',
['_ixor_',
['_or_',
['_rand_',
['_ror_',
['_rsub_',
['_rxor_',
['_sub_',
['_xor_',
'add',
'difference',
'difference_update',
'discard',
'intersection',
'intersection_update',
'isdisjoint',
'issubset',
'issuperset']]
```

Objetos y Estructuras Integradas

Principales objetos Integrados en la Standard Python Library

Booleanos (bool): True (0), False (False)

Enteros (int): 0, 1, 2, ...

Valores en coma flotante (float): 0.2434, 7.2135e3, ...

Caracteres/cadenas: 'uno', 'hola', 'h', '1', '', ...

Conjuntos (set): {1, 5, 3}, {'hola', 'mundo'}

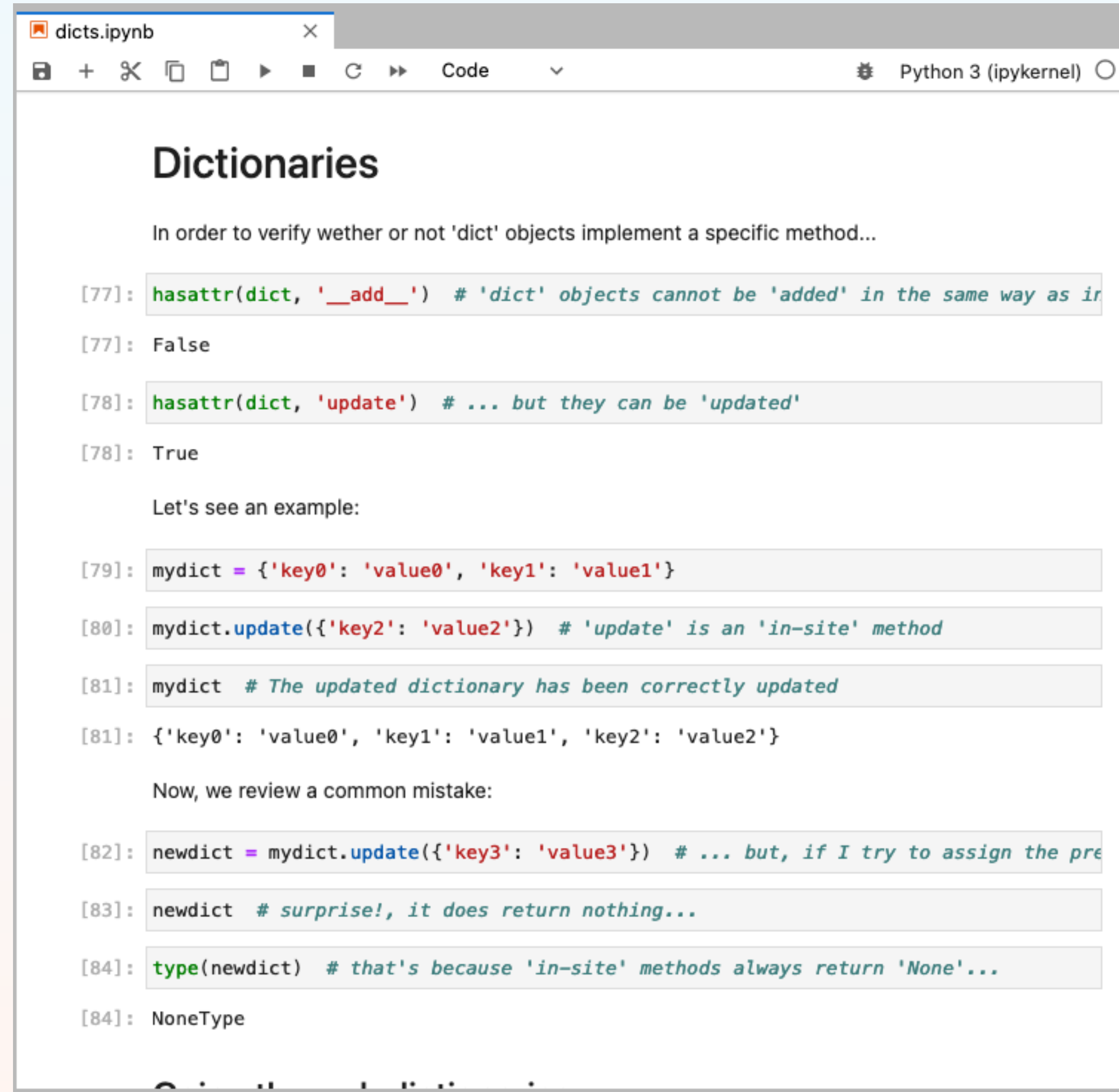
Lista (list): [3, 4, 5, 6], ['letraA', 'letraB', 'letraC'], ...

Diccionarios (dict): {'key1': 'value1', 'key2': 'value2'}

Funciones (function): hasattr()

Métodos (method...): list.append, set.add, dict.update, ...

clases (type): int, float, function, ...



```
dicts.ipynb
Python 3 (ipykernel)

Dictionaries

In order to verify wether or not 'dict' objects implement a specific method...

[77]: hasattr(dict, '__add__') # 'dict' objects cannot be 'added' in the same way as in
[77]: False

[78]: hasattr(dict, 'update') # ... but they can be 'updated'
[78]: True

Let's see an example:

[79]: mydict = {'key0': 'value0', 'key1': 'value1'}

[80]: mydict.update({'key2': 'value2'}) # 'update' is an 'in-site' method

[81]: mydict # The updated dictionary has been correctly updated
[81]: {'key0': 'value0', 'key1': 'value1', 'key2': 'value2'}

Now, we review a common mistake:

[82]: newdict = mydict.update({'key3': 'value3'}) # ... but, if I try to assign the pre
[83]: newdict # surprise!, it does return nothing...
[84]: type(newdict) # that's because 'in-site' methods always return 'None'...
[84]: NoneType
```


Objetos y Estructuras Integradas

Principales objetos Integrados en la Standard Python Library

Booleanos (bool): True (0), False (False)

Enteros (int): 0, 1, 2, ...

Valores en coma flotante (float): 0.2434, 7.2135e3, ...

Caracteres/cadenas: 'uno', 'hola', 'h', '1', '', ...

Conjuntos (set): {1, 5, 3}, {'hola', 'mundo'}

Lista (list): [3, 4, 5, 6], ['letraA', 'letraB', 'letraC'], ...

Diccionarios (dict): {'key1': 'value1', 'key2': 'value2'}

Funciones (function): hasattr()

Métodos (method...): list.append, set.add, dict.update, ...

clases (type): int, float, function, ...

```
dicts.ipynb
+ ✂ 📄 ▶ ■ ↺ ▶▶ Code Python 3 (ipykernel)

Going through dictionaries

[85]: for key in mydict: # dict elements are identified by their key...
      print(key)

key0
key1
key2
key3

We can do that because 'dict' objects are iterables...let's check this assessment:

It can be manually created an iterator over an object applying the integrated function 'iter' to the object ...

[86]: i = iter(mydict)

and then going through its elements one by one by using the '__next__()' method or calling to the 'next()' function...

[87]: [i.__next__(), i.__next__(), i.__next__(), i.__next__()]

[87]: ['key0', 'key1', 'key2', 'key3']

An additional instruction in the same form would raise an 'StopIteration' exception...

If our desire is to enumerate the entries of the dictionary, we should apply the integrated function 'enumerate':

[88]: for n, key in enumerate(mydict): # but they can also be 'enumerated'...
      print(f'Number {n}: {key}')

Number 0: key0
Number 1: key1
Number 2: kev2
```

Objetos y Estructuras Integradas

Principales objetos Integrados en la Standard Python Library

Booleanos (bool): True (0), False (False)

Enteros (int): 0, 1, 2, ...

Valores en coma flotante (float): 0.2434, 7.2135e3, ...

Caracteres/cadenas: 'uno', 'hola', 'h', '1', "", ...

Conjuntos (set): {1, 5, 3}, {'hola', 'mundo'}

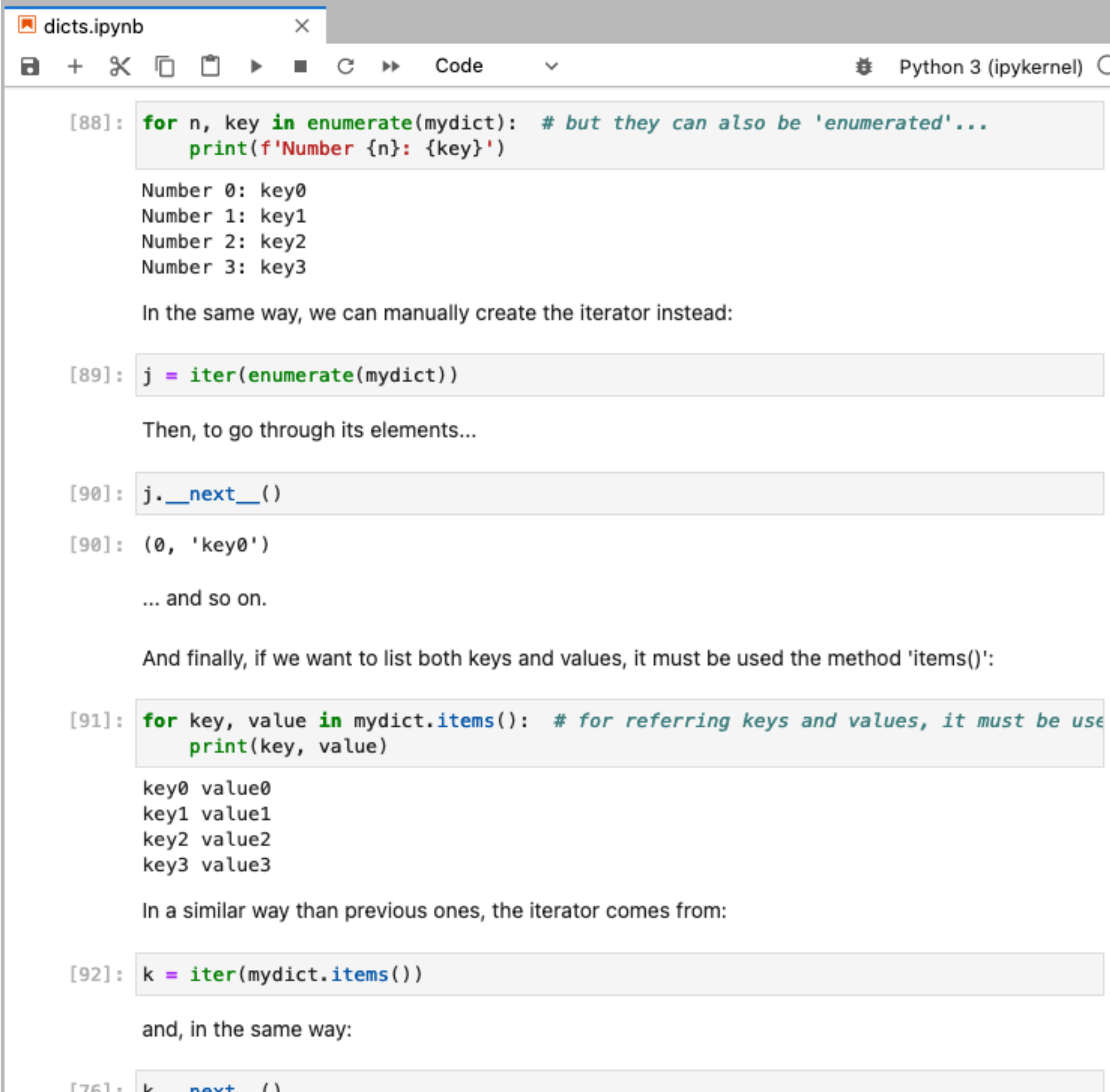
Lista (list): [3, 4, 5, 6], ['letraA', 'letraB', 'letraC'], ...

Diccionarios (dict): {'key1': 'value1', 'key2': 'value2'}

Funciones (function): hasattr()

Métodos (method...): list.append, set.add, dict.update, ...

clases (type): int, float, function, ...



```
dicts.ipynb
Python 3 (ipykernel)

[88]: for n, key in enumerate(mydict): # but they can also be 'enumerated'...
      print(f'Number {n}: {key}')

Number 0: key0
Number 1: key1
Number 2: key2
Number 3: key3

In the same way, we can manually create the iterator instead:

[89]: j = iter(enumerate(mydict))

Then, to go through its elements...

[90]: j.__next__()

[90]: (0, 'key0')

... and so on.

And finally, if we want to list both keys and values, it must be used the method 'items()':

[91]: for key, value in mydict.items(): # for referring keys and values, it must be use
      print(key, value)

key0 value0
key1 value1
key2 value2
key3 value3

In a similar way than previous ones, the iterator comes from:

[92]: k = iter(mydict.items())

and, in the same way:

[76]: k.__next__()
```

Objetos y Estructuras Integradas

Principales objetos Integrados en la Standard Python Library

Booleanos (bool): True (0), False (False)

Enteros (int): 0, 1, 2, ...

Valores en coma flotante (float): 0.2434, 7.2135e3, ...

Caracteres/cadenas: 'uno', 'hola', 'h', '1', "", ...

Conjuntos (set): {1, 5, 3}, {'hola', 'mundo'}

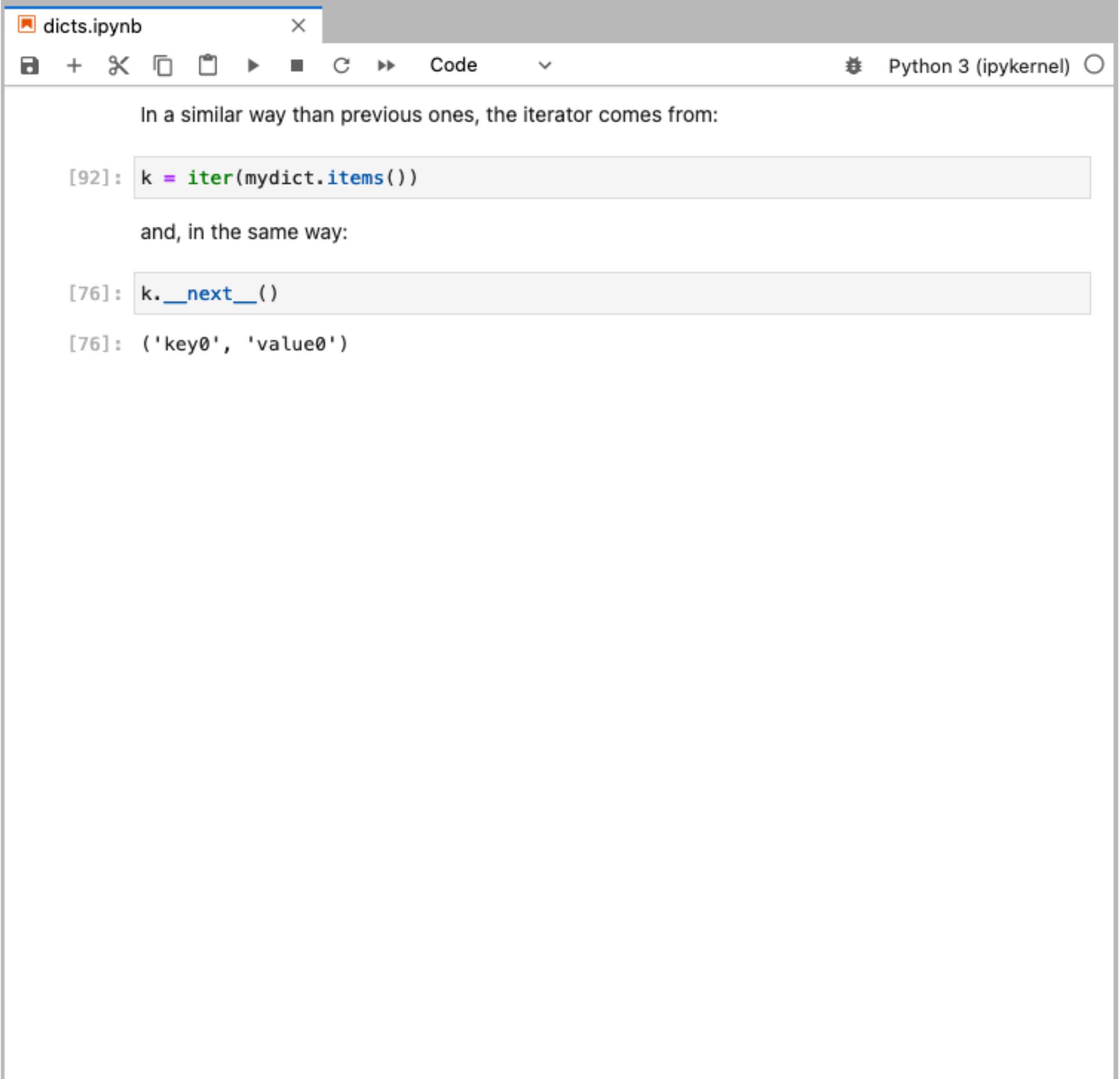
Lista (list): [3, 4, 5, 6], ['letraA', 'letraB', 'letraC'], ...

Diccionarios (dict): {'key1': 'value1', 'key2': 'value2'}

Funciones (function): hasattr()

Métodos (method...): list.append, set.add, dict.update, ...

clases (type): int, float, function, ...



```
dicts.ipynb
Python 3 (ipykernel)

In a similar way than previous ones, the iterator comes from:

[92]: k = iter(mydict.items())

and, in the same way:

[76]: k.__next__()

[76]: ('key0', 'value0')
```


Objetos y Estructuras Integradas

Principales Estructuras Integradas en la Standard Python Library

- A. Bucle **for**: recorre cualquier objeto que tenga definido el método `'__iter__'`.
- B. Bucle **while**: repite la ejecución de cualquier código mientras se cumpla la condición especificada.
- C. Condicional **if...elif...else**: ejecuta el código enmarcado bajo las respectivas instrucciones `if`, `elif` o `else`.
 - A. Instrucción `continue`: pasa a la siguiente iteración.
 - B. Instrucción `break`: finaliza el bucle.
- D. Estructura **try...except...else...finally**:
 - A. Intenta ejecutar el código bajo la instrucción `try`.
 - B. En caso negativo (excepciones), ejecuta el código bajo `except`.
 - C. Si no han ocurrido excepciones, ejecuta el código bajo `else`.
 - D. En cualquier caso, ejecuta el código bajo `finally`.

The screenshot shows a Jupyter Notebook interface with the following content:

Estructuras Integradas en Python

Bucle for

```
[1]: for chr in 'Hello World':  
      print(chr)
```

H
e
l
l
o

W
o
r
l
d

```
[3]: for item in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]:  
      if item%2 == 0:  
          print(item)
```

0
2
4
6
8

```
[10]: hasattr('Hello World', '__iter__')
```

[10]: True

```
[11]: hasattr([1, 2, 3], '__iter__')
```

[11]: True

Objetos y Estructuras Integradas

Principales Estructuras Integradas en la Standard Python Library

- A. Bucle **for**: recorre cualquier objeto que tenga definido el método `'__iter__'`.
- B. Bucle **while**: repite la ejecución de cualquier código mientras se cumpla la condición especificada.
- C. Condicional **if...elif...else**: ejecuta el código enmarcado bajo las respectivas instrucciones `if`, `elif` o `else`.
 - A. Instrucción `continue`: pasa a la siguiente iteración.
 - B. Instrucción `break`: finaliza el bucle.
- D. Estructura **try...except...else...finally**:
 - A. Intenta ejecutar el código bajo la instrucción `try`.
 - B. En caso negativo (excepciones), ejecuta el código bajo `except`.
 - C. Si no han ocurrido excepciones, ejecuta el código bajo `else`.
 - D. En cualquier caso, ejecuta el código bajo `finally`.

```
estructuras_02.ipynb
Python 3 (ipykernel)

Bucle While

[6]: myrange = range(0, 11)

[7]: print(myrange)
range(0, 11)

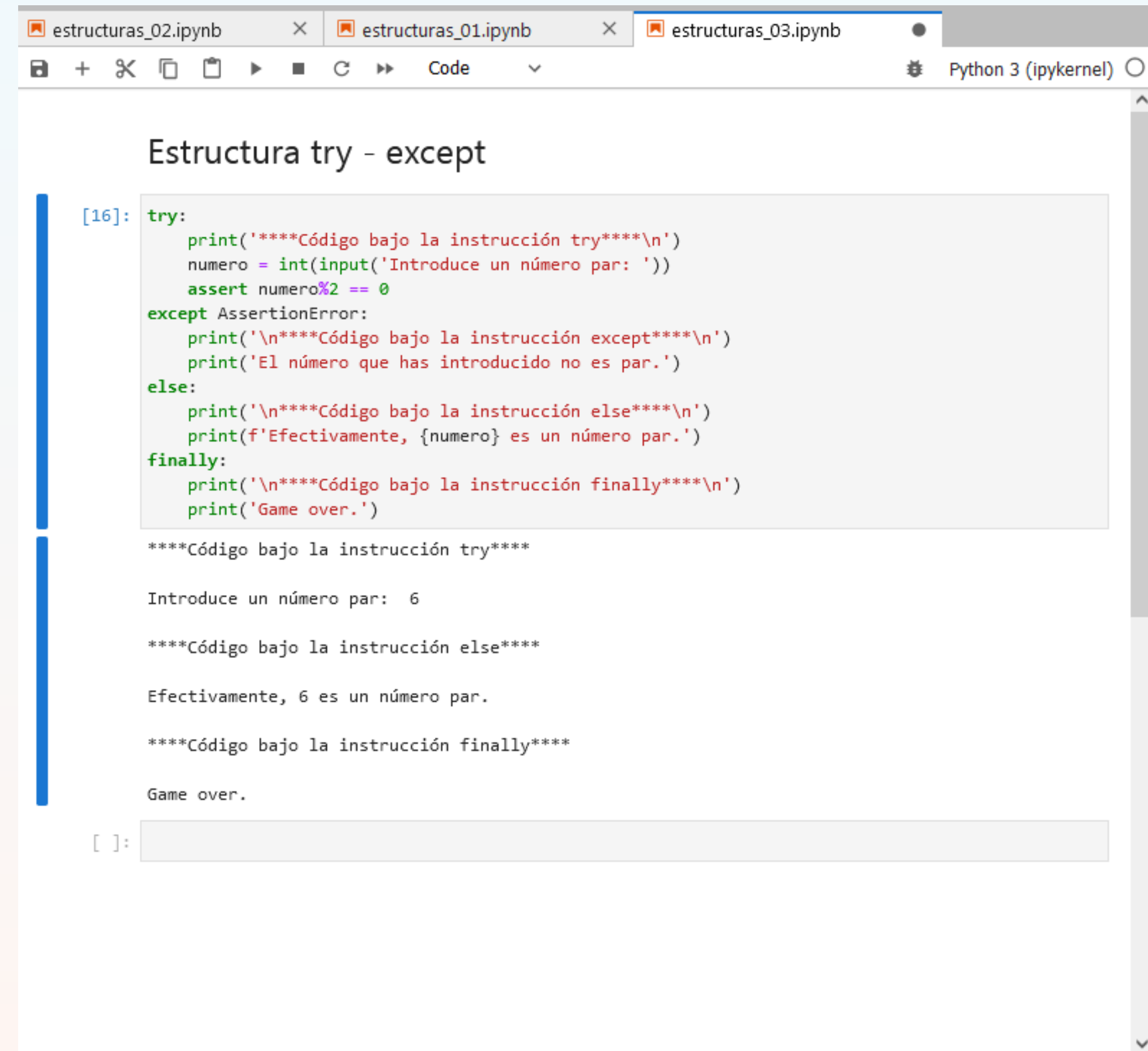
[12]: while True:
        for chr in myrange:
            if chr%2 == 0:
                print(chr, 'is even')
                if chr == 6:
                    print('End of loop FOR')
                    break
                continue
            elif chr%2 != 0:
                print(chr, 'is odd')
                continue
        print('End of loop WHILE')
        break
        print('After break: this message never appears')
```

0 is even
1 is odd
2 is even
3 is odd
4 is even
5 is odd
6 is even
End of loop FOR
End of loop WHILE

Objetos y Estructuras Integradas

Principales Estructuras Integradas en la Standard Python Library

- A. Bucle **for**: recorre cualquier objeto que tenga definido el método `'__iter__'`.
- B. Bucle **while**: repite la ejecución de cualquier código mientras se cumpla la condición especificada.
- C. Condicional **if...elif...else**: ejecuta el código enmarcado bajo las respectivas instrucciones `if`, `elif` o `else`.
 - A. Instrucción `continue`: pasa a la siguiente iteración.
 - B. Instrucción `break`: finaliza el bucle.
- D. Estructura **try...except...else...finally**:
 - A. Intenta ejecutar el código bajo la instrucción `try`.
 - B. En caso negativo (excepciones), ejecuta el código bajo `except`.
 - C. Si no han ocurrido excepciones, ejecuta el código bajo `else`.
 - D. En cualquier caso, ejecuta el código bajo `finally`.



The screenshot shows a Jupyter Notebook interface with three tabs: `estructuras_02.ipynb`, `estructuras_01.ipynb`, and `estructuras_03.ipynb`. The active tab is `estructuras_03.ipynb`. The notebook title is "Estructura try - except". The code cell [16] contains the following Python code:

```
[16]: try:
    print('****Código bajo la instrucción try****\n')
    numero = int(input('Introduce un número par: '))
    assert numero%2 == 0
except AssertionError:
    print('\n****Código bajo la instrucción except****\n')
    print('El número que has introducido no es par.')
else:
    print('\n****Código bajo la instrucción else****\n')
    print(f'Efectivamente, {numero} es un número par.')
finally:
    print('\n****Código bajo la instrucción finally****\n')
    print('Game over.')
```

The output of the code is:

```
****Código bajo la instrucción try****

Introduce un número par: 6

****Código bajo la instrucción else****

Efectivamente, 6 es un número par.

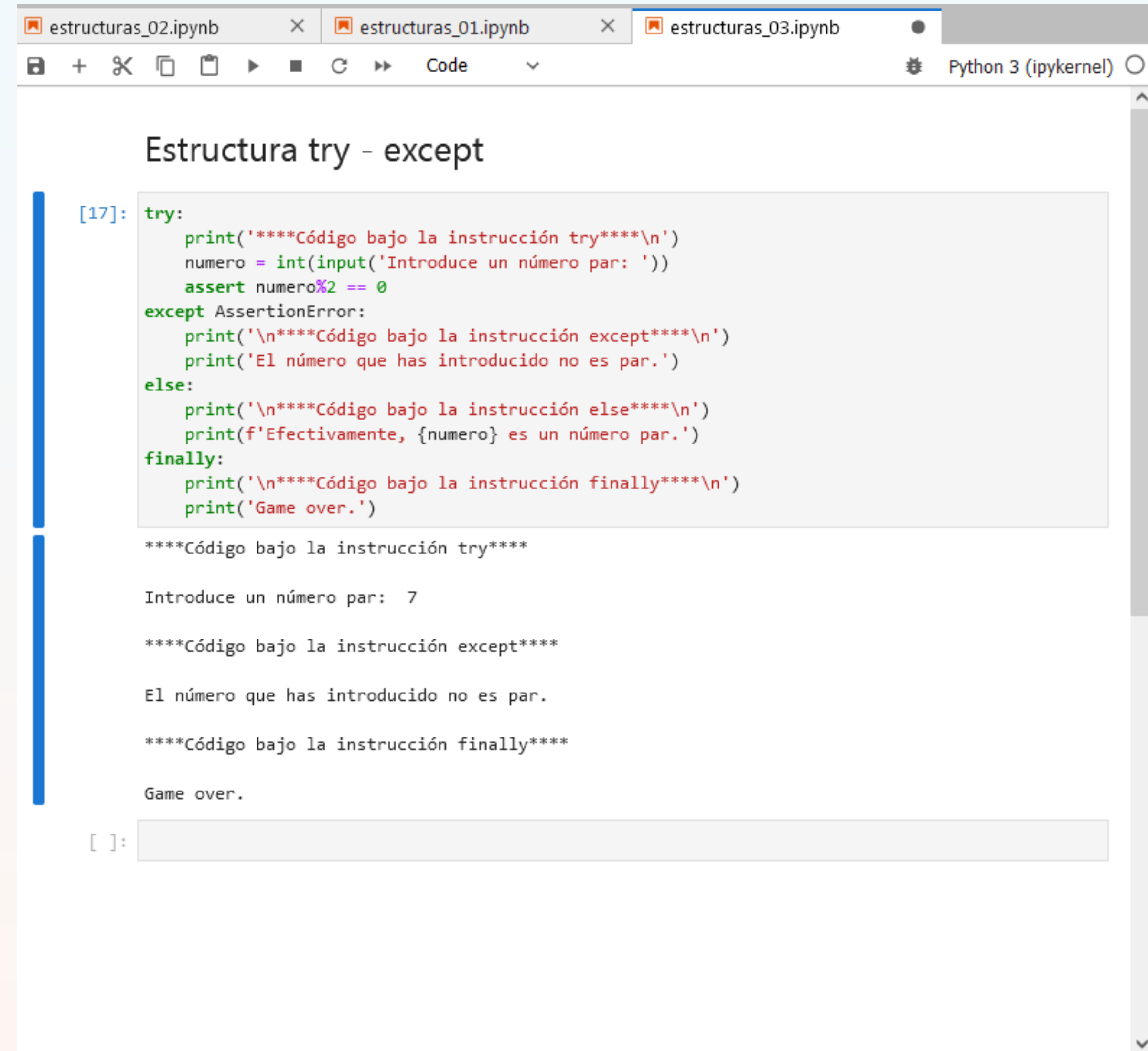
****Código bajo la instrucción finally****

Game over.
```


Objetos y Estructuras Integradas

Principales Estructuras Integradas en la Standard Python Library

- A. Bucle **for**: recorre cualquier objeto que tenga definido el método `'__iter__'`.
- B. Bucle **while**: repite la ejecución de cualquier código mientras se cumpla la condición especificada.
- C. Condicional **if...elif...else**: ejecuta el código enmarcado bajo las respectivas instrucciones `if`, `elif` o `else`.
 - A. Instrucción `continue`: pasa a la siguiente iteración.
 - B. Instrucción `break`: finaliza el bucle.
- D. Estructura **try...except...else...finally**:
 - A. Intenta ejecutar el código bajo la instrucción `try`.
 - B. En caso negativo (excepciones), ejecuta el código bajo `except`.
 - C. Si no han ocurrido excepciones, ejecuta el código bajo `else`.
 - D. En cualquier caso, ejecuta el código bajo `finally`.



The screenshot shows a Jupyter Notebook interface with three tabs: `estructuras_02.ipynb`, `estructuras_01.ipynb`, and `estructuras_03.ipynb`. The active tab is `estructuras_03.ipynb`. The notebook title is "Estructura try - except". The code cell [17]: contains the following Python code:

```
try:
    print('****Código bajo la instrucción try****\n')
    numero = int(input('Introduce un número par: '))
    assert numero%2 == 0
except AssertionError:
    print('\n****Código bajo la instrucción except****\n')
    print('El número que has introducido no es par.')
else:
    print('\n****Código bajo la instrucción else****\n')
    print(f'Efectivamente, {numero} es un número par.')
finally:
    print('\n****Código bajo la instrucción finally****\n')
    print('Game over.')
```

The output of the code is shown below the code cell:

```
****Código bajo la instrucción try****

Introduce un número par: 7

****Código bajo la instrucción except****

El número que has introducido no es par.

****Código bajo la instrucción finally****

Game over.
```

The code cell []: is empty.

Bases de la Programación en Python

3. Funciones y Decoradores

Funciones y Decoradores

Funciones integradas

No es preciso importar ningún módulo ni paquete para utilizarlas.

Están diseñadas para múltiples tipos de objetos, por eso no están definidas como **métodos** en una **clase**.

Algunas, por ejemplo **filter** y **map**, son muy potentes y rápidas.

A abs() aiter() all() any() anext() ascii()	E enumerate() eval() exec()	L len() list() locals()	R range() repr() reversed() round()
B bin() bool() breakpoint() bytearray() bytes()	F filter() float() format() frozenset()	M map() max() memoryview() min()	S set() setattr() slice() sorted() staticmethod() str() sum() super()
C callable() chr() classmethod() compile() complex()	G getattr() globals()	N next()	T tuple() type()
D delattr() dict() dir() divmod()	H hasattr() hash() help() hex()	O object() oct() open() ord()	V vars()
	I id() input() int() isinstance() issubclass() iter()	P pow() print() property()	Z zip() __import__()

Funciones integradas en **Python**.

Función **filter**.

Funciones y Decoradores

Funciones integradas

No es preciso importar ningún módulo ni paquete para utilizarlas.

Están diseñadas para múltiples tipos de objetos, por eso no están definidas como **métodos** en una **clase**.

Algunas, por ejemplo **filter** y **map**, son muy potentes y rápidas.

A abs() aiter() all() any() anext() ascii()	E enumerate() eval() exec()	L len() list() locals()	R range() repr() reversed() round()
B bin() bool() breakpoint() bytearray() bytes()	F filter() float() format() frozenset()	M map() max() memoryview() min()	S set() setattr() slice() sorted() staticmethod() str() sum() super()
C callable() chr() classmethod() compile() complex()	G getattr() globals()	N next()	T tuple() type()
D delattr() dict() dir() divmod()	H hasattr() hash() help() hex()	O object() oct() open() ord()	V vars()
	I id() input() int() isinstance() issubclass() iter()	P pow() print() property()	Z zip() __import__()

Funciones integradas en **Python**.

filter.ipynb

map.ipynb

+

✂

▶

■

↺

↻

▶▶

Markdown

⚙

...

Built-in function map

Map a data *range* using a **declared** function:

```
[6]: def square(x: float)->float:
      return x**2
```

```
[7]: data = range(0, 10)
```

```
[8]: list(map(square, data))
```

```
[8]: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Map a data *range* using an **anonymous** function:

```
[9]: list(map(lambda x: x**2, range(0, 10)))
```

```
[9]: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Map a data *range* using a **comprehension**:

```
[5]: [x**2 for x in range(0, 10)]
```

```
[5]: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Función **map**.

Funciones y Decoradores

Funciones

Las funciones son unas de las principales estructuras del lenguaje Python.

```
def nombre_de_la_funcion(*args, **kwargs): # Firma  
    ....  
    ....  
    ....[código] # Código estructurado (try - except, etc.)  
    ....  
    ....return objeto1, objeto2, ... # Objetos de retorno (si hay)
```

Junto con las definiciones de clase, constituyen una gran parte del contenido de los módulos importados.

Si una función está definida dentro de una **clase**, se denomina **método**. Un método solo es aplicable a objetos de la clase donde se define.

The screenshot shows a Jupyter Notebook interface with a tab titled 'functions_1.ipynb'. The notebook content is titled 'Function design' and includes the following text and code:

Let's check some tips when it comes to type a function:

```
[123]: def what_my_function1_performs(arg): # remember the "duck typing" principle...  
        """My function's docstring""" # docstring (not mandatory ... but almost) opti  
        return 'Argument is: ' + str(arg) # this is just a trivial example of the 're  
        # ... and naturally, even the 'return' statement itself is not mandatory ...
```

```
[124]: what_my_function1_performs(5)  
[124]: 'Argument is: 5'  
[125]: what_my_function1_performs.__doc__  
[125]: "My function's docstring"
```

If we define more than one positional argument:

```
[126]: def what_my_function2_performs(arg1, arg2):  
        """My function's docstring"""  
        return arg1 + arg2
```

When a positional argument is neglected, the function call raises a 'TypeError':

```
[127]: what_my_function2_performs(5)
```

The output for [127] shows a **TypeError** traceback:

```
-----  
TypeError                                Traceback (most recent call last)  
/var/folders/1k/jw4tpsp92d9_n1n1y2h52wfh0000gn/T/ipykernel_25339/3560065051.py in  
<module>  
----> 1 what_my_function2_performs(5)  
  
TypeError: what_my_function2_performs() missing 1 required positional argument: '  
arg2'
```


Funciones y Decoradores

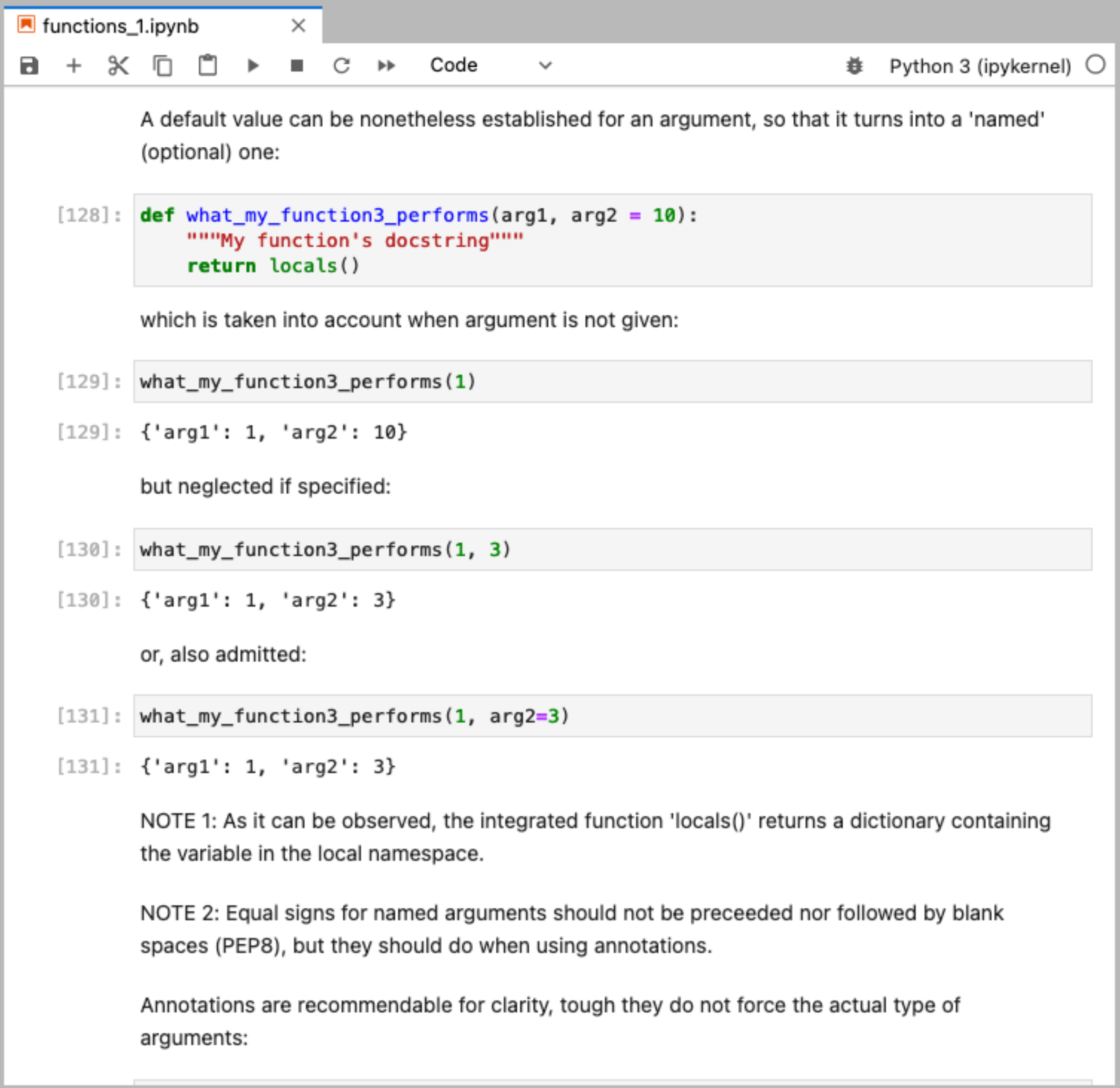
Funciones

Las funciones son unas de las principales estructuras del lenguaje Python.

```
def nombre_de_la_funcion(*args, **kwargs): # Firma  
..."""Descripción de la función""" # doctring  
...[código] # Código estructurado (try - except, etc.)  
...return objeto1, objeto2, ... # Objetos de retorno (si hay)
```

Junto con las definiciones de clase, constituyen una gran parte del contenido de los módulos importados.

Si una función está definida dentro de una **clase**, se denomina **método**. Un método solo es aplicable a objetos de la clase donde se define.



```
functions_1.ipynb x  
+ ✂ 📄 ▶ ■ ↺ ⏩ Code ▾ Python 3 (ipykernel) ○  
  
A default value can be nonetheless established for an argument, so that it turns into a 'named'  
(optional) one:  
  
[128]: def what_my_function3_performs(arg1, arg2 = 10):  
        """My function's docstring"""  
        return locals()  
  
which is taken into account when argument is not given:  
  
[129]: what_my_function3_performs(1)  
[129]: {'arg1': 1, 'arg2': 10}  
  
but neglected if specified:  
  
[130]: what_my_function3_performs(1, 3)  
[130]: {'arg1': 1, 'arg2': 3}  
  
or, also admitted:  
  
[131]: what_my_function3_performs(1, arg2=3)  
[131]: {'arg1': 1, 'arg2': 3}  
  
NOTE 1: As it can be observed, the integrated function 'locals()' returns a dictionary containing  
the variable in the local namespace.  
  
NOTE 2: Equal signs for named arguments should not be preceeded nor followed by blank  
spaces (PEP8), but they should do when using annotations.  
  
Annotations are recommendable for clarity, tough they do not force the actual type of  
arguments:
```


Funciones y Decoradores

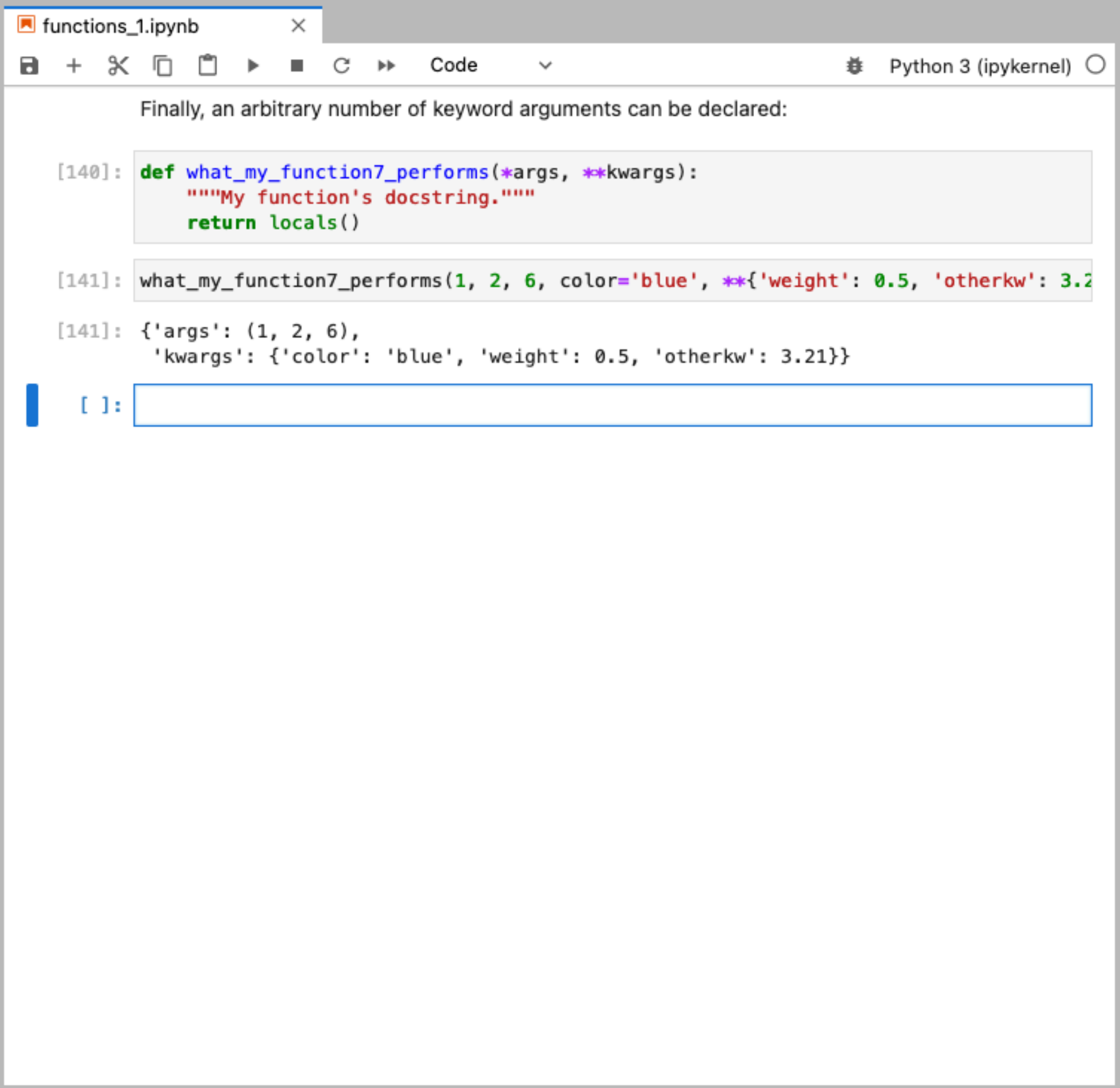
Funciones

Las funciones son unas de las principales estructuras del lenguaje Python.

```
def nombre_de_la_funcion(*args, **kwargs): # Firma  
    ....  
    ....  
    ....[código] # Código estructurado (try - except, etc.)  
    ....  
    ....return objeto1, objeto2, ... # Objetos de retorno (si hay)
```

Junto con las definiciones de clase, constituyen una gran parte del contenido de los módulos importados.

Si una función está definida dentro de una **clase**, se denomina **método**. Un método solo es aplicable a objetos de la clase donde se define.



The screenshot shows a Jupyter Notebook interface with a single cell. The cell contains the following Python code:

```
Finally, an arbitrary number of keyword arguments can be declared:  
  
[140]: def what_my_function7_performs(*args, **kwargs):  
        """My function's docstring."""  
        return locals()  
  
[141]: what_my_function7_performs(1, 2, 6, color='blue', **{'weight': 0.5, 'otherkw': 3.21})  
  
[141]: {'args': (1, 2, 6),  
        'kwargs': {'color': 'blue', 'weight': 0.5, 'otherkw': 3.21}}
```

The output of the code is displayed below the cell, showing the dictionary returned by the function call.

Funciones y Decoradores

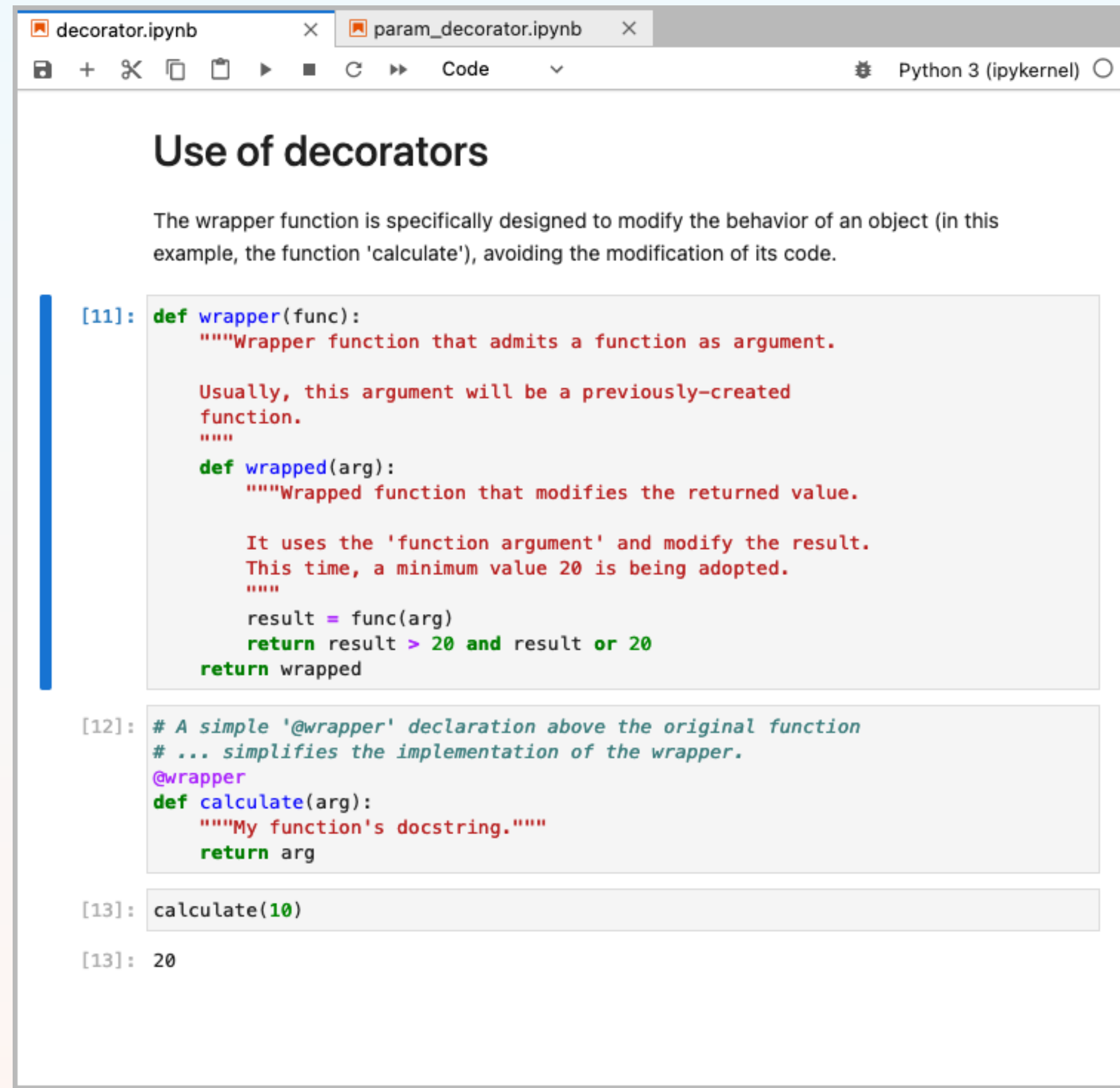
Herramientas avanzadas: decoradores

Son funciones construidas para modificar el comportamiento de otros objetos, usualmente otras funciones, métodos o clases.

Son herramientas muy potentes, pues pueden diseñarse y aplicarse sobre cualquier objeto previamente construido.

El lenguaje python incluye una herramienta (la declaración '@decorator') para la implementación simple de este concepto.

Se considera un recurso fundamental para dotar al trabajo realizado de **extensibilidad** y **versatilidad**, sin que éste pierda generalidad, y eludiendo la necesidad de replicar el código (p.ej. funciones de librerías externas) para introducir modificaciones.



```
decorator.ipynb x param_decorator.ipynb x
Python 3 (ipykernel)

Use of decorators

The wrapper function is specifically designed to modify the behavior of an object (in this
example, the function 'calculate'), avoiding the modification of its code.

[11]: def wrapper(func):
        """Wrapper function that admits a function as argument.

        Usually, this argument will be a previously-created
        function.
        """
        def wrapped(arg):
            """Wrapped function that modifies the returned value.

            It uses the 'function argument' and modify the result.
            This time, a minimum value 20 is being adopted.
            """
            result = func(arg)
            return result > 20 and result or 20
        return wrapped

[12]: # A simple '@wrapper' declaration above the original function
# ... simplifies the implementation of the wrapper.
@wrapper
def calculate(arg):
    """My function's docstring."""
    return arg

[13]: calculate(10)

[13]: 20
```


Funciones y Decoradores

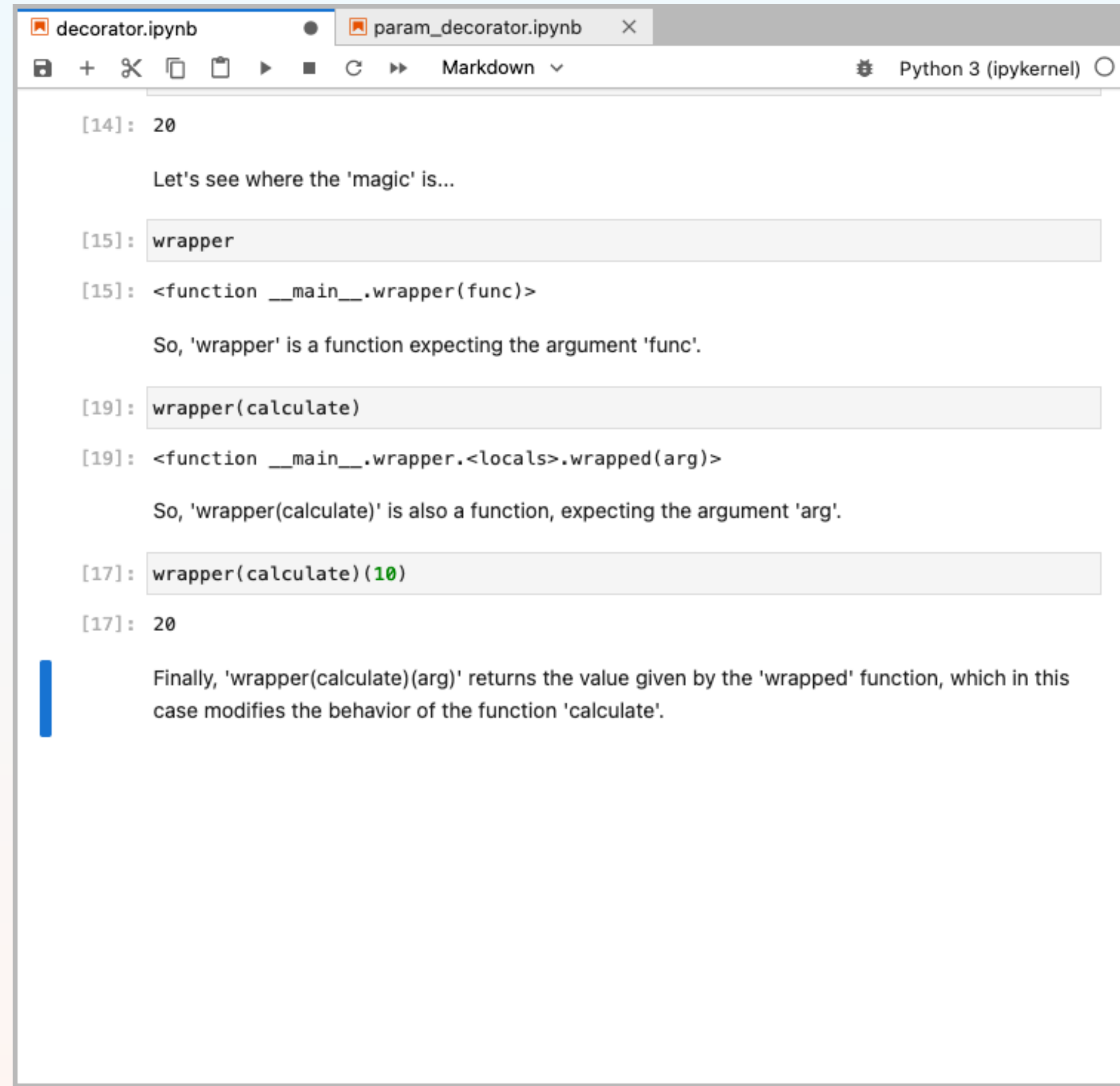
Herramientas avanzadas: decoradores

Son funciones construidas para modificar el comportamiento de otros objetos, usualmente otras funciones, métodos o clases.

Son herramientas muy potentes, pues pueden diseñarse y aplicarse sobre cualquier objeto previamente construido.

El lenguaje python incluye una herramienta (la declaración '@decorator') para la implementación simple de este concepto.

Se considera un recurso fundamental para dotar al trabajo realizado de **extensibilidad** y **versatilidad**, sin que éste pierda generalidad, y eludiendo la necesidad de replicar el código (p.ej. funciones de librerías externas) para introducir modificaciones.



```
decorator.ipynb  param_decorator.ipynb
+  ✂  📄  📌  ▶  ■  ↺  ⏩  Markdown  Python 3 (ipykernel)

[14]: 20

      Let's see where the 'magic' is...

[15]: wrapper
[15]: <function __main__.wrapper(func)>

      So, 'wrapper' is a function expecting the argument 'func'.

[19]: wrapper(calculate)
[19]: <function __main__.wrapper.<locals>.wrapped(arg)>

      So, 'wrapper(calculate)' is also a function, expecting the argument 'arg'.

[17]: wrapper(calculate)(10)
[17]: 20

      Finally, 'wrapper(calculate)(arg)' returns the value given by the 'wrapped' function, which in this
      case modifies the behavior of the function 'calculate'.
```

Funciones y Decoradores

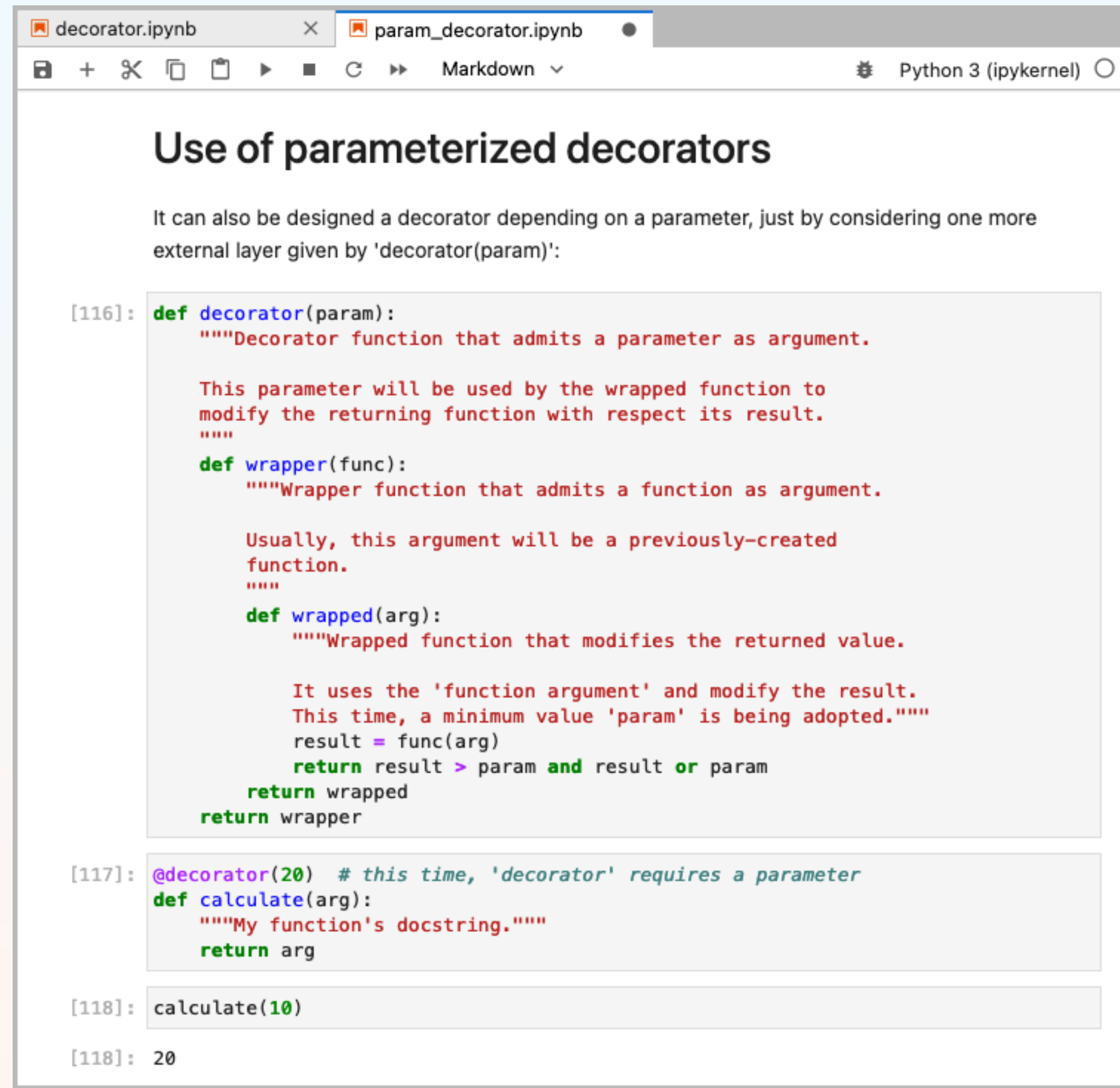
Herramientas avanzadas: decoradores

Son funciones construidas para modificar el comportamiento de otros objetos, usualmente otras funciones, métodos o clases.

Son herramientas muy potentes, pues pueden diseñarse y aplicarse sobre cualquier objeto previamente construido.

El lenguaje python incluye una herramienta (la declaración '@decorator') para la implementación simple de este concepto.

Se considera un recurso fundamental para dotar al trabajo realizado de **extensibilidad** y **versatilidad**, sin que éste pierda generalidad, y eludiendo la necesidad de replicar el código (p.ej. funciones de librerías externas) para introducir modificaciones.



```
decorator.ipynb x param_decorator.ipynb
+ ✂ 📄 📌 ▶ ■ ↺ ⏩ Markdown Python 3 (ipykernel)

Use of parameterized decorators

It can also be designed a decorator depending on a parameter, just by considering one more
external layer given by 'decorator(param)':

[116]: def decorator(param):
        """Decorator function that admits a parameter as argument.

        This parameter will be used by the wrapped function to
        modify the returning function with respect its result.
        """
        def wrapper(func):
            """Wrapper function that admits a function as argument.

            Usually, this argument will be a previously-created
            function.
            """
            def wrapped(arg):
                """Wrapped function that modifies the returned value.

                It uses the 'function argument' and modify the result.
                This time, a minimum value 'param' is being adopted."""
                result = func(arg)
                return result > param and result or param
            return wrapped
        return wrapper

[117]: @decorator(20) # this time, 'decorator' requires a parameter
def calculate(arg):
    """My function's docstring."""
    return arg

[118]: calculate(10)

[118]: 20
```

Funciones y Decoradores

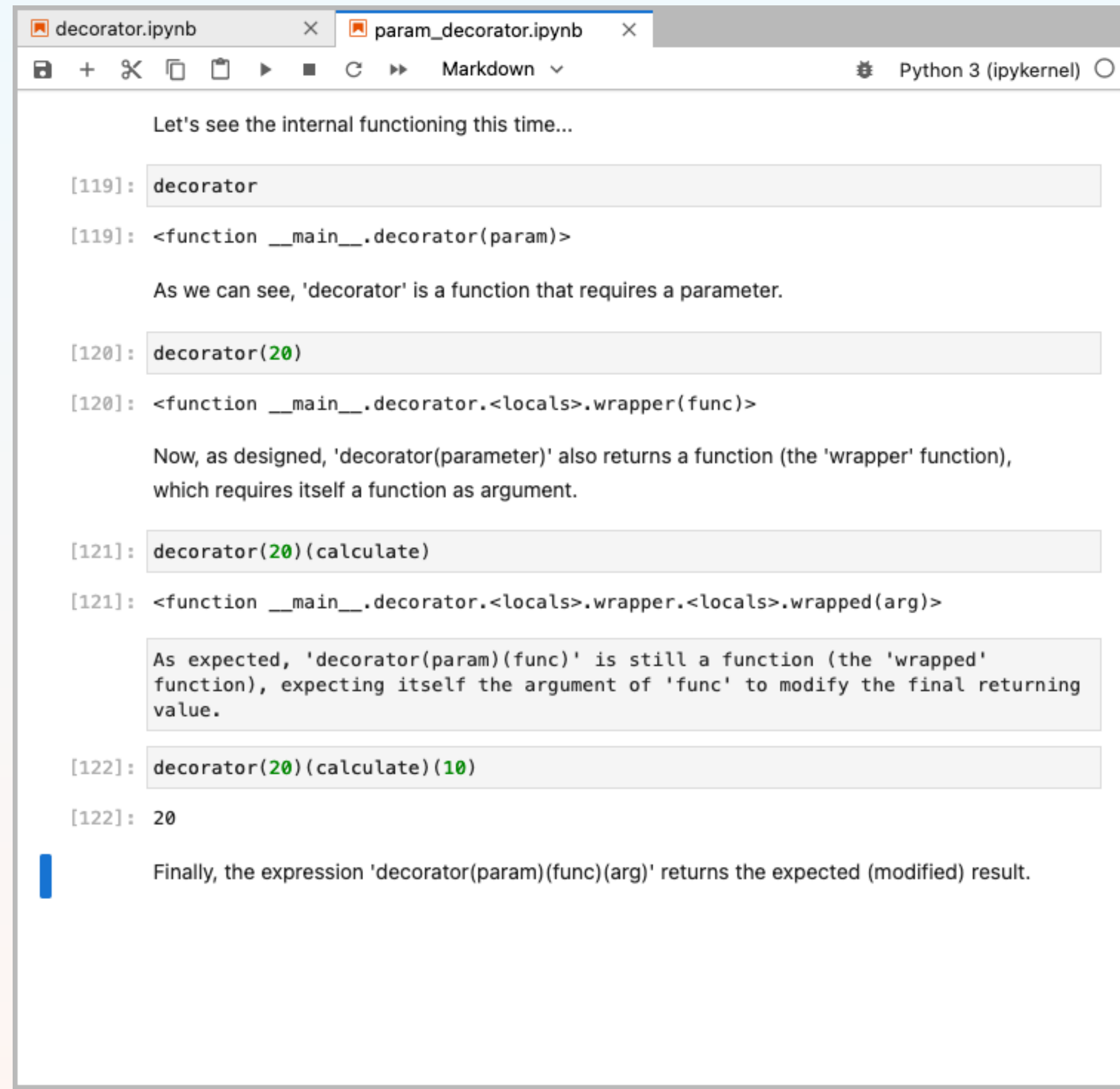
Herramientas avanzadas: decoradores

Son funciones construidas para modificar el comportamiento de otros objetos, usualmente otras funciones, métodos o clases.

Son herramientas muy potentes, pues pueden diseñarse y aplicarse sobre cualquier objeto previamente construido.

El lenguaje python incluye una herramienta (la declaración '@decorator') para la implementación simple de este concepto.

Se considera un recurso fundamental para dotar al trabajo realizado de **extensibilidad** y **versatilidad**, sin que éste pierda generalidad, y eludiendo la necesidad de replicar el código (p.ej. funciones de librerías externas) para introducir modificaciones.



```
decorator.ipynb x param_decorator.ipynb x
+ ✂ 📄 📌 ▶ ■ ↺ ⏩ Markdown v Python 3 (ipykernel) ○

Let's see the internal functioning this time...

[119]: decorator
[119]: <function __main__.decorator(param)>

As we can see, 'decorator' is a function that requires a parameter.

[120]: decorator(20)
[120]: <function __main__.decorator.<locals>.wrapper(func)>

Now, as designed, 'decorator(parameter)' also returns a function (the 'wrapper' function),
which requires itself a function as argument.

[121]: decorator(20)(calculate)
[121]: <function __main__.decorator.<locals>.wrapper.<locals>.wrapped(arg)>

As expected, 'decorator(param)(func)' is still a function (the 'wrapped'
function), expecting itself the argument of 'func' to modify the final returning
value.

[122]: decorator(20)(calculate)(10)
[122]: 20

Finally, the expression 'decorator(param)(func)(arg)' returns the expected (modified) result.
```


Bases de la Programación en Python

4. Programación Orientada a Objetos

Bases de la Programación en Python

Bases de Programación en Python

Declaración de clases

Se dice que en *Python* todo es un objeto, porque cualquier variable pertenece a una clase determinada.

class **Nombre_de_la_clase**(clase o clases de las que hereda):

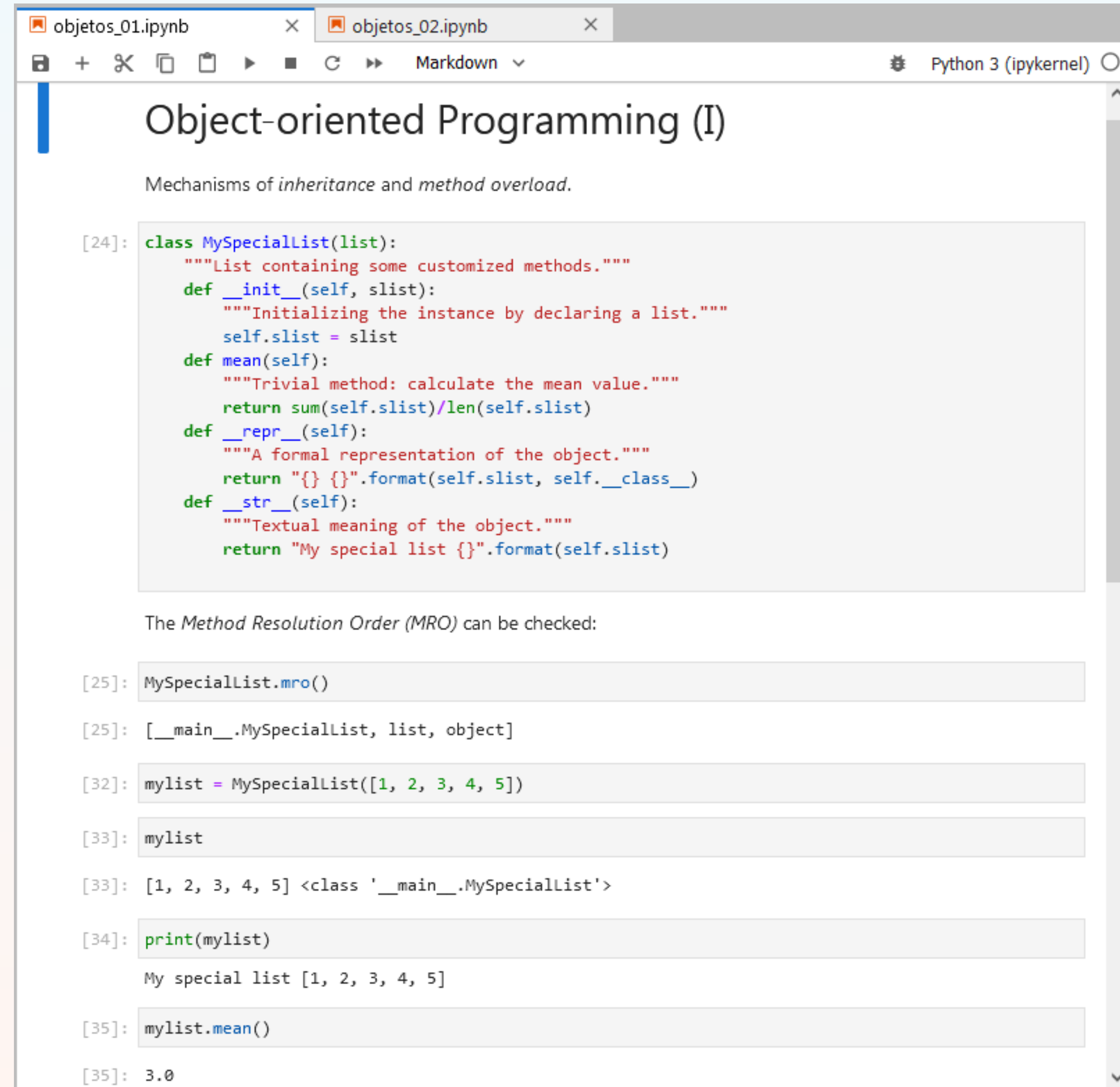
....**"""Documentación de la clase"""** # Descripción completa

....Atributos de clase # Por ejemplo: `color = 'black'`

....Métodos de clase # Por ejemplo: `__init__(args iniciales)`

Una vez declarada una clase, se pueden declarar instancias de dicho tipo de objetos, es decir, de dicha clase.

Ejemplo: `mipiranometro = Piranómetro(numero de serie)`



```
objetos_01.ipynb  objetos_02.ipynb
Python 3 (ipykernel)

Object-oriented Programming (I)

Mechanisms of inheritance and method overload.

[24]: class MySpecialList(list):
      """List containing some customized methods."""
      def __init__(self, slist):
          """Initializing the instance by declaring a list."""
          self.slist = slist
      def mean(self):
          """Trivial method: calculate the mean value."""
          return sum(self.slist)/len(self.slist)
      def __repr__(self):
          """A formal representation of the object."""
          return "{} {}".format(self.slist, self.__class__)
      def __str__(self):
          """Textual meaning of the object."""
          return "My special list {}".format(self.slist)

The Method Resolution Order (MRO) can be checked:

[25]: MySpecialList.mro()

[25]: [__main__.MySpecialList, list, object]

[32]: mylist = MySpecialList([1, 2, 3, 4, 5])

[33]: mylist

[33]: [1, 2, 3, 4, 5] <class '__main__.MySpecialList'>

[34]: print(mylist)

My special list [1, 2, 3, 4, 5]

[35]: mylist.mean()

[35]: 3.0
```

Bases de la Programación en Python

Bases de Programación en Python

Declaración de clases

Se dice que en *Python* todo es un objeto, porque cualquier variable pertenece a una clase determinada.

class **Nombre_de_la_clase**(clase o clases de las que hereda):

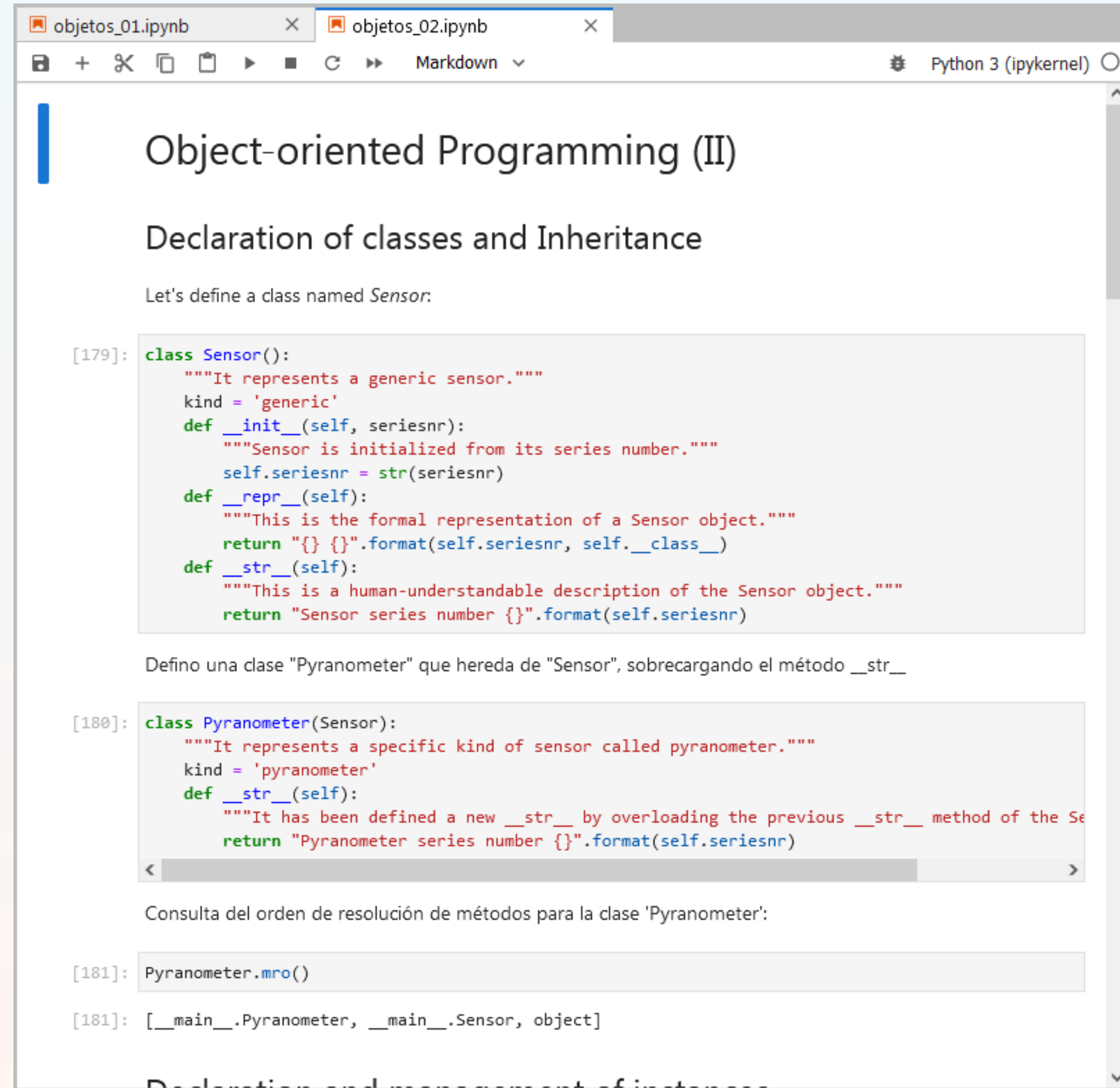
....**"""Documentación de la clase"""** # Descripción completa

....Atributos de clase # Por ejemplo: `color = 'black'`

....Métodos de clase # Por ejemplo: `__init__(args iniciales)`

Una vez declarada una clase, se pueden declarar instancias de dicho tipo de objetos, es decir, de dicha clase.

Ejemplo: `mipiranometro = Piranómetro(numero de serie)`



The screenshot shows a Jupyter Notebook with two tabs: 'objetos_01.ipynb' and 'objetos_02.ipynb'. The active tab is 'objetos_02.ipynb', which contains the following content:

Object-oriented Programming (II)

Declaration of classes and Inheritance

Let's define a class named *Sensor*:

```
[179]: class Sensor():
        """It represents a generic sensor."""
        kind = 'generic'
        def __init__(self, seriesnr):
            """Sensor is initialized from its series number."""
            self.seriesnr = str(seriesnr)
        def __repr__(self):
            """This is the formal representation of a Sensor object."""
            return "{} {}".format(self.seriesnr, self.__class__)
        def __str__(self):
            """This is a human-understandable description of the Sensor object."""
            return "Sensor series number {}".format(self.seriesnr)
```

Defino una clase "Pyranometer" que hereda de "Sensor", sobrecargando el método `__str__`

```
[180]: class Pyranometer(Sensor):
        """It represents a specific kind of sensor called pyranometer."""
        kind = 'pyranometer'
        def __str__(self):
            """It has been defined a new __str__ by overloading the previous __str__ method of the Se
            return "Pyranometer series number {}".format(self.seriesnr)
```

Consulta del orden de resolución de métodos para la clase 'Pyranometer':

```
[181]: Pyranometer.mro()
[181]: [__main__.Pyranometer, __main__.Sensor, object]
```


Bases de la Programación en Python

Bases de Programación en Python

Declaración de clases

Se dice que en *Python* todo es un objeto, porque cualquier variable pertenece a una clase determinada.

class Nombre_de_la_clase(clase o clases de las que hereda):

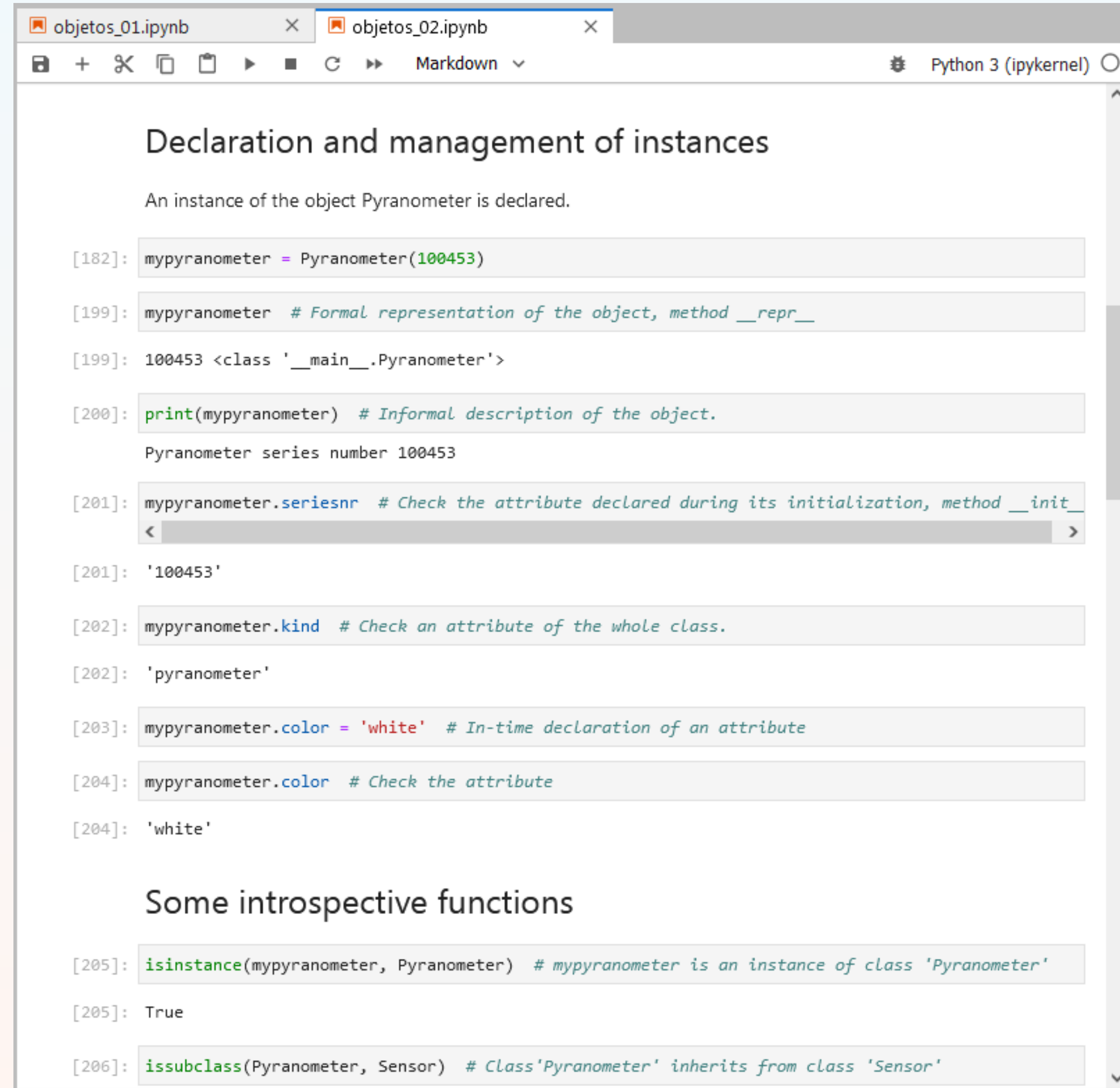
....**"""Documentación de la clase"""** *# Descripción completa*

....Atributos de clase *# Por ejemplo: color = 'black'*

....Métodos de clase *# Por ejemplo: __init__(args iniciales)*

Una vez declarada una clase, se pueden declarar instancias de dicho tipo de objetos, es decir, de dicha clase.

Ejemplo: mipiranometro = Piranómetro(numero de serie)



The screenshot shows a Jupyter Notebook with two tabs: 'objetos_01.ipynb' and 'objetos_02.ipynb'. The active tab is 'objetos_02.ipynb', which contains the following content:

Declaration and management of instances

An instance of the object Pyranometer is declared.

```
[182]: mypyranometer = Pyranometer(100453)
```

```
[199]: mypyranometer # Formal representation of the object, method __repr__
```

```
[199]: 100453 <class '__main__.Pyranometer'>
```

```
[200]: print(mypyranometer) # Informal description of the object.
```

```
Pyranometer series number 100453
```

```
[201]: mypyranometer.seriesnr # Check the attribute declared during its initialization, method __init__
```

```
[201]: '100453'
```

```
[202]: mypyranometer.kind # Check an attribute of the whole class.
```

```
[202]: 'pyranometer'
```

```
[203]: mypyranometer.color = 'white' # In-time declaration of an attribute
```

```
[204]: mypyranometer.color # Check the attribute
```

```
[204]: 'white'
```

Some introspective functions

```
[205]: isinstance(mypyranometer, Pyranometer) # mypyranometer is an instance of class 'Pyranometer'
```

```
[205]: True
```

```
[206]: issubclass(Pyranometer, Sensor) # Class 'Pyranometer' inherits from class 'Sensor'
```

Bases de la Programación en Python

Bases de Programación en Python

Declaración de clases

Se dice que en *Python* todo es un objeto, porque cualquier variable pertenece a una clase determinada.

class Nombre_de_la_clase(clase o clases de las que hereda):

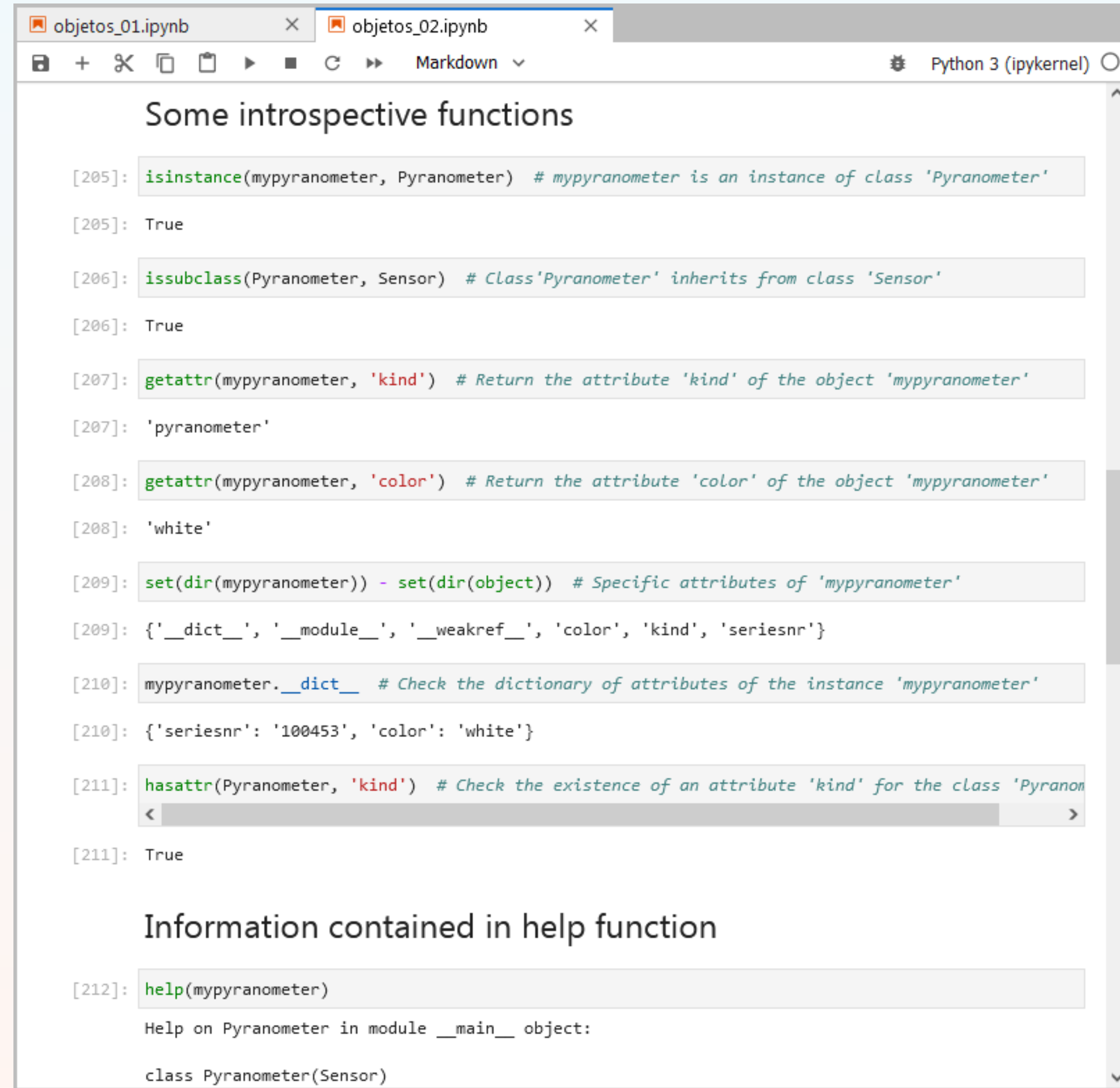
....**"""Documentación de la clase"""** *# Descripción completa*

....Atributos de clase *# Por ejemplo: color = 'black'*

....Métodos de clase *# Por ejemplo: __init__(args iniciales)*

Una vez declarada una clase, se pueden declarar instancias de dicho tipo de objetos, es decir, de dicha clase.

Ejemplo: mipiranometro = Piranómetro(numero de serie)



The screenshot shows a Jupyter Notebook with two tabs: 'objetos_01.ipynb' and 'objetos_02.ipynb'. The active tab is 'objetos_02.ipynb', which contains a section titled 'Some introspective functions'. The notebook displays several code cells with their outputs:

- Cell [205]: `isinstance(mypyranometer, Pyranometer)` with output `True`. Comment: *# mypyranometer is an instance of class 'Pyranometer'*
- Cell [206]: `issubclass(Pyranometer, Sensor)` with output `True`. Comment: *# Class 'Pyranometer' inherits from class 'Sensor'*
- Cell [207]: `getattr(mypyranometer, 'kind')` with output `'pyranometer'`. Comment: *# Return the attribute 'kind' of the object 'mypyranometer'*
- Cell [208]: `getattr(mypyranometer, 'color')` with output `'white'`. Comment: *# Return the attribute 'color' of the object 'mypyranometer'*
- Cell [209]: `set(dir(mypyranometer)) - set(dir(object))` with output `{'__dict__', '__module__', '__weakref__', 'color', 'kind', 'seriesnr'}`. Comment: *# Specific attributes of 'mypyranometer'*
- Cell [210]: `mypyranometer.__dict__` with output `{'seriesnr': '100453', 'color': 'white'}`. Comment: *# Check the dictionary of attributes of the instance 'mypyranometer'*
- Cell [211]: `hasattr(Pyranometer, 'kind')` with output `True`. Comment: *# Check the existence of an attribute 'kind' for the class 'Pyranometer'*

Below the code cells, there is a section titled 'Information contained in help function'. Cell [212] shows `help(mypyranometer)` with the output: `Help on Pyranometer in module __main__ object:`
`class Pyranometer(Sensor)`

Bases de la Programación en Python

Bases de Programación en Python

Declaración de clases

Se dice que en *Python* todo es un objeto, porque cualquier variable pertenece a una clase determinada.

class **Nombre_de_la_clase**(clase o clases de las que hereda):

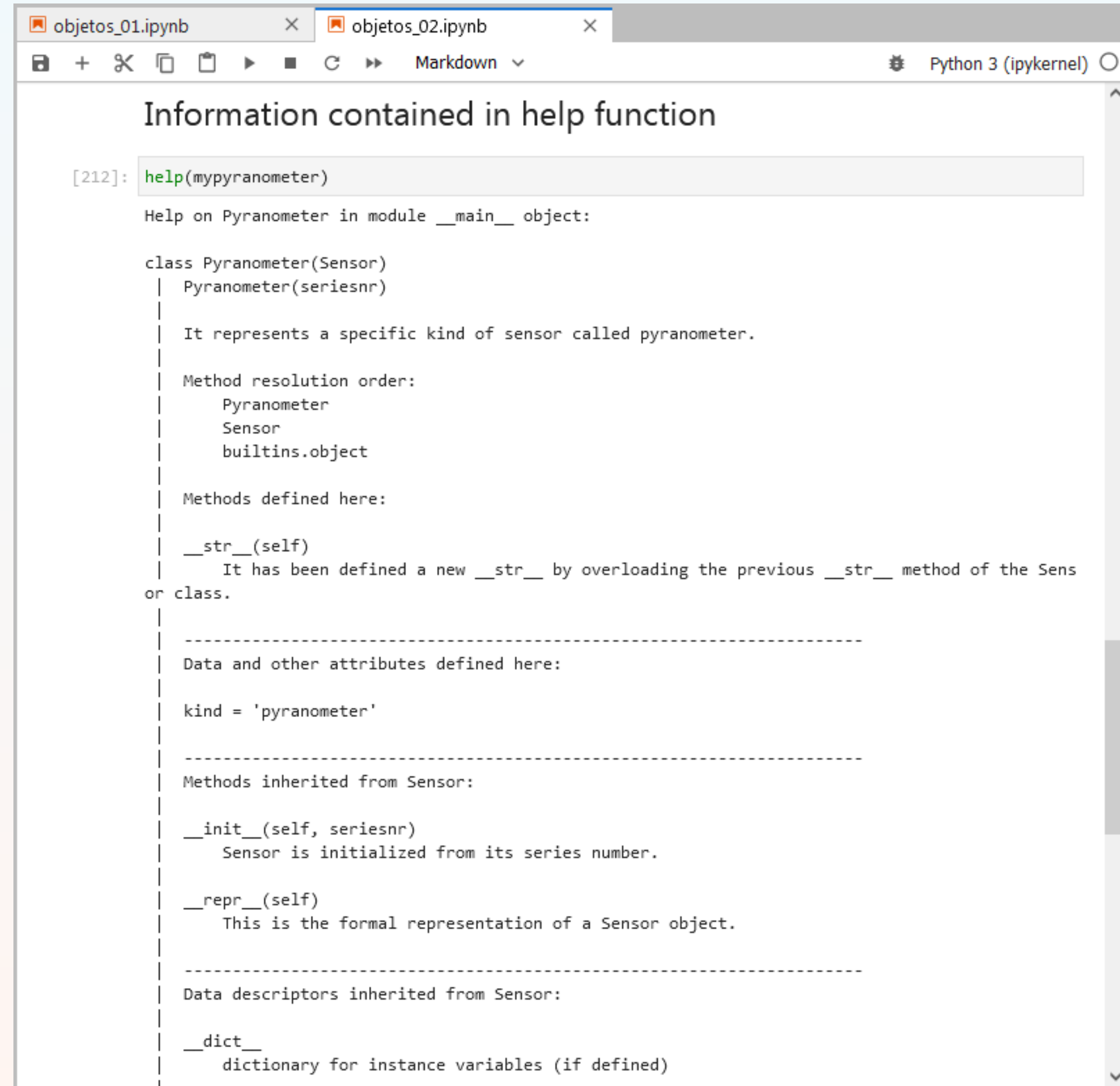
....**"""Documentación de la clase"""** *# Descripción completa*

....Atributos de clase *# Por ejemplo: color = 'black'*

....Métodos de clase *# Por ejemplo: __init__(args iniciales)*

Una vez declarada una clase, se pueden declarar instancias de dicho tipo de objetos, es decir, de dicha clase.

Ejemplo: mipiranometro = Piranómetro(numero de serie)



The screenshot shows a Jupyter Notebook interface with two tabs: 'objetos_01.ipynb' and 'objetos_02.ipynb'. The active tab is 'objetos_02.ipynb'. The notebook displays the output of the command `help(mypyranometer)`. The output is titled 'Information contained in help function' and shows the following information:

```
[212]: help(mypyranometer)

Help on Pyranometer in module __main__ object:

class Pyranometer(Sensor)
|   Pyranometer(seriesnr)
|
|   It represents a specific kind of sensor called pyranometer.
|
|   Method resolution order:
|       Pyranometer
|       Sensor
|       builtins.object
|
|   Methods defined here:
|
|   __str__(self)
|       It has been defined a new __str__ by overloading the previous __str__ method of the Sens
or class.
|
|   -----
|   Data and other attributes defined here:
|
|   kind = 'pyranometer'
|
|   -----
|   Methods inherited from Sensor:
|
|   __init__(self, seriesnr)
|       Sensor is initialized from its series number.
|
|   __repr__(self)
|       This is the formal representation of a Sensor object.
|
|   -----
|   Data descriptors inherited from Sensor:
|
|   __dict__
|       dictionary for instance variables (if defined)
```


Bases de la Programación en Python

5. Documentación del Código

Bases de la Programación en Python

Bases de Programación en Python

Documentación del código con doctest

Se trata de documentar el código de modo que pueda ser interpretado con facilidad.

El módulo debe contener una descripción antes de las importaciones.

La descripción del paquete completo debe ir en su archivo '`__init__.py`'.

Las funciones deben contener descripción y ejemplos de uso ¡válidos!

Ejemplo:

```
def Nombre_de_la_funcion(*args, **kwargs):
```

```
.... """Descripción de la función
```

```
.... **Ejemplo 1: caso de uso 1**
```

```
....>>> Nombre_de_la_funcion(argumentos caso 1) # Llamada a la función
```

```
....[Resultado esperado]
```

```
    # Linea en blanco
```

```
.... **Ejemplo 2**: caso de uso 2
```

```
    # etc.
```

```
.....
```

```
File Edit View Selection Find Packages Help
channels.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  # License: BSD 3-clause
4  # Copyright 2019-2021, José P. Silva. All rights reserved.
5  # See archive 'LICENSE' for further information.
6  """Provide tools to facilitate the selection of relevant data."""
7
8  from typing import Iterable
9
10
11 def set_channels(numbers: Iterable[int],
12                 names: Iterable[str],
13                 nameafter: bool = True) -> Iterable[str]:
14     """
15     Generate a list of channel names from a set of numbers and a set of names.
16
17     It is designed to automate the selection of specific active
18     channels, e.g. within data frames containing a big number of data
19     columns.
20     Given a list of *n* numbers (`numbers`) and *m* names (`names`), it
21     generates a list of channel names in the form:
22
23     [
24     [**number_1**][**name_1**],
25     [**number_1**][**name_2**], ...,
26     [**number_1**][**name_m**]
```

Bases de la Programación en Python

Bases de Programación en Python

Documentación del código con doctest

Se trata de documentar el código de modo que pueda ser interpretado con facilidad.

El módulo debe contener una descripción antes de las importaciones.

La descripción del paquete completo debe ir en su archivo '`__init__.py`'.

Las funciones deben contener descripción y ejemplos de uso ¡válidos!

Ejemplo:

```
def Nombre_de_la_funcion(*args, **kwargs):
```

```
...."""Descripción de la función
```

```
....**Ejemplo 1: caso de uso 1**
```

```
....>>> Nombre_de_la_funcion(argumentos caso 1) # Llamada a la función
```

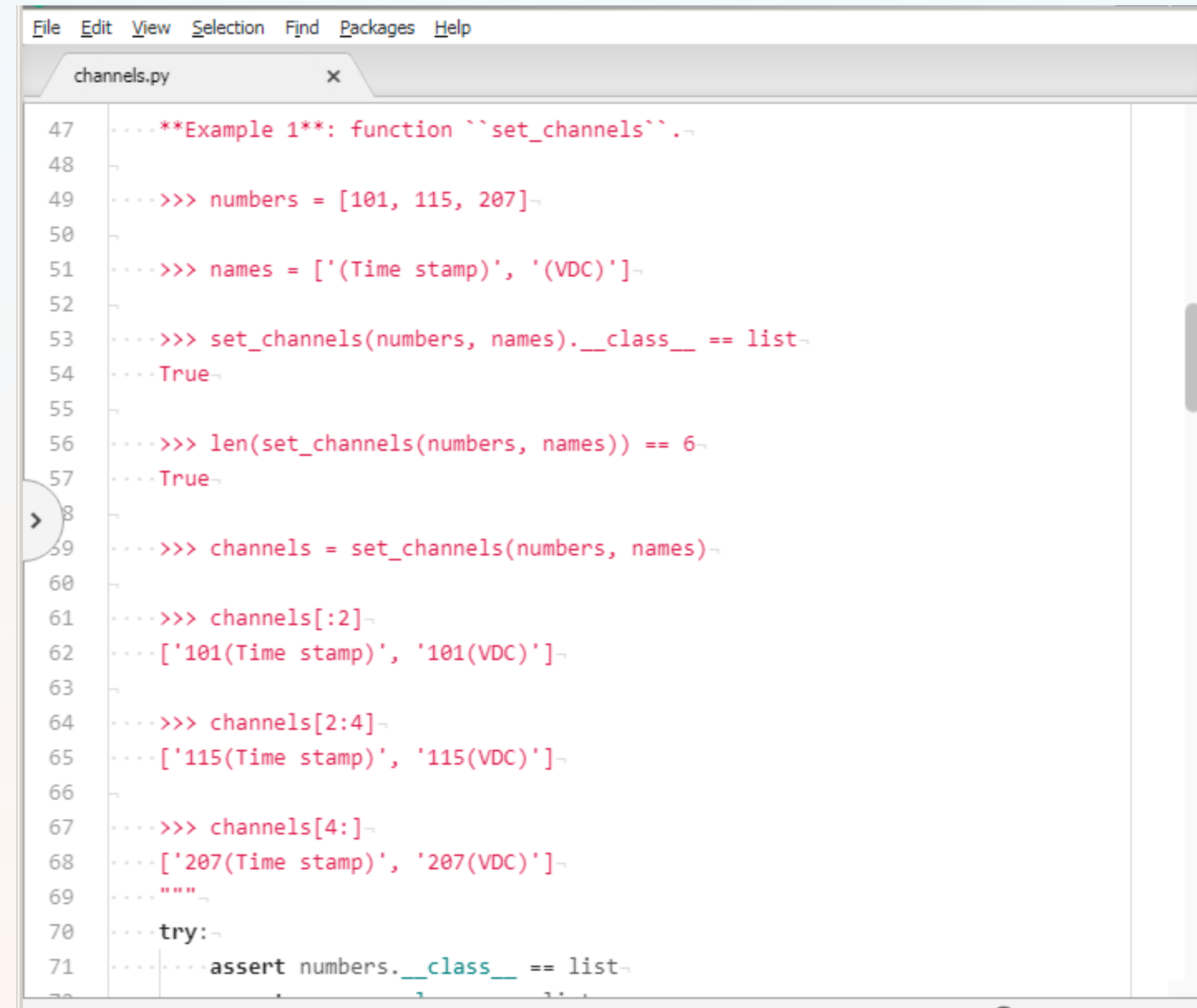
```
....[Resultado esperado]
```

```
    # Linea en blanco
```

```
....**Ejemplo 2**: caso de uso 2
```

```
    # etc.
```

```
.....
```



```
File Edit View Selection Find Packages Help
channels.py x
47 ....**Example 1**: function ``set_channels``.
48
49 ....>>> numbers = [101, 115, 207]
50
51 ....>>> names = ['(Time stamp)', '(VDC)']
52
53 ....>>> set_channels(numbers, names).__class__ == list
54 ....True
55
56 ....>>> len(set_channels(numbers, names)) == 6
57 ....True
58
59 ....>>> channels = set_channels(numbers, names)
60
61 ....>>> channels[:2]
62 ....['101(Time stamp)', '101(VDC)']
63
64 ....>>> channels[2:4]
65 ....['115(Time stamp)', '115(VDC)']
66
67 ....>>> channels[4:]
68 ....['207(Time stamp)', '207(VDC)']
69 ....
70 ....try:
71 ....    assert numbers.__class__ == list
```


Bases de la Programación en Python

Bases de Programación en Python

Documentación del código con doctest

Se trata de documentar el código de modo que pueda ser interpretado con facilidad.

El módulo debe contener una descripción antes de las importaciones.

La descripción del paquete completo debe ir en su archivo '`__init__.py`'.

Las funciones deben contener descripción y ejemplos de uso ¡válidos!

Ejemplo:

```
def Nombre_de_la_funcion(*args, **kwargs):
```

```
...."""Descripción de la función
```

```
....**Ejemplo 1: caso de uso 1**
```

```
....>>> Nombre_de_la_funcion(argumentos caso 1) # Llamada a la función
```

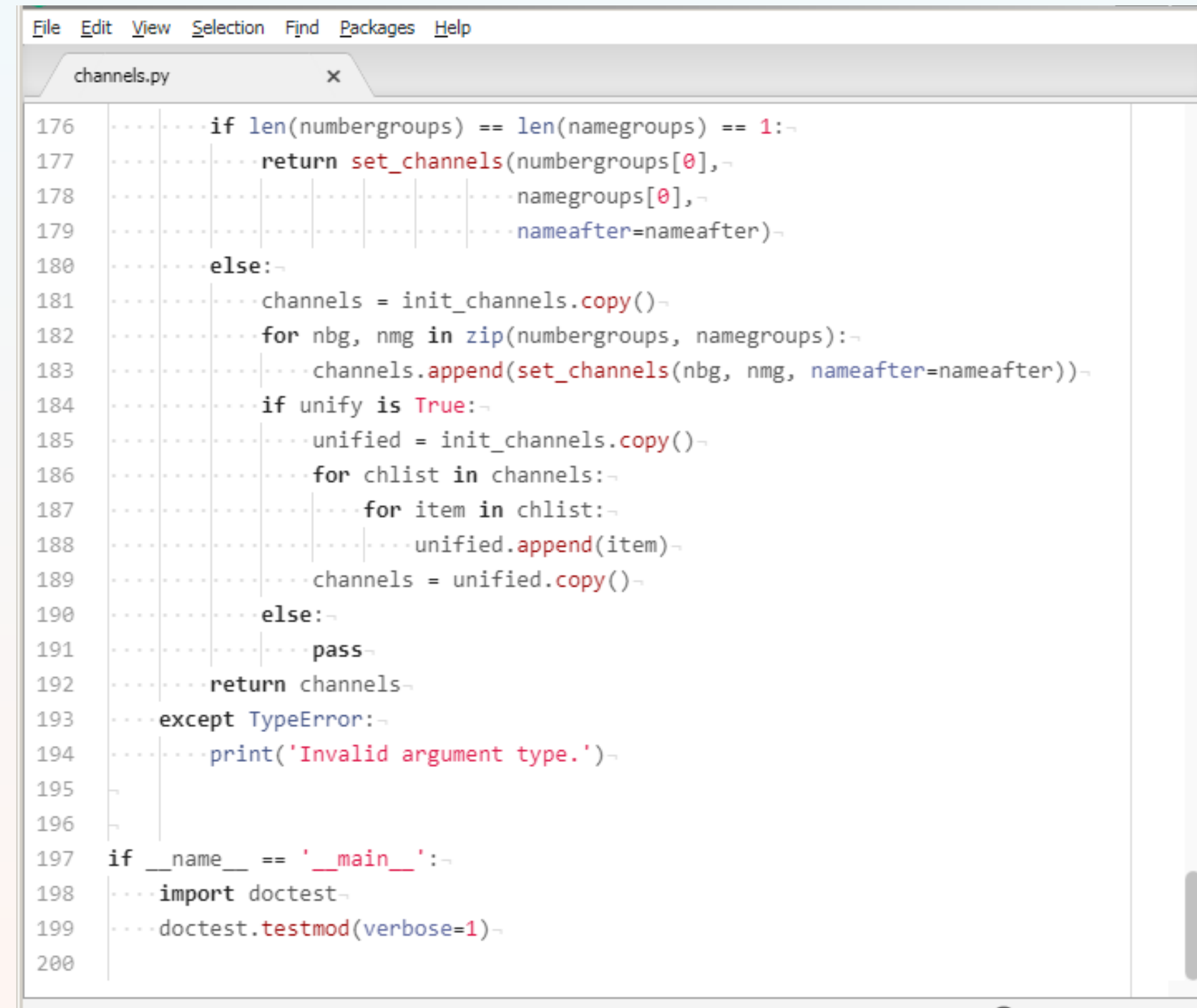
```
....[Resultado esperado]
```

```
....# Linea en blanco
```

```
....**Ejemplo 2**: caso de uso 2
```

```
....# etc.
```

```
...."""
```



```
File Edit View Selection Find Packages Help
channels.py x
176 .....if len(numbergroups) == len(namegroups) == 1:-
177 .....    return set_channels(numbergroups[0],-
178 .....                           namegroups[0],-
179 .....                           nameafter=nameafter)-
180 .....else:-
181 .....    channels = init_channels.copy()-
182 .....    for nb, nmg in zip(numbergroups, namegroups):-
183 .....        channels.append(set_channels(nb, nmg, nameafter=nameafter))-
184 .....    if unify is True:-
185 .....        unified = init_channels.copy()-
186 .....        for chlist in channels:-
187 .....            for item in chlist:-
188 .....                unified.append(item)-
189 .....        channels = unified.copy()-
190 .....    else:-
191 .....        pass-
192 .....    return channels-
193 .....except TypeError:-
194 .....    print('Invalid argument type.')->
195 .....
196 .....
197 if __name__ == '__main__':-
198 .....import doctest-
199 .....doctest.testmod(verbose=1)-
200 .....
```


Bases de la Programación en Python

Bases de Programación en Python

Documentación del código con doctest

Se trata de documentar el código de modo que pueda ser interpretado con facilidad.

El módulo debe contener una descripción antes de las importaciones.

La descripción del paquete completo debe ir en su archivo '`__init__.py`'.

Las funciones deben contener descripción y ejemplos de uso ¡válidos!

Ejemplo:

```
def Nombre_de_la_funcion(*args, **kwargs):
```

```
...."""Descripción de la función
```

```
....**Ejemplo 1: caso de uso 1**
```

```
....>>> Nombre_de_la_funcion(argumentos caso 1) # Llamada a la función
```

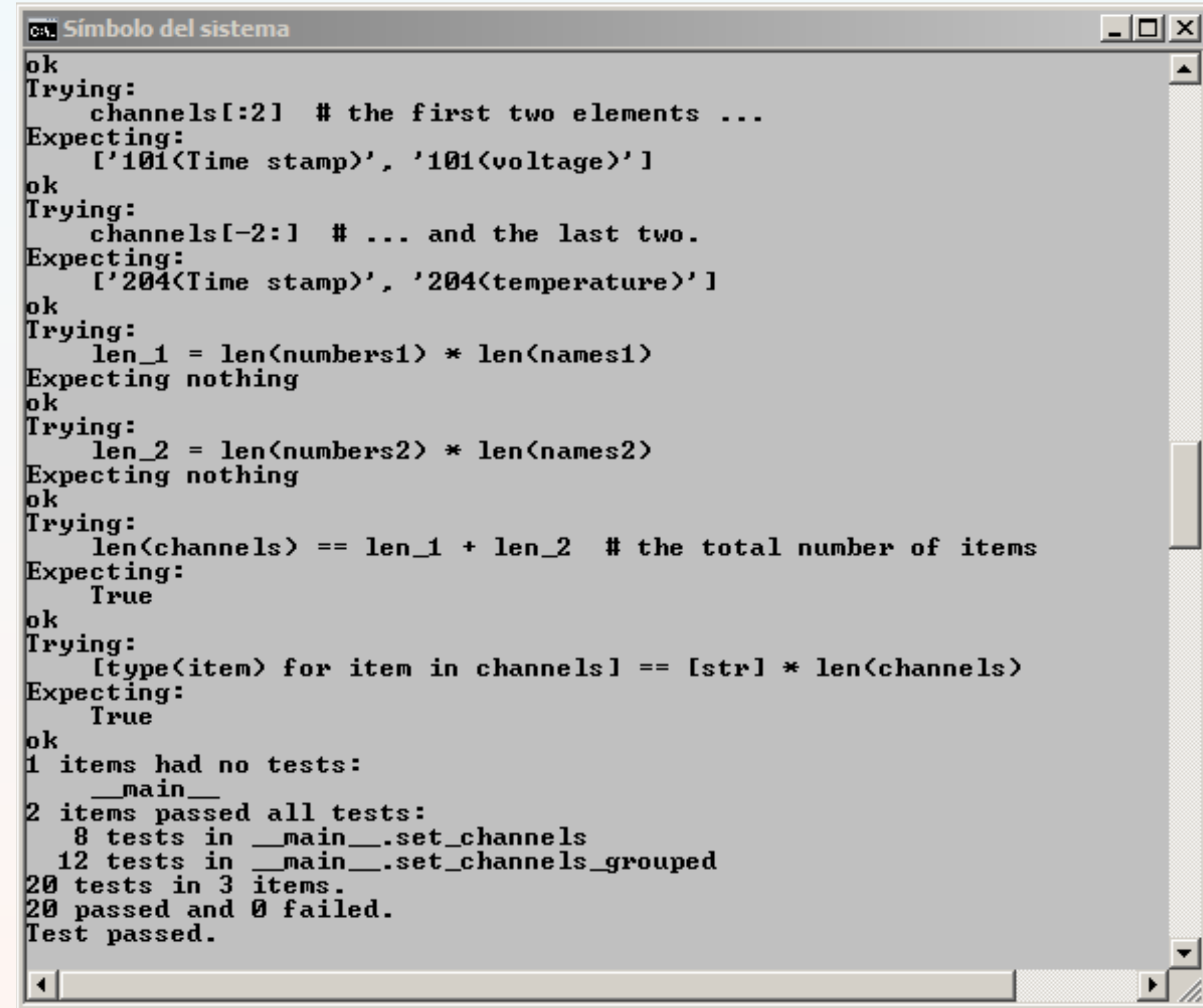
```
....[Resultado esperado]
```

```
....# Linea en blanco
```

```
....**Ejemplo 2**: caso de uso 2
```

```
....# etc.
```

```
.....
```



```
Símbolo del sistema
ok
Trying:
    channels[:2] # the first two elements ...
Expecting:
    ['101<Time stamp>', '101<voltage>']
ok
Trying:
    channels[-2:] # ... and the last two.
Expecting:
    ['204<Time stamp>', '204<temperature>']
ok
Trying:
    len_1 = len(numbers1) * len(names1)
Expecting nothing
ok
Trying:
    len_2 = len(numbers2) * len(names2)
Expecting nothing
ok
Trying:
    len(channels) == len_1 + len_2 # the total number of items
Expecting:
    True
ok
Trying:
    [type(item) for item in channels] == [str] * len(channels)
Expecting:
    True
ok
1 items had no tests:
  __main__
2 items passed all tests:
  8 tests in __main__.set_channels
 12 tests in __main__.set_channels_grouped
20 tests in 3 items.
20 passed and 0 failed.
Test passed.
```


Bases de la Programación en Python

6. Validación del Código

Bases de la Programación en Python

Bases de Programación en Python

Validación del código con *unittest*

Cada módulo '**module**' se valida con un módulo '**test_module**'.

Para cada función '**function**' de '**module**' se define una clase test, que hereda de la clase '**unittest.TestCase**'.

La validación se basa en definir un método 'test_function_use**' para verificar cada caso de uso de la función '**function**'.**

class FunctionTest(unittest.TestCase):

....**"""Tests de la función '**function**'."""** # Descripción de la clase

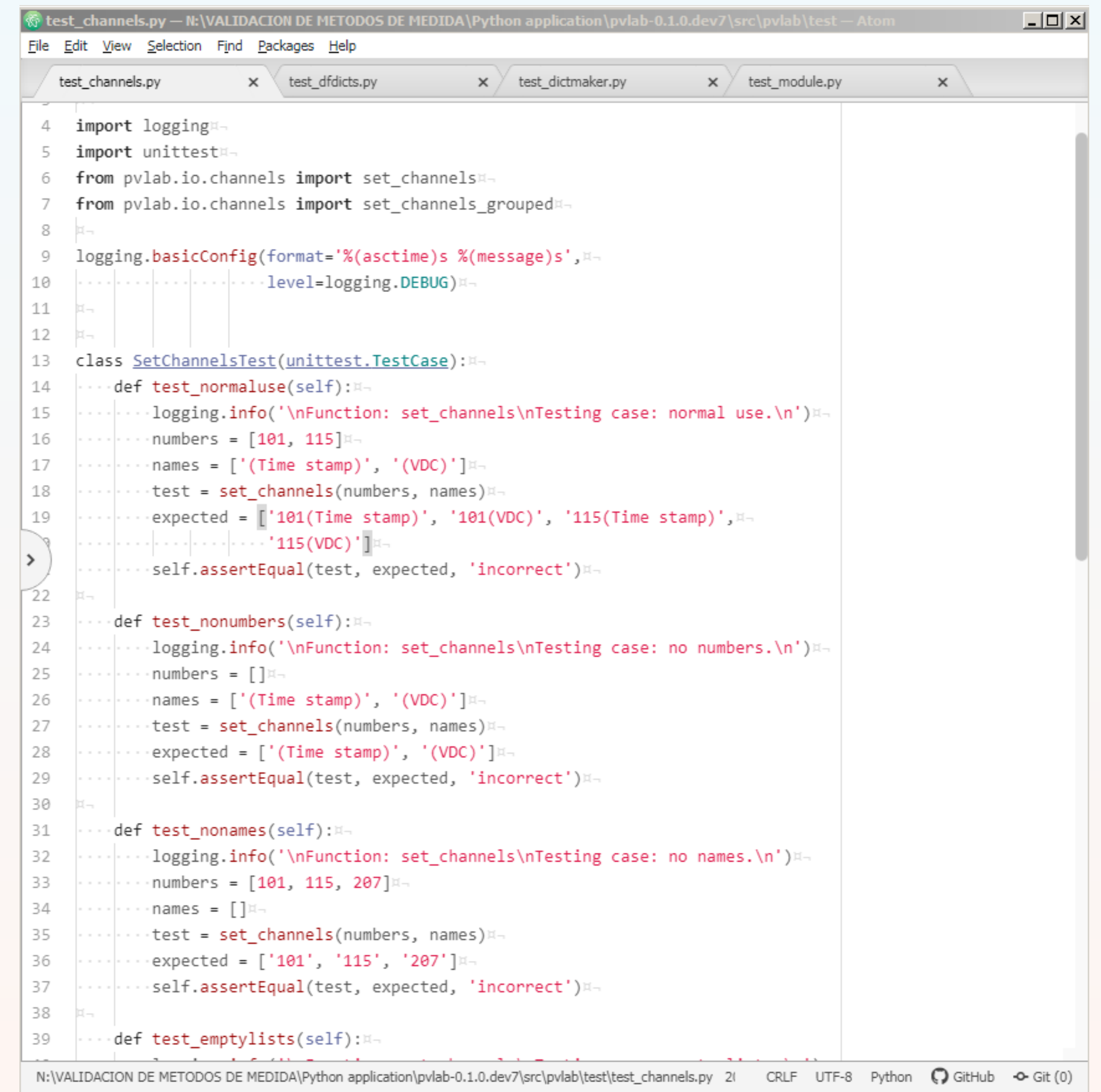
....def **test_function_use1**(self): # Método para el caso de uso 1

.... def **test_function_use2**(self): # Método para el caso de uso 2, ...

Una vez programados los métodos test, se ejecutan todos de una vez:

Ejemplo: if `__name__ == '__main__':`

....`unittest.main(verbosity=1)`



```
test_channels.py — N:\VALIDACION DE METODOS DE MEDIDA\Python application\pvlab-0.1.0.dev7\src\pvlab\test — Atom
File Edit View Selection Find Packages Help
test_channels.py x test_dfdicts.py x test_dictmaker.py x test_module.py x
4 import logging
5 import unittest
6 from pvlab.io.channels import set_channels
7 from pvlab.io.channels import set_channels_grouped
8
9 logging.basicConfig(format='%(asctime)s %(message)s',
10                     level=logging.DEBUG)
11
12
13 class SetChannelsTest(unittest.TestCase):
14     def test_normaluse(self):
15         logging.info('\nFunction: set_channels\nTesting case: normal use.\n')
16         numbers = [101, 115]
17         names = ['(Time stamp)', '(VDC)']
18         test = set_channels(numbers, names)
19         expected = ['101(Time stamp)', '101(VDC)', '115(Time stamp)',
20                   '115(VDC)']
21         self.assertEqual(test, expected, 'incorrect')
22
23     def test_nonumbers(self):
24         logging.info('\nFunction: set_channels\nTesting case: no numbers.\n')
25         numbers = []
26         names = ['(Time stamp)', '(VDC)']
27         test = set_channels(numbers, names)
28         expected = ['(Time stamp)', '(VDC)']
29         self.assertEqual(test, expected, 'incorrect')
30
31     def test_nonames(self):
32         logging.info('\nFunction: set_channels\nTesting case: no names.\n')
33         numbers = [101, 115, 207]
34         names = []
35         test = set_channels(numbers, names)
36         expected = ['101', '115', '207']
37         self.assertEqual(test, expected, 'incorrect')
38
39     def test_emptylists(self):
```

Bases de la Programación en Python

Bases de Programación en Python

Validación del código con *unittest*

Cada módulo '**module**' se valida con un módulo '**test_module**'.

Para cada función '**function**' de '**module**' se define una clase test, que hereda de la clase '**unittest.TestCase**'.

La validación se basa en definir un método 'test_function_use**' para verificar cada caso de uso de la función '**function**'.**

class FunctionTest(unittest.TestCase):

....**"""Tests de la función '**function**'."""** *# Descripción de la clase*

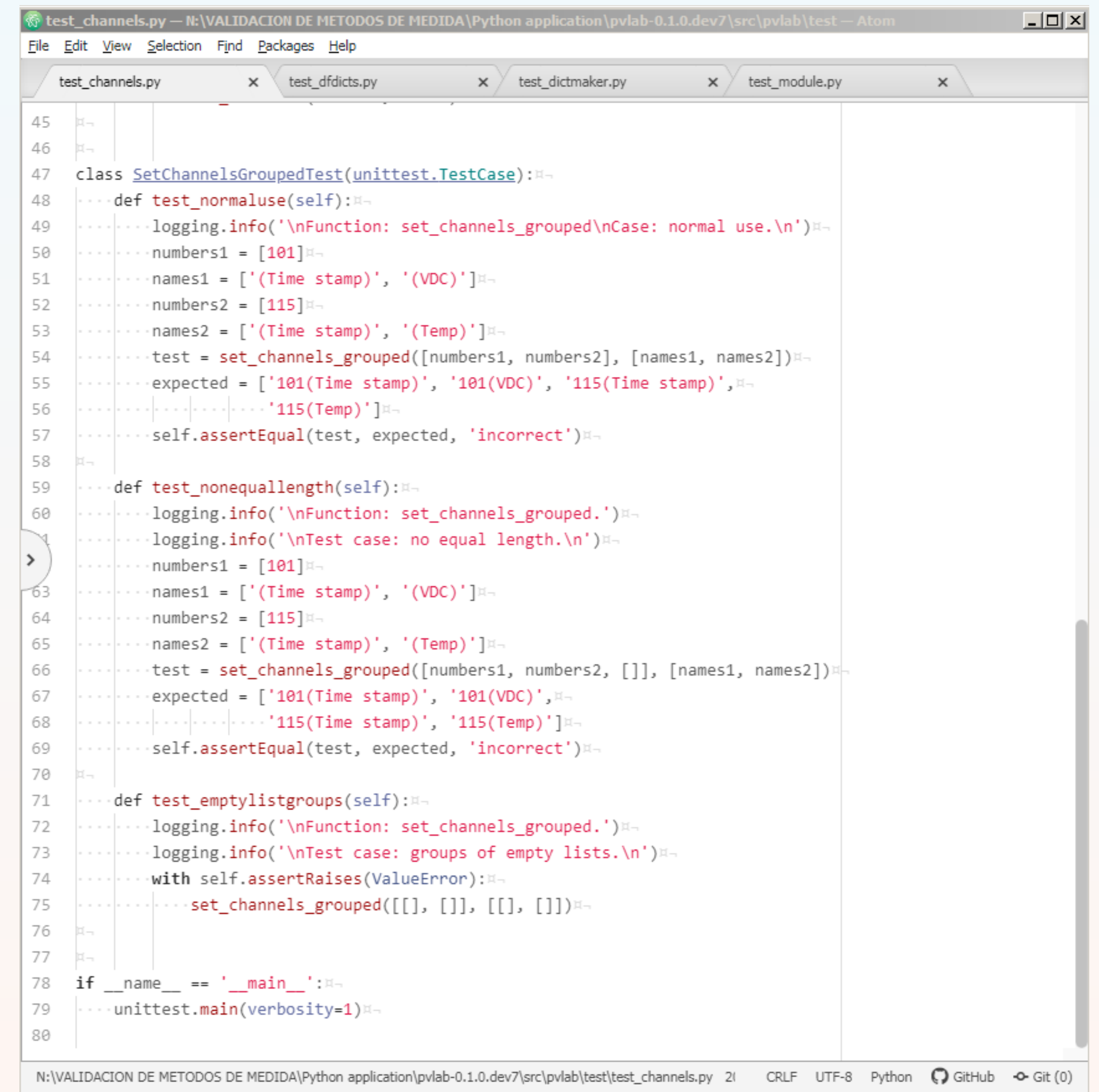
....def **test_function_use1**(self): *# Método para el caso de uso 1*

.... def **test_function_use2**(self): *# Método para el caso de uso 2, ...*

Una vez programados los métodos test, se ejecutan todos de una vez:

Ejemplo: if __name__ == '__main__':

....unittest.main(verbosity=1)



```
test_channels.py — N:\VALIDACION DE METODOS DE MEDIDA\Python application\pvlab-0.1.0.dev7\src\pvlab\test — Atom
File Edit View Selection Find Packages Help
test_channels.py x test_dfdicts.py x test_dictmaker.py x test_module.py x
45
46
47 class SetChannelsGroupedTest(unittest.TestCase):
48     def test_normaluse(self):
49         logging.info('\nFunction: set_channels_grouped\nCase: normal use.\n')
50         numbers1 = [101]
51         names1 = ['(Time stamp)', '(VDC)']
52         numbers2 = [115]
53         names2 = ['(Time stamp)', '(Temp)']
54         test = set_channels_grouped([numbers1, numbers2], [names1, names2])
55         expected = ['101(Time stamp)', '101(VDC)', '115(Time stamp)',
56                     '115(Temp)']
57         self.assertEqual(test, expected, 'incorrect')
58
59     def test_nonequallength(self):
60         logging.info('\nFunction: set_channels_grouped.')
61         logging.info('\nTest case: no equal length.\n')
62         numbers1 = [101]
63         names1 = ['(Time stamp)', '(VDC)']
64         numbers2 = [115]
65         names2 = ['(Time stamp)', '(Temp)']
66         test = set_channels_grouped([numbers1, numbers2, []], [names1, names2])
67         expected = ['101(Time stamp)', '101(VDC)',
68                     '115(Time stamp)', '115(Temp)']
69         self.assertEqual(test, expected, 'incorrect')
70
71     def test_emptylistgroups(self):
72         logging.info('\nFunction: set_channels_grouped.')
73         logging.info('\nTest case: groups of empty lists.\n')
74         with self.assertRaises(ValueError):
75             set_channels_grouped([], [], [], [])
76
77
78 if __name__ == '__main__':
79     unittest.main(verbosity=1)
80
```


Bases de la Programación en Python

Bases de Programación en Python

Validación del código con *unittest*

Cada módulo '**module**' se valida con un módulo '**test_module**'.

Para cada función '**function**' de '**module**' se define una clase test, que hereda de la clase '**unittest.TestCase**'.

La validación se basa en definir un método 'test_function_use**' para verificar cada caso de uso de la función '**function**'.**

class FunctionTest(unittest.TestCase):

....**"""Tests de la función '**function**'."""** *# Descripción de la clase*

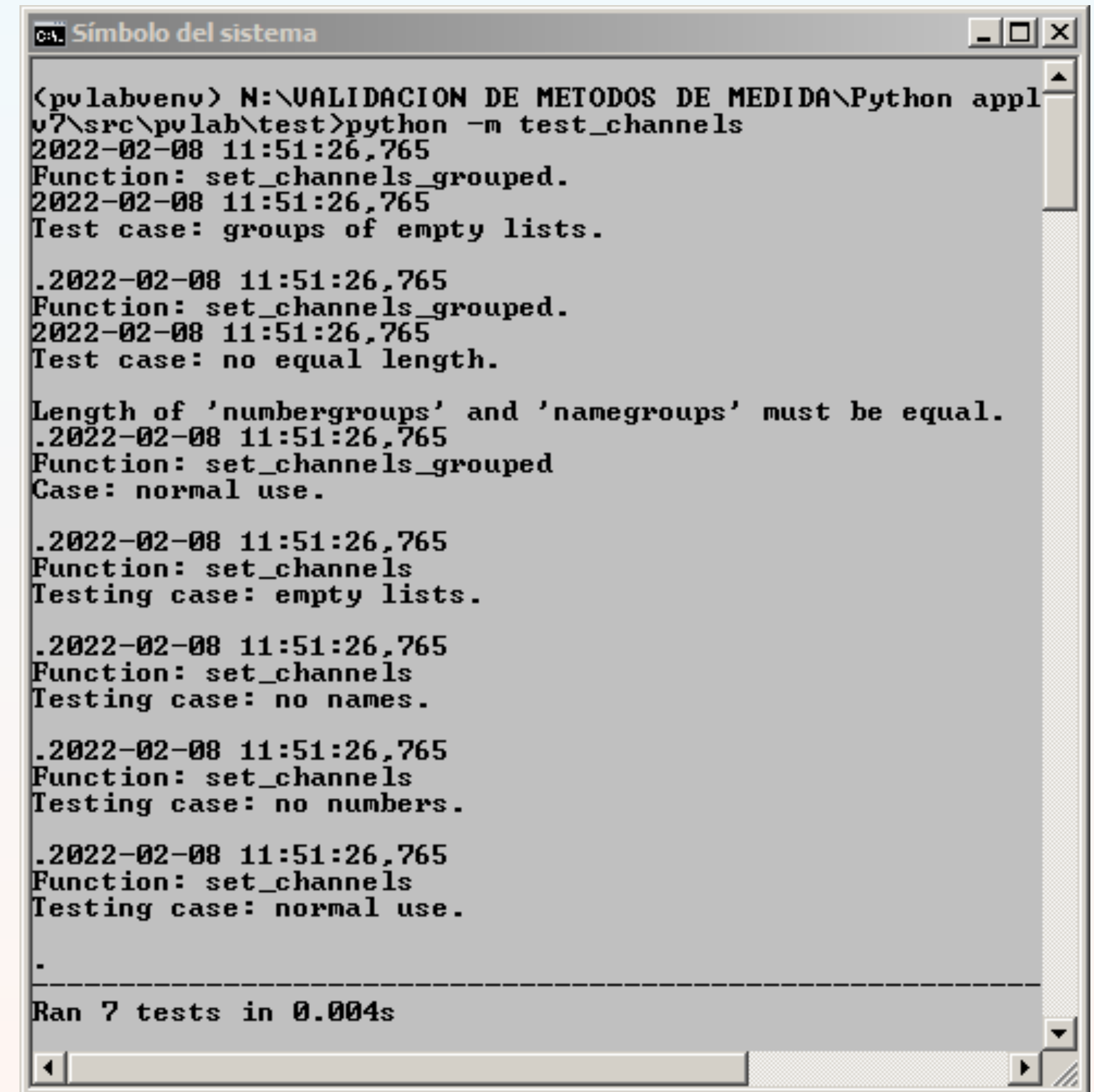
....def **test_function_use1**(self): *# Método para el caso de uso 1*

.... def **test_function_use2**(self): *# Método para el caso de uso 2, ...*

Una vez programados los métodos test, se ejecutan todos de una vez:

Ejemplo: if __name__ == '__main__':

....unittest.main(verbosity=1)



```
<pvlabvnu> N:\VALIDACION DE METODOS DE MEDIDA\Python appl
v7\src\pvlab\test>python -m test_channels
2022-02-08 11:51:26,765
Function: set_channels_grouped.
2022-02-08 11:51:26,765
Test case: groups of empty lists.

.2022-02-08 11:51:26,765
Function: set_channels_grouped.
2022-02-08 11:51:26,765
Test case: no equal length.

Length of 'numbergroups' and 'namegroups' must be equal.
.2022-02-08 11:51:26,765
Function: set_channels_grouped
Case: normal use.

.2022-02-08 11:51:26,765
Function: set_channels
Testing case: empty lists.

.2022-02-08 11:51:26,765
Function: set_channels
Testing case: no names.

.2022-02-08 11:51:26,765
Function: set_channels
Testing case: no numbers.

.2022-02-08 11:51:26,765
Function: set_channels
Testing case: normal use.

.
-----
Ran 7 tests in 0.004s
```

(II - 4/5) Elaboración del Proyecto Documental

Recursos para el Desarrollo de Software

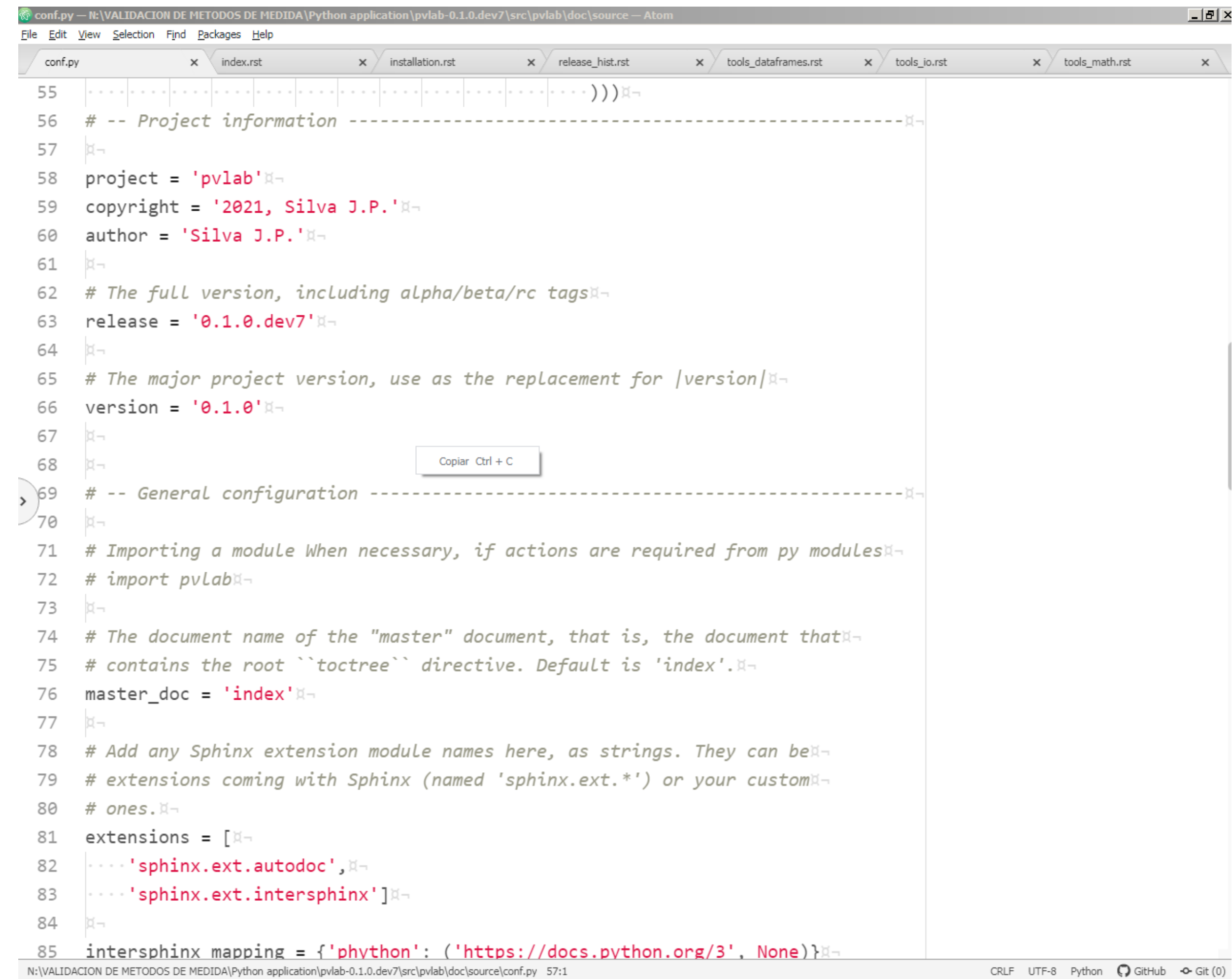
Proyecto Documental

Documentación del Proyecto con *sphinx*

Se trata de generar la documentación del proyecto, de modo que ésta pueda ser consultada por terceras personas. Es especialmente recomendable para **proyectos públicos**, excepto aplicaciones muy pequeñas.

Se utiliza una aplicación, p.ej. **sphinx**, que 'compila' un directorio con **archivos de configuración** y **archivos de texto**, **generando un proyecto documental** en formato *.tex* (convertible a *.pdf* mediante *miktex*, p.ej.), o bien *.html*, etc.

La aplicación **sphinx** contiene archivos de estilo (fuentes para títulos, cuerpo del texto, código, etc.) y las directivas de los lenguajes de texto **Markdown** (archivos *.md*) y **ReStructuredText** (archivos *.rst*), aparte de directivas adicionales para este último (extensiones, p.ej. **autodoc** o **intersphinx**).



```
conf.py — N:\VALIDACION DE METODOS DE MEDIDA\Python application\pvlab-0.1.0.dev7\src\pvlab\doc\source — Atom
File Edit View Selection Find Packages Help
conf.py x index.rst x installation.rst x release_hist.rst x tools_dataframes.rst x tools_io.rst x tools_math.rst x
55 | .....)))
56 | # -- Project information -----
57 |
58 | project = 'pvlab'
59 | copyright = '2021, Silva J.P.'
60 | author = 'Silva J.P.'
61 |
62 | # The full version, including alpha/beta/rc tags
63 | release = '0.1.0.dev7'
64 |
65 | # The major project version, use as the replacement for |version|
66 | version = '0.1.0'
67 |
68 |
69 | # -- General configuration -----
70 |
71 | # Importing a module When necessary, if actions are required from py modules
72 | # import pvlab
73 |
74 | # The document name of the "master" document, that is, the document that
75 | # contains the root ``toctree`` directive. Default is 'index'.
76 | master_doc = 'index'
77 |
78 | # Add any Sphinx extension module names here, as strings. They can be
79 | # extensions coming with Sphinx (named 'sphinx.ext.*') or your custom
80 | # ones.
81 | extensions = [
82 |     'sphinx.ext.autodoc',
83 |     'sphinx.ext.intersphinx'
84 | ]
85 | intersphinx_mapping = {'python': ('https://docs.python.org/3', None)}
```


Recursos para el Desarrollo de Software

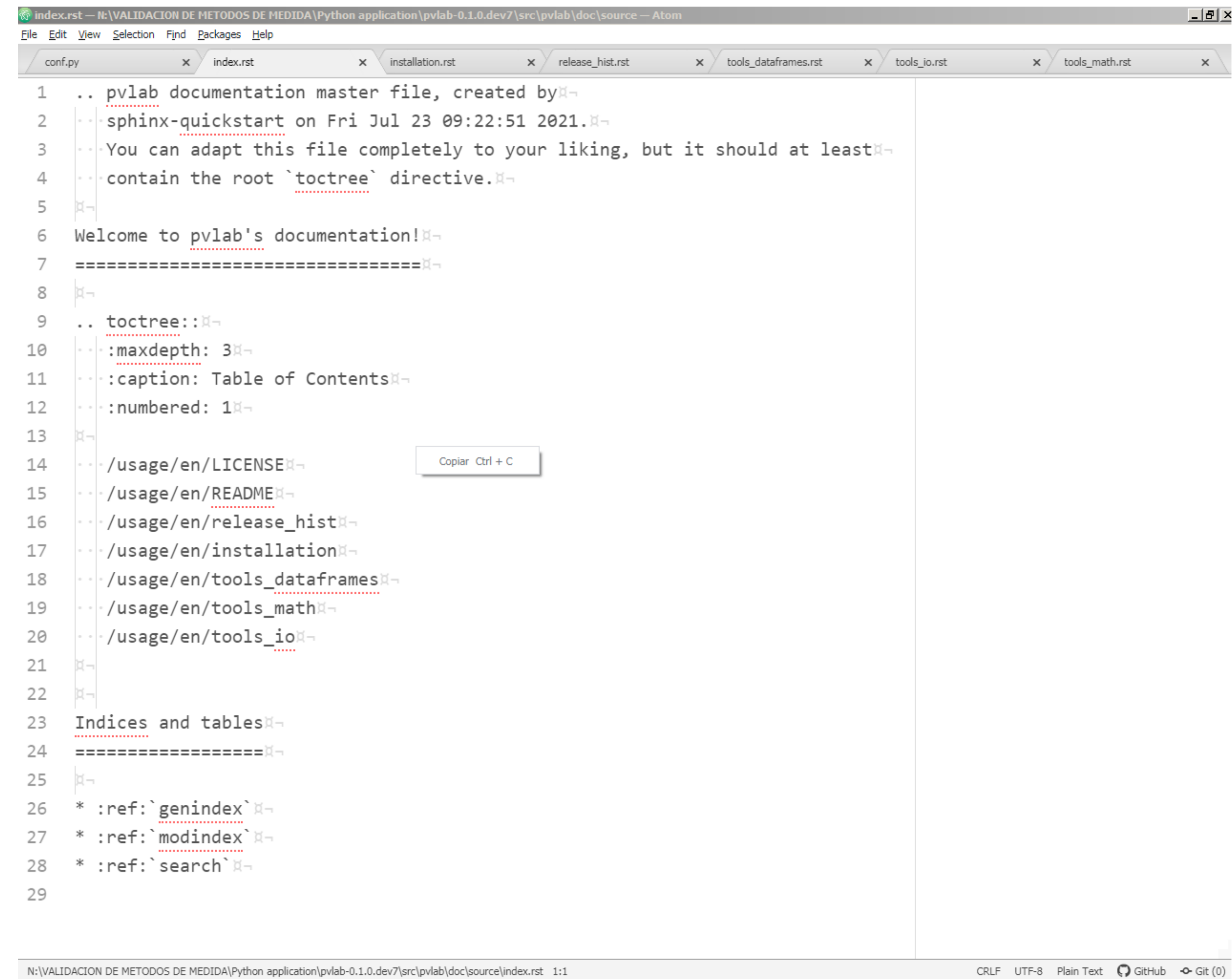
Proyecto Documental

Documentación del Proyecto con *sphinx*

Se trata de generar la documentación del proyecto, de modo que ésta pueda ser consultada por terceras personas. Es especialmente recomendable para **proyectos públicos**, excepto aplicaciones muy pequeñas.

Se utiliza una aplicación, p.ej. **sphinx**, que 'compila' un directorio con **archivos de configuración** y **archivos de texto**, **generando un proyecto documental** en formato *.tex* (convertible a *.pdf* mediante *miktex*, p.ej.), o bien *.html*, etc.

La aplicación **sphinx** contiene archivos de estilo (fuentes para títulos, cuerpo del texto, código, etc.) y las directivas de los lenguajes de texto **Markdown** (archivos *.md*) y **ReStructuredText** (archivos *.rst*), aparte de directivas adicionales para este último (extensiones, p.ej. **autodoc** o **intersphinx**).



```
index.rst — N:\VALIDACION DE METODOS DE MEDIDA\Python application\pvlab-0.1.0.dev7\src\pvlab\doc\source — Atom
File Edit View Selection Find Packages Help
conf.py x index.rst x installation.rst x release_hist.rst x tools_dataframes.rst x tools_io.rst x tools_math.rst x
1 .. pvlab documentation master file, created by
2 .. sphinx-quickstart on Fri Jul 23 09:22:51 2021.
3 .. You can adapt this file completely to your liking, but it should at least
4 .. contain the root `toctree` directive.
5
6 Welcome to pvlab's documentation!
7
8
9 .. toctree::
10 ..     :maxdepth: 3
11 ..     :caption: Table of Contents
12 ..     :numbered: 1
13
14 .. /usage/en/LICENSE
15 .. /usage/en/README
16 .. /usage/en/release_hist
17 .. /usage/en/installation
18 .. /usage/en/tools_dataframes
19 .. /usage/en/tools_math
20 .. /usage/en/tools_io
21
22
23 Indices and tables
24
25
26 * :ref:`genindex`
27 * :ref:`modindex`
28 * :ref:`search`
29
```

Recursos para el Desarrollo de Software

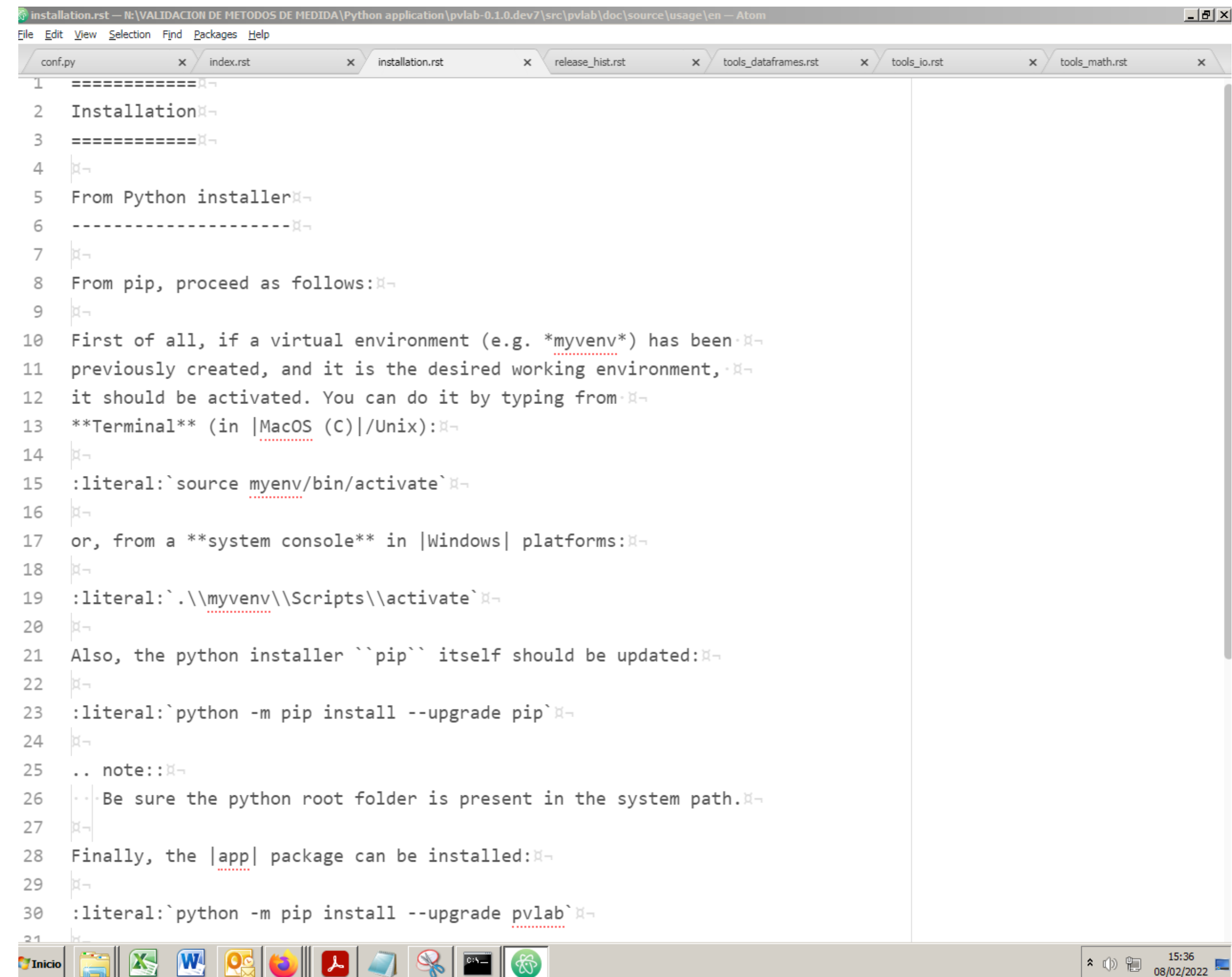
Proyecto Documental

Documentación del Proyecto con *sphinx*

Se trata de generar la documentación del proyecto, de modo que ésta pueda ser consultada por terceras personas. Es especialmente recomendable para **proyectos públicos**, excepto aplicaciones muy pequeñas.

Se utiliza una aplicación, p.ej. **sphinx**, que 'compila' un directorio con **archivos de configuración** y **archivos de texto**, **generando un proyecto documental** en formato *.tex* (convertible a *.pdf* mediante *miktex*, p.ej.), o bien *.html*, etc.

La aplicación **sphinx** contiene archivos de estilo (fuentes para títulos, cuerpo del texto, código, etc.) y las directivas de los lenguajes de texto **Markdown** (archivos *.md*) y **ReStructuredText** (archivos *.rst*), aparte de directivas adicionales para este último (extensiones, p.ej. **autodoc** o **intersphinx**).



```
1 =====
2 Installation
3 =====
4
5 From Python installer
6 -----
7
8 From pip, proceed as follows:
9
10 First of all, if a virtual environment (e.g. *myenv*) has been
11 previously created, and it is the desired working environment,
12 it should be activated. You can do it by typing from
13 **Terminal** (in |MacOS (C)|/Unix):
14
15 :literal:`source myenv/bin/activate`
16
17 or, from a **system console** in |Windows| platforms:
18
19 :literal:`.\myenv\Scripts\activate`
20
21 Also, the python installer `pip` itself should be updated:
22
23 :literal:`python -m pip install --upgrade pip`
24
25 .. note::
26 .. Be sure the python root folder is present in the system path.
27
28 Finally, the |app| package can be installed:
29
30 :literal:`python -m pip install --upgrade pvlab`
```

Recursos para el Desarrollo de Software

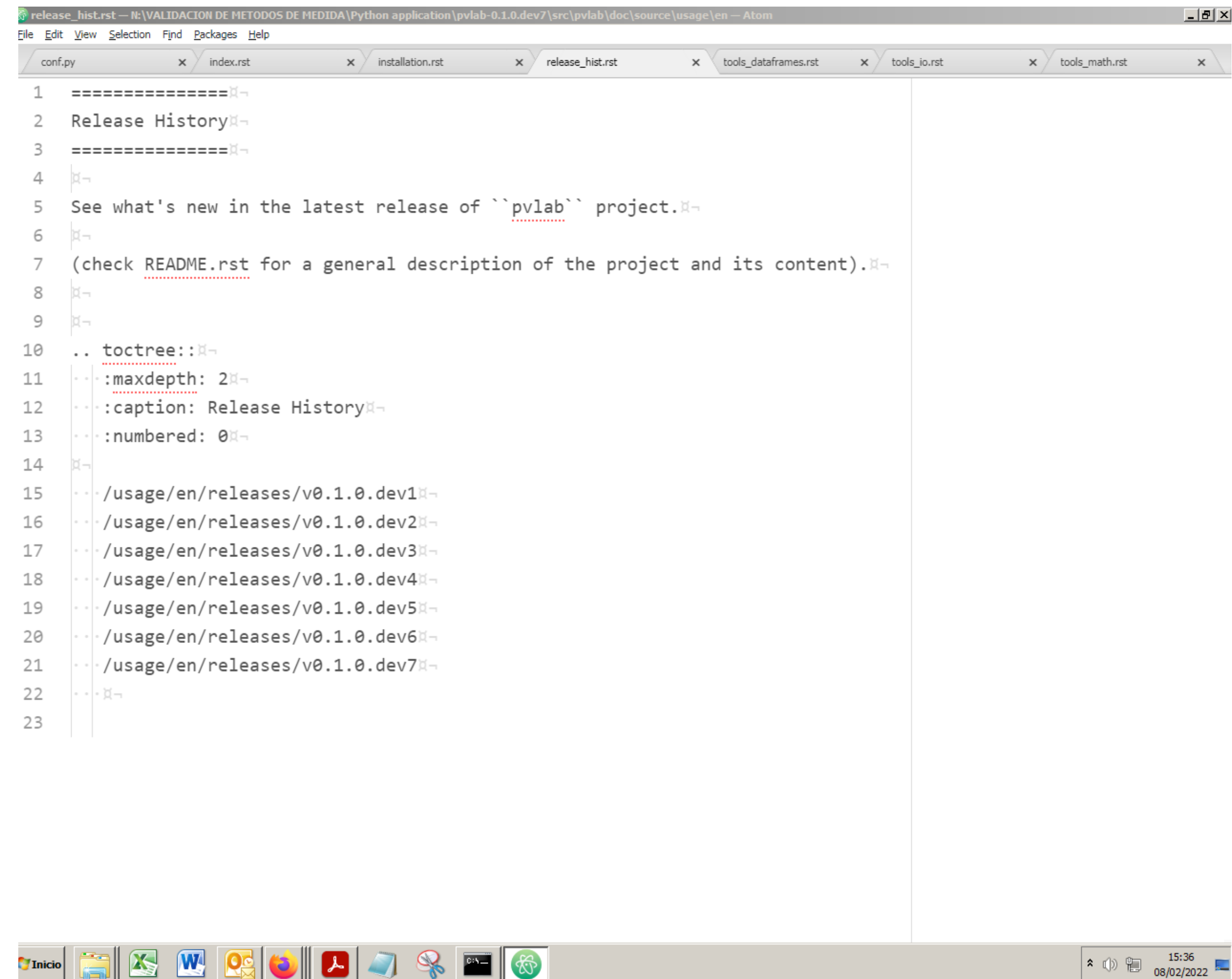
Proyecto Documental

Documentación del Proyecto con *sphinx*

Se trata de generar la documentación del proyecto, de modo que ésta pueda ser consultada por terceras personas. Es especialmente recomendable para **proyectos públicos**, excepto aplicaciones muy pequeñas.

Se utiliza una aplicación, p.ej. **sphinx**, que 'compila' un directorio con **archivos de configuración** y **archivos de texto**, **generando un proyecto documental** en formato *.tex* (convertible a *.pdf* mediante *miktex*, p.ej.), o bien *.html*, etc.

La aplicación **sphinx** contiene archivos de estilo (fuentes para títulos, cuerpo del texto, código, etc.) y las directivas de los lenguajes de texto **Markdown** (archivos *.md*) y **ReStructuredText** (archivos *.rst*), aparte de directivas adicionales para este último (extensiones, p.ej. **autodoc** o **intersphinx**).



```
1 =====
2 Release History
3 =====
4
5 See what's new in the latest release of ``pvlab`` project.
6
7 (check README.rst for a general description of the project and its content).
8
9
10 .. toctree::
11    :maxdepth: 2
12    :caption: Release History
13    :numbered: 0
14
15    /usage/en/releases/v0.1.0.dev1
16    /usage/en/releases/v0.1.0.dev2
17    /usage/en/releases/v0.1.0.dev3
18    /usage/en/releases/v0.1.0.dev4
19    /usage/en/releases/v0.1.0.dev5
20    /usage/en/releases/v0.1.0.dev6
21    /usage/en/releases/v0.1.0.dev7
22
23
```


Recursos para el Desarrollo de Software

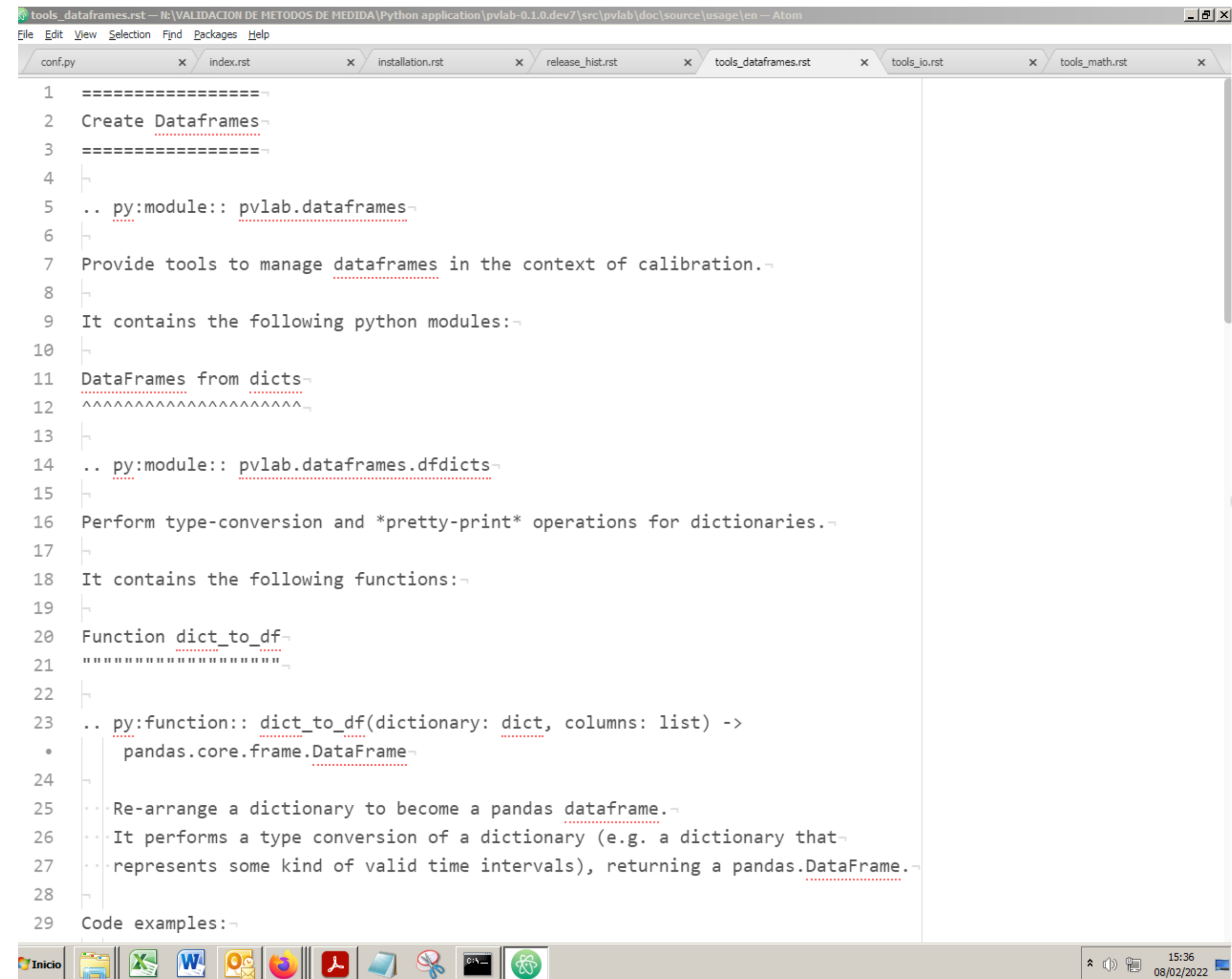
Proyecto Documental

Documentación del Proyecto con *sphinx*

Se trata de generar la documentación del proyecto, de modo que ésta pueda ser consultada por terceras personas. Es especialmente recomendable para **proyectos públicos**, excepto aplicaciones muy pequeñas.

Se utiliza una aplicación, p.ej. **sphinx**, que 'compila' un directorio con **archivos de configuración** y **archivos de texto**, **generando un proyecto documental** en formato *.tex* (convertible a *.pdf* mediante *miktex*, p.ej.), o bien *.html*, etc.

La aplicación **sphinx** contiene archivos de estilo (fuentes para títulos, cuerpo del texto, código, etc.) y las directivas de los lenguajes de texto **Markdown** (archivos *.md*) y **ReStructuredText** (archivos *.rst*), aparte de directivas adicionales para este último (extensiones, p.ej. **autodoc** o **intersphinx**).



```
1  =====~
2  Create Dataframes~
3  =====~
4  |
5  .. py:module:: pvlab.dataframes~
6  |
7  Provide tools to manage dataframes in the context of calibration.~
8  |
9  It contains the following python modules:~
10 |
11 DataFrames from dicts~
12 ~~~~~~
13 |
14 .. py:module:: pvlab.dataframes.dfdicts~
15 |
16 Perform type-conversion and *pretty-print* operations for dictionaries.~
17 |
18 It contains the following functions:~
19 |
20 Function dict_to_df~
21 ~~~~~~
22 |
23 .. py:function:: dict_to_df(dictionary: dict, columns: list) ->
24     * pandas.core.frame.DataFrame~
25 |
26 .. Re-arrange a dictionary to become a pandas dataframe.~
27 .. It performs a type conversion of a dictionary (e.g. a dictionary that~
28 .. represents some kind of valid time intervals), returning a pandas.DataFrame.~
29 |
30 Code examples:~
```

Recursos para el Desarrollo de Software

Proyecto Documental

Documentación del Proyecto con *sphinx*

Se trata de generar la documentación del proyecto, de modo que ésta pueda ser consultada por terceras personas. Es especialmente recomendable para **proyectos públicos**, excepto aplicaciones muy pequeñas.

Se utiliza una aplicación, p.ej. **sphinx**, que 'compila' un directorio con **archivos de configuración** y **archivos de texto**, **generando un proyecto documental** en formato `.tex` (convertible a `.pdf` mediante *miktex*, p.ej.), o bien `.html`, etc.

La aplicación **sphinx** contiene archivos de estilo (fuentes para títulos, cuerpo del texto, código, etc.) y las directivas de los lenguajes de texto **Markdown** (archivos .md) y **ReStructuredText** (archivos .rst), aparte de directivas adicionales para este último (extensiones, p.ej. **autodoc** o **intersphinx**).

tools_io.rst — N:\VALIDACION DE METODOS DE MEDIDA\Python application\pvlab-0.1.0.dev7\src\pvlab\doc\source\usage\en — Atom

File Edit View Selection Find Packages Help

conf.py x index.rst x installation.rst x release_hist.rst x tools_dataframes.rst x tools_io.rst x tools_math.rst x

```
1 =====
2 I/O Management
3 =====
4
5 Provide tools for data input/output.
6
7 Create dicts from Source Files
8 ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
9
10 .. py:module:: pvlab.io.dictmaker
11
12 Generate python dictionaries from multiple files and data types.
13
14 New functions are intended to read files of parameters and convert each
15 data file into a python dictionary. Different type of parameters should be
16 located in different files, for better performance.
17
18 It contains the following functions:
19
20 Function get_dict
21
22
23 .. py:function:: pvlab.io.dictmaker.get_dict(file: TypeVar('File', str,
24 * io.StringIO), dtype: str, sep: str = ':', isStringIO: bool = False) -> dict:
25
26 .. Generate a python dictionary from a file of parameters.
27
28 .. Function ``get_dict`` reads a file that contains parameters in the form
29 .. *[name]: [value]* (or in other general form *[name][sep] [value]*, if
30 .. specified).
31
32 .. It requires specifying the type of data contained in the file, admitting
```

N:\VALIDACION DE METODOS DE MEDIDA\Python application\pvlab-0.1.0.dev7\src\pvlab\doc\source\usage\en\tools_io.rst 1:1

CRLF UTF-8 Plain Text GitHub Git (0)

Recursos para el Desarrollo de Software

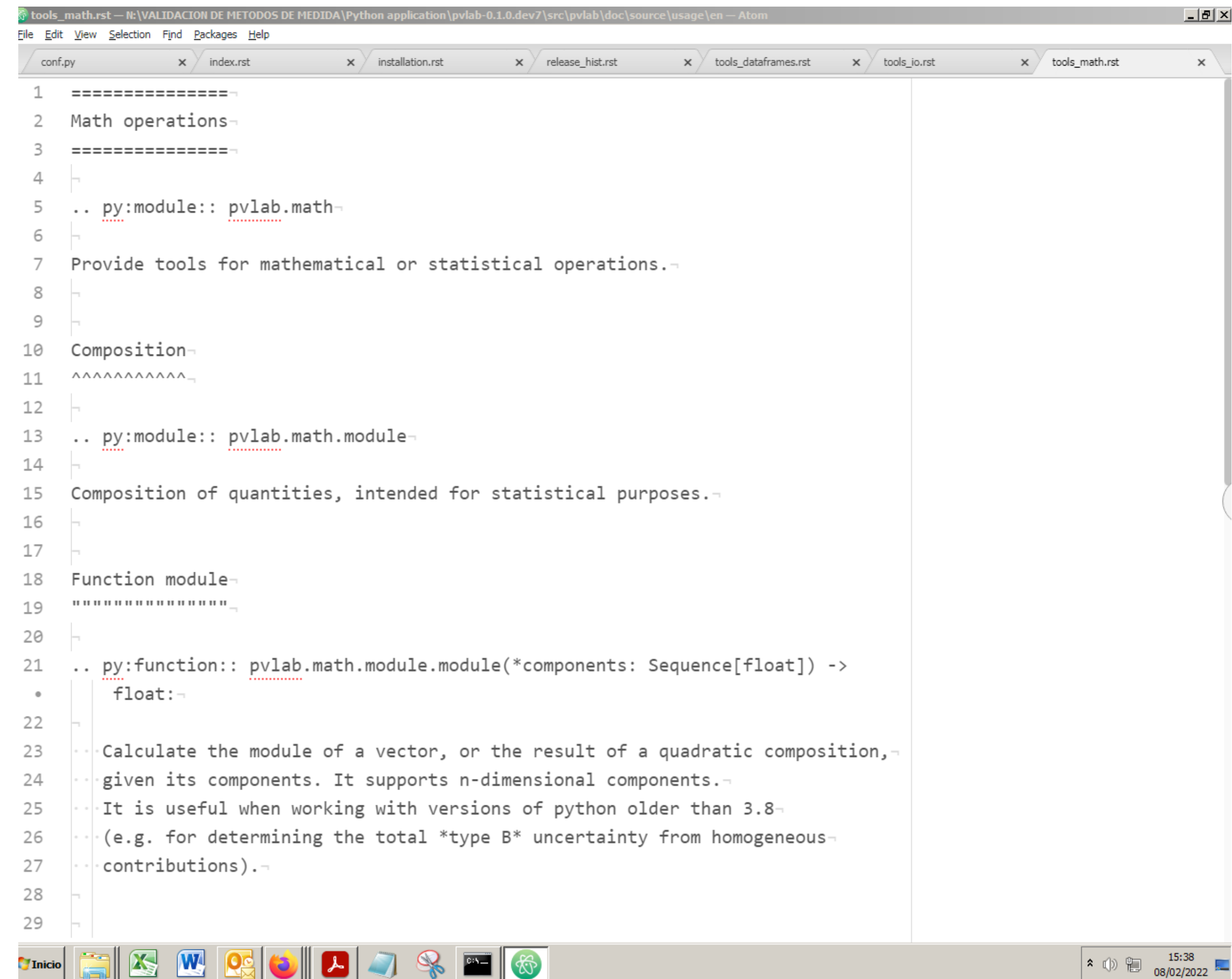
Proyecto Documental

Documentación del Proyecto con *sphinx*

Se trata de generar la documentación del proyecto, de modo que ésta pueda ser consultada por terceras personas. Es especialmente recomendable para **proyectos públicos**, excepto aplicaciones muy pequeñas.

Se utiliza una aplicación, p.ej. **sphinx**, que 'compila' un directorio con **archivos de configuración** y **archivos de texto**, **generando un proyecto documental** en formato *.tex* (convertible a *.pdf* mediante *miktex*, p.ej.), o bien *.html*, etc.

La aplicación **sphinx** contiene archivos de estilo (fuentes para títulos, cuerpo del texto, código, etc.) y las directivas de los lenguajes de texto **Markdown** (archivos *.md*) y **ReStructuredText** (archivos *.rst*), aparte de directivas adicionales para este último (extensiones, p.ej. **autodoc** o **intersphinx**).



```
1 =====
2 Math operations
3 =====
4
5 .. py:module:: pvlab.math
6
7 Provide tools for mathematical or statistical operations.
8
9
10 Composition
11 ^^^^^^^^^^^
12
13 .. py:module:: pvlab.math.module
14
15 Composition of quantities, intended for statistical purposes.
16
17
18 Function module
19 =====
20
21 .. py:function:: pvlab.math.module.module(*components: Sequence[float]) ->
22     float:
23
24     Calculate the module of a vector, or the result of a quadratic composition,
25     given its components. It supports n-dimensional components.
26     It is useful when working with versions of python older than 3.8
27     (e.g. for determining the total *type B* uncertainty from homogeneous
28     contributions).
29
```


Recursos para el Desarrollo de Software

Proyecto Documental

Documentación del Proyecto con *sphinx*

Se trata de generar la documentación del proyecto, de modo que ésta pueda ser consultada por terceras personas. Es especialmente recomendable para **proyectos públicos**, excepto aplicaciones muy pequeñas.

Se utiliza una aplicación, p.ej. **sphinx**, que 'compila' un directorio con **archivos de configuración** y **archivos de texto**, **generando un proyecto documental** en formato *.tex* (convertible a *.pdf* mediante *miktex*, p.ej.), o bien *.html*, etc.

La aplicación **sphinx** contiene archivos de estilo (fuentes para títulos, cuerpo del texto, código, etc.) y las directivas de los lenguajes de texto **Markdown** (archivos *.md*) y **ReStructuredText** (archivos *.rst*), aparte de directivas adicionales para este último (extensiones, p.ej. **autodoc** o **intersphinx**).

I/O MANAGEMENT

Provide tools for data input/output.

7.1 dicts from Source Files

Generate python dictionaries from multiple files and data types.

New functions are intended to read files of parameters and convert each data file into a python dictionary. Different type of parameters should be located in different files, for better performance.

It contains the following functions:

7.1.1 Function `get_dict`

```
pylab.io.dictmaker.get_dict(file: TypeVar('File'), str, io.StringIO), dtype: str, sep: str = ':', isStringIO: bool = False) → dict:
```

Generate a python dictionary from a file of parameters.

Function `get_dict` reads a file that contains parameters in the form `[name]: [value]` (or in other general form `[name][sep] [value]`, if specified). It requires specifying the type of data contained in the file, admitting types `int`, `float`, `tuple` or `str` (unknown data types are admitted, but they will be parsed as *raw* strings). Originally designed for data-acquisition purposes. It also admits `io.StringIO` objects instead, if argument `isStringIO` is specified as `True` (e.g. useful for exemplification purposes).

Example 1: int type arguments:

```
data = "readings:21\nminG:600\nrefG:1000"
settings = get_dict(io.StringIO(data), dtype='int', isStringIO=True)
# ... argument ``io.StringIO(data)`` can be replaced by a file name.
settings
{'readings': 21, 'minG': 600, 'refG': 1000}
```

Example 2: str type arguments:

```
data = "man.: 'manufacturer'\nmod.: 'model'\nsn.: 'seriesnr'"
mydict = get_dict(io.StringIO(data), dtype='str', isStringIO=True)
```

(II - 5/5) Distribución y Publicación

Recursos para el Desarrollo de Software

Distribución y Publicación

Distribución del Proyecto con *flit*

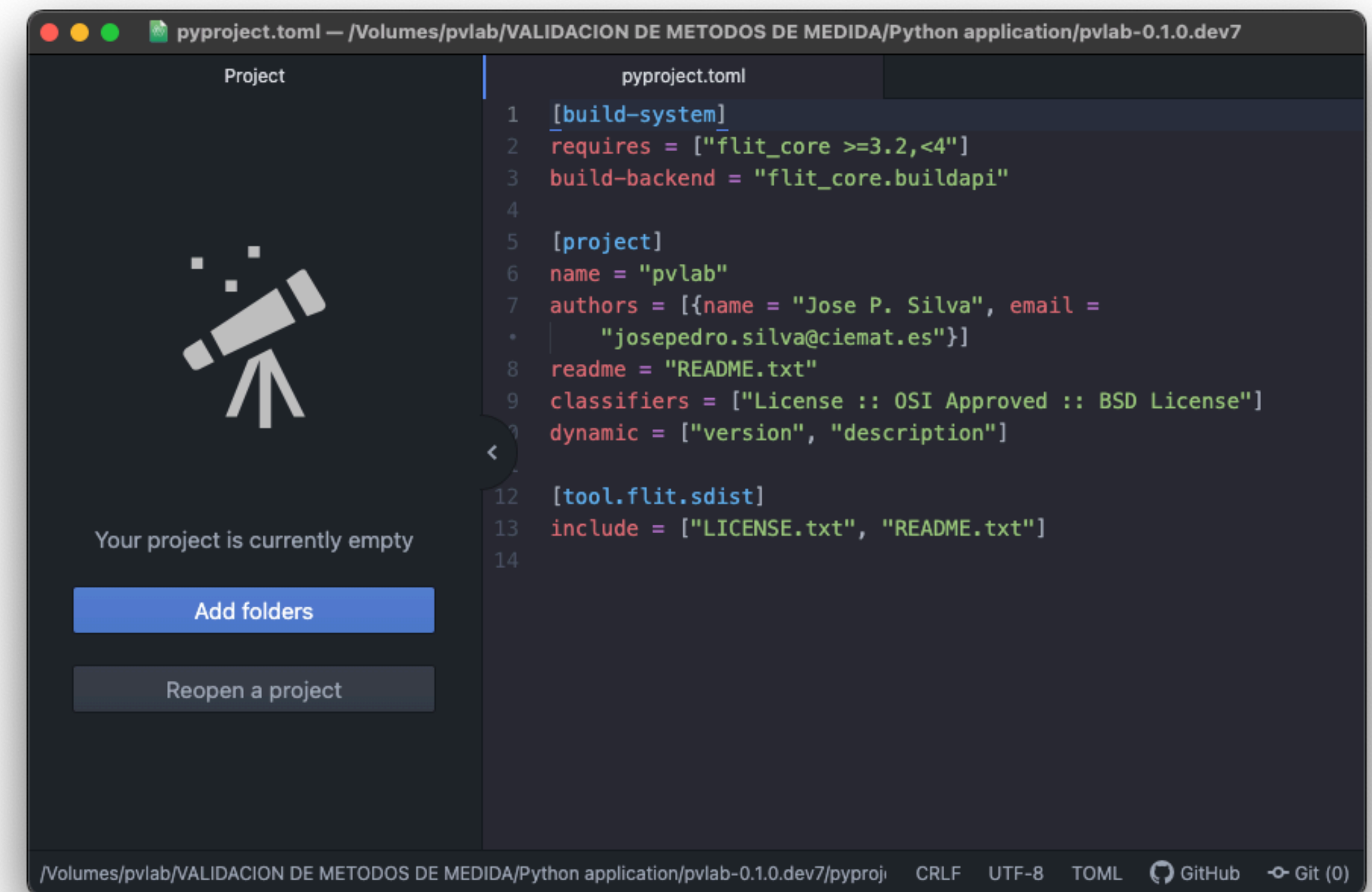
Se trata de generar los archivos finales, es decir, las distribuciones, que contienen el paquete de software.

Se utiliza una aplicación p.ej. *flit*, que genera dos tipos de distribuciones, por ejemplo:

`pvlab-0.1.0.dev7.tar.gz` # “Source distribution” o “distribución fuente”, un archivo comprimido que contiene el directorio del paquete de software.

`pvlab-0.1.0.dev7-py2.py3-none-any.whl` # “Wheel distribution”, es una distribución binaria instalable.

Precisa la elaboración de un archivo de configuración llamado `pyproject.toml` (según PEP517, PEP518, PEP660).



Recursos para el Desarrollo de Software

Distribución y Publicación

Publicación en el *Python Package Index*

Se trata de subir las distribuciones a un servidor, para su posterior acceso, y que el paquete sea **instalable** en el entorno *Python*.

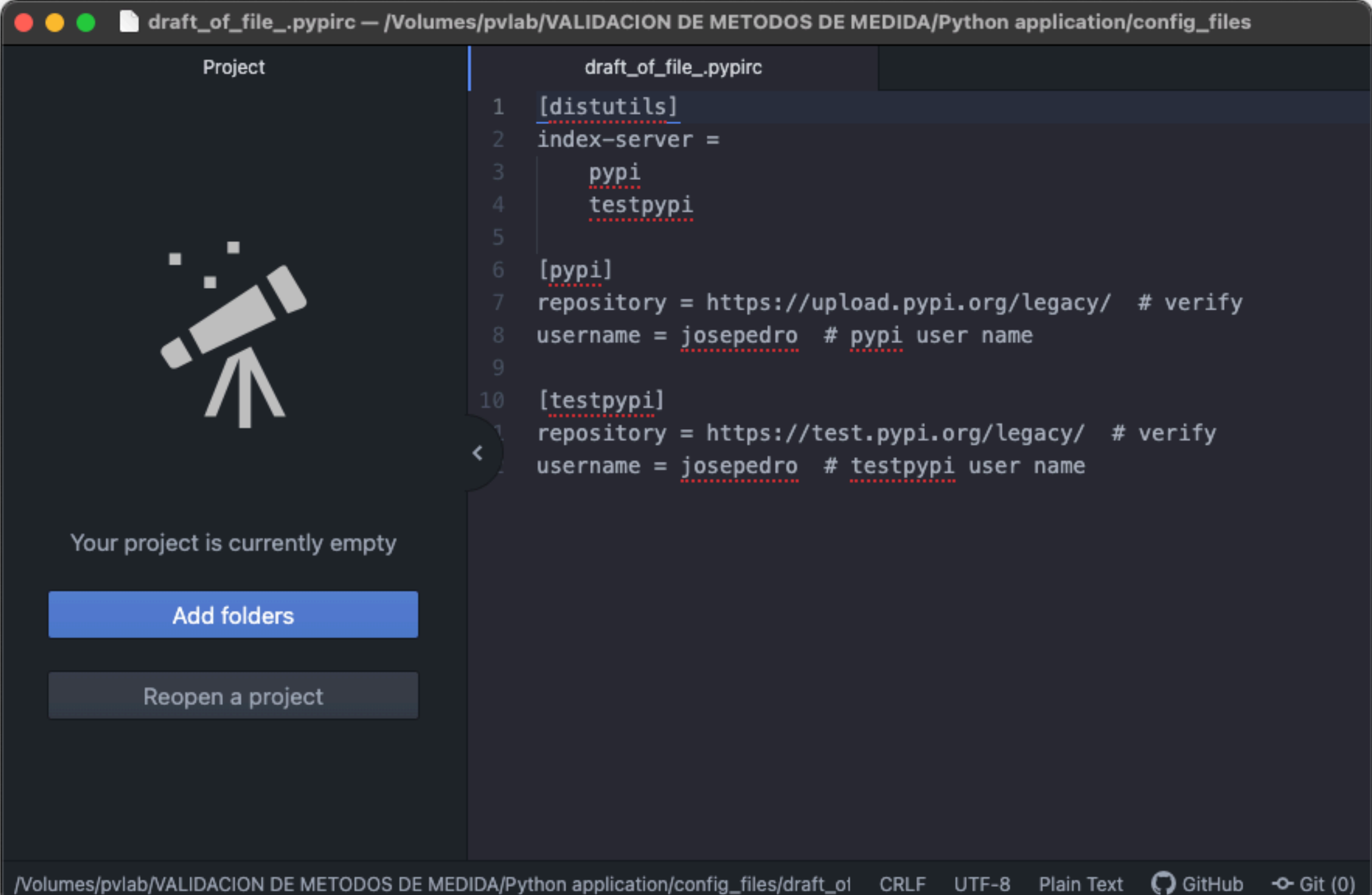
Puede utilizarse para ello la misma aplicación (p.ej. *flit*).

Una vez publicado en PYPI, puede ser

Precisa la elaboración de un archivo de configuración llamado `.pypirc`.

Las credenciales pueden ser guardadas y gestionadas de modo seguro mediante aplicaciones de encriptado como *keyring* (*flit* la busca por defecto, si no está instalada, solicita las credenciales).

Puede establecerse en PYPI un factor de doble autenticación.



```
draft_of_file_.pypirc
1 [distutils]
2 index-server =
3     pypi
4     testpypi
5
6 [pypi]
7 repository = https://upload.pypi.org/legacy/ # verify
8 username = josepedro # pypi user name
9
10 [testpypi]
11 repository = https://test.pypi.org/legacy/ # verify
12 username = josepedro # testpypi user name
```

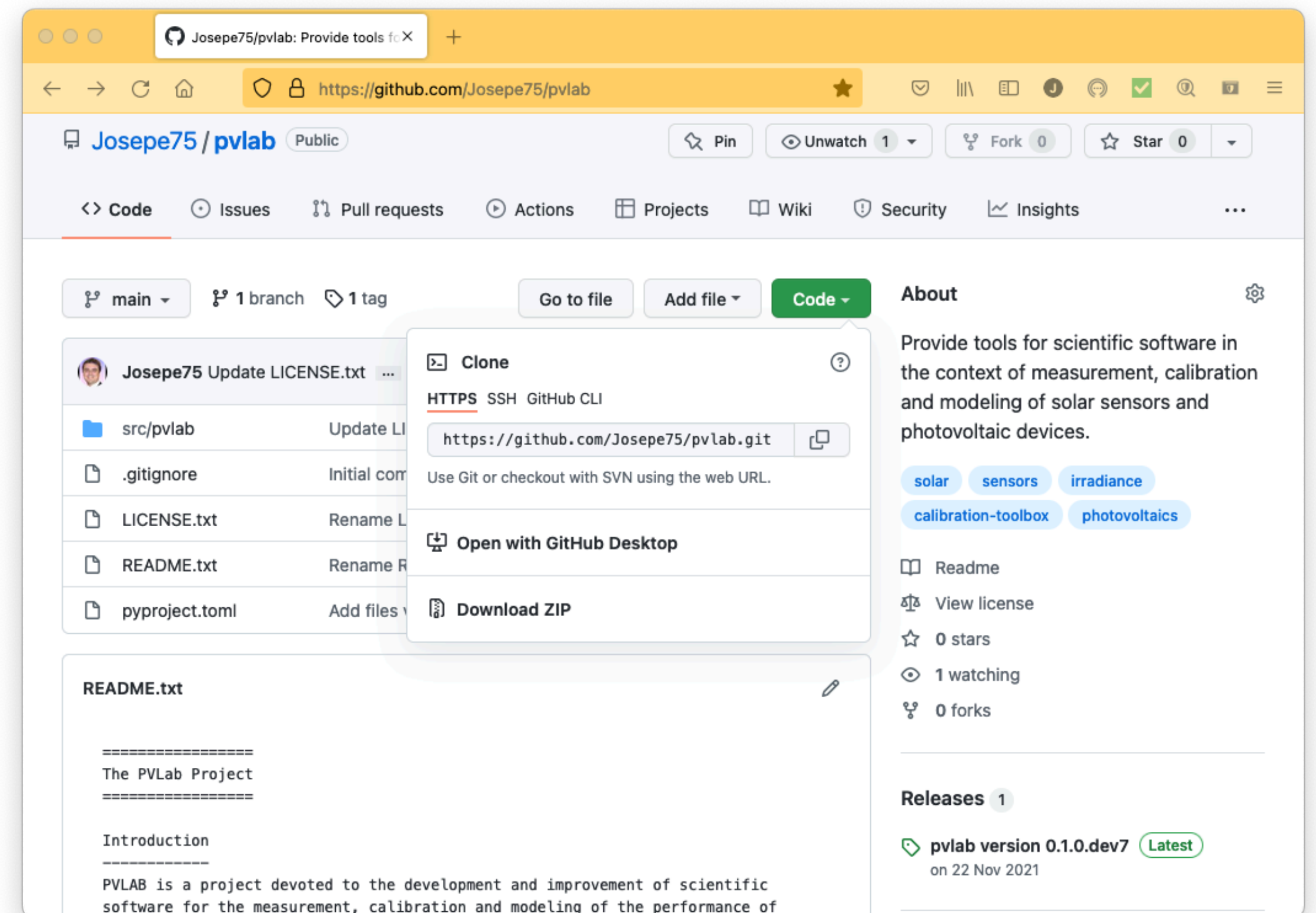
Recursos para el Desarrollo de Software

Distribución y Publicación

Publicación en *GitHub*

Los archivos y directorios contenidos en la "source distribution" pueden ser directamente subidos a *GitHub*, con las siguientes finalidades:

- A. Gestionar el desarrollo del proyecto de software, introduciendo los cambios de la nueva versión.
- B. Disponer de copias seguras del proyecto, en cualquier rama de desarrollo actual (main por defecto) y en cualquier momento de desarrollo del mismo.
- C. Servir de plataforma para la publicación de una entrega (release) del proyecto en repositorios de acceso abierto, obteniendo así una referencia de la autoría (p.ej. el DOI en el repositorio Zenodo).
- D. Servir de plataforma para la publicación de la documentación del proyecto en plataformas web documentales, por ejemplo *ReadtheDocs*.



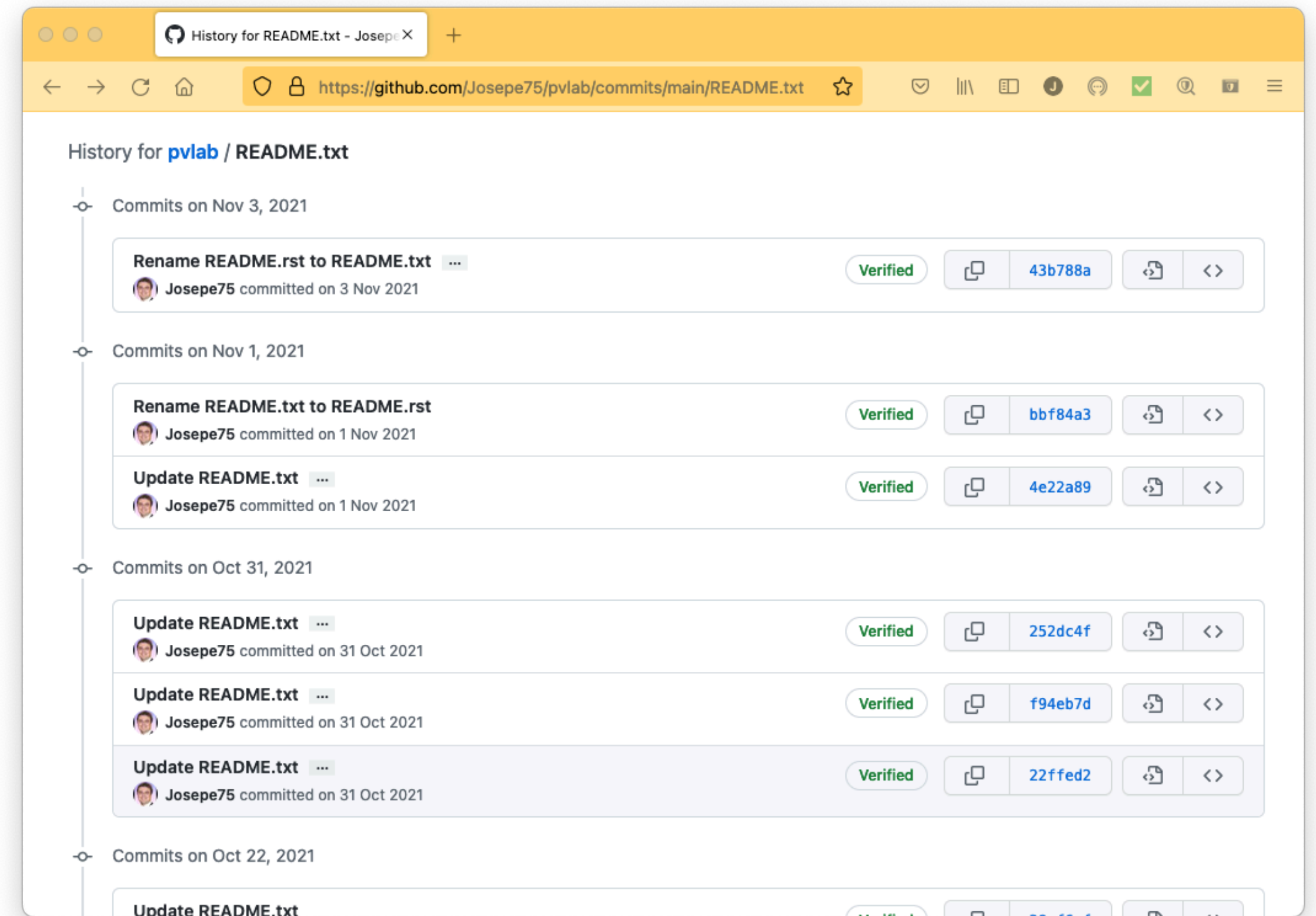
Recursos para el Desarrollo de Software

Distribución y Publicación

Publicación en *GitHub*

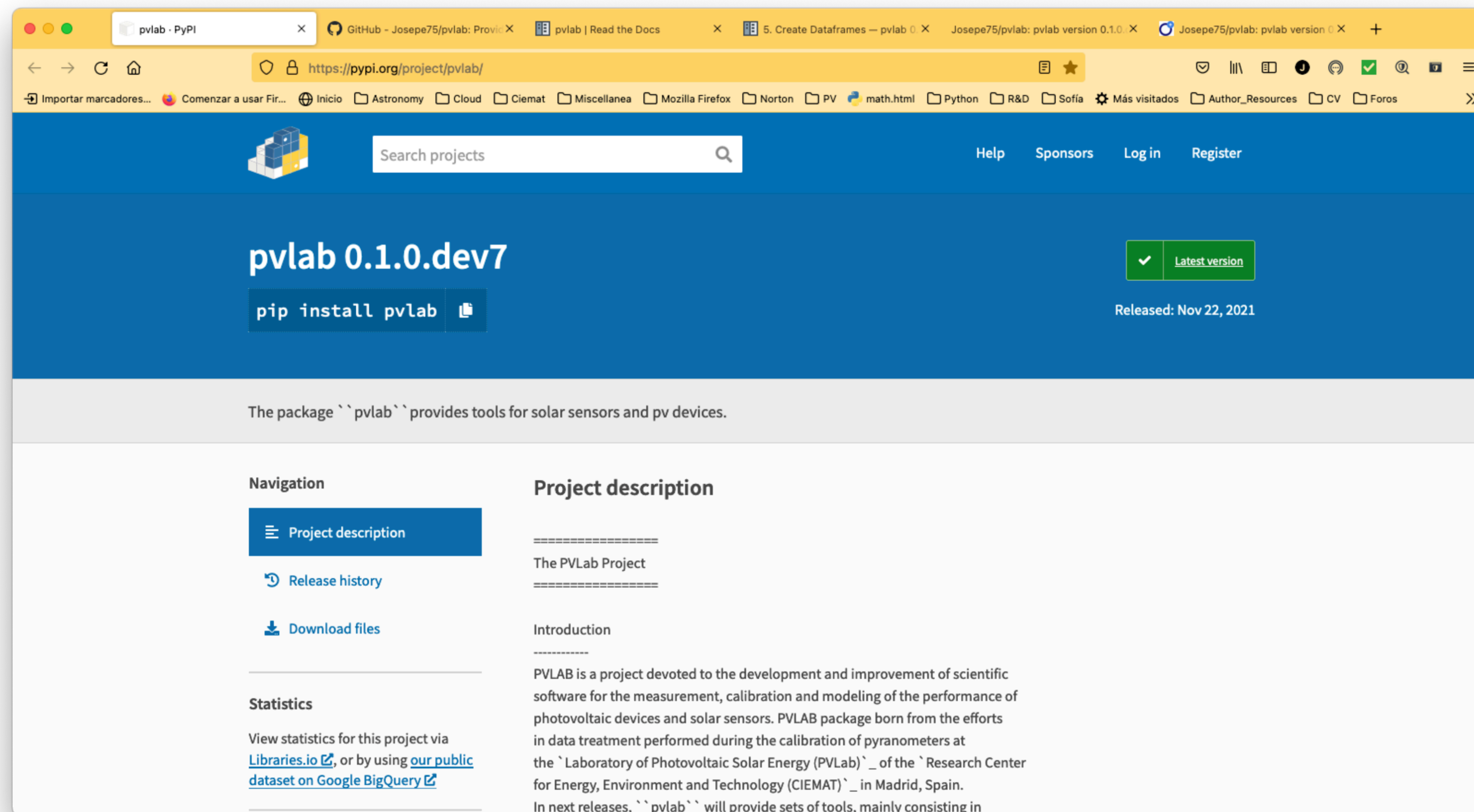
Los archivos y directorios contenidos en la "source distribution" pueden ser directamente subidos a *GitHub*, con las siguientes finalidades:

- A. Gestionar el desarrollo del proyecto de software, introduciendo los cambios de la nueva versión.
- B. Disponer de copias seguras del proyecto, en cualquier rama de desarrollo actual (main por defecto) y en cualquier momento de desarrollo del mismo.
- C. Servir de plataforma para la publicación de una entrega (release) del proyecto en repositorios de acceso abierto, obteniendo así una referencia de la autoría (p.ej. el DOI en el repositorio Zenodo).
- D. Servir de plataforma para la publicación de la documentación del proyecto en plataformas web documentales, por ejemplo ReadtheDocs.



Recursos para el Desarrollo de Software

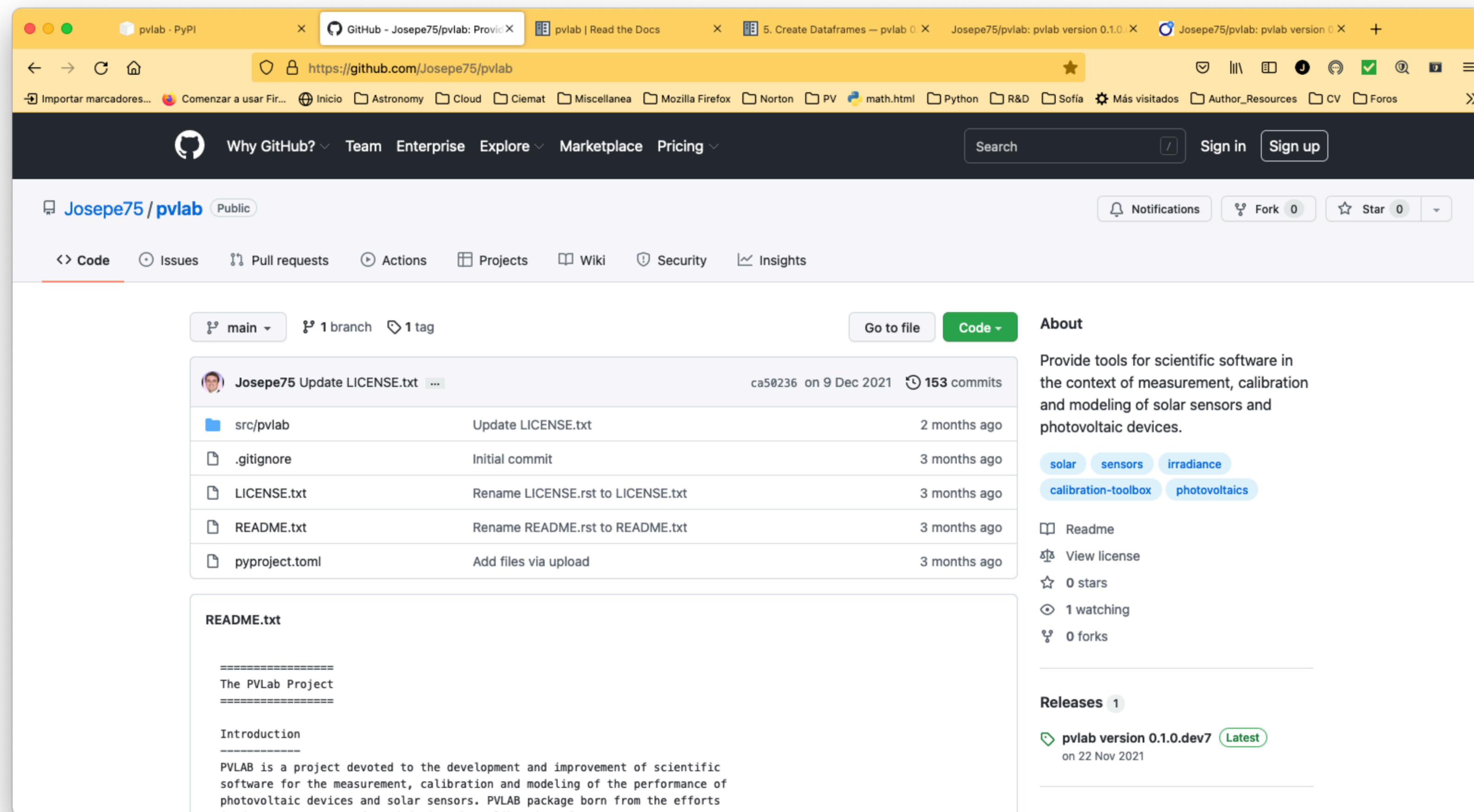
Distribución y Publicación



Ubicación del Proyecto pvlab en el *Python Package Index*.

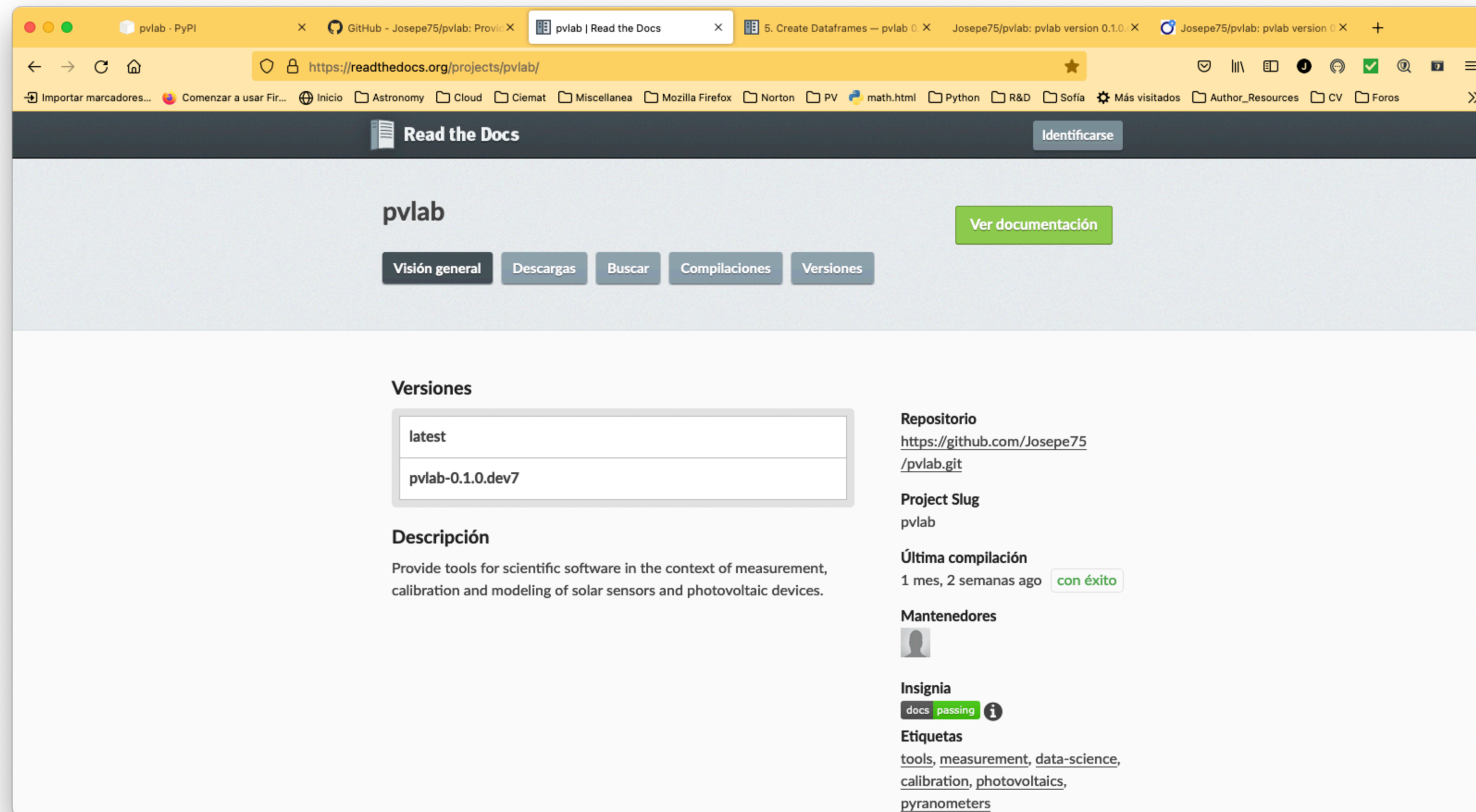
Recursos para el Desarrollo de Software

Distribución y Publicación



Recursos para el Desarrollo de Software

Distribución y Publicación



Documentación del proyecto pvlab en Read the Docs.

Recursos para el Desarrollo de Software

Distribución y Publicación

The screenshot shows a web browser window with multiple tabs. The active tab is titled '5. Create Dataframes — pvlab 0.1.0'. The address bar shows the URL 'https://pvlab.readthedocs.io/en/latest/usage/en/tools_dataframes.html'. The page content is divided into a left sidebar and a main content area. The sidebar includes a 'pvlab' logo, a 'Navigation' section with a 'Table of Contents' list (License, The PVLab Project, Release History, Installation, Create Dataframes, Math operations, I/O Management), a 'Quick search' bar, and an advertisement for 'EthicalAds'. The main content area is titled '5. Create Dataframes' and contains the following text: 'Provide tools to manage dataframes in the context of calibration. It contains the following python modules: DataFrames from dicts'. Below this, it states 'Perform type-conversion and *pretty-print* operations for dictionaries. It contains the following functions: Function dict_to_df'. The 'Function dict_to_df' section includes the signature 'pvlab.dataframes.dfdicts.dict_to_df(dictionary: dict, columns: list) -> pandas.core.frame.DataFrame' and a description: 'Re-arrange a dictionary to become a pandas dataframe. It performs a type conversion of a dictionary (e.g. a dictionary that represents some kind of valid time intervals), returning a pandas.DataFrame.' It also provides 'Code examples' and states 'When correct parameters are provided, it returns a pandas.DataFrame object:'. An 'Example 1' shows the correct use of the function 'pvlab.dataframes.dfdicts'. A code block contains the following Python code: '>>> from pvlab.dataframes.dfdicts import dict_to_df', '>>> dates = {\'START_1\': (2021,5,5,8,1,0), \'END_1\': (2021,5,6,22,52,0)}', and '>>> columns = [\'%Y\', \'%m\', \'%d\', \'%H\', \'%M\', \'%S\']'. A 'v: latest' dropdown menu is visible in the bottom right corner of the code block.

pvlab

Navigation

Table of Contents

1. License
2. The PVLab Project
3. Release History
4. Installation
5. Create Dataframes
 - DataFrames from dicts
 - Function dict_to_df
 - Function print_dict
6. Math operations
7. I/O Management

Quick search

Go

5. Create Dataframes

Provide tools to manage dataframes in the context of calibration.

It contains the following python modules:

DataFrames from dicts

Perform type-conversion and *pretty-print* operations for dictionaries.

It contains the following functions:

Function dict_to_df

`pvlab.dataframes.dfdicts.dict_to_df(dictionary: dict, columns: list) -> pandas.core.frame.DataFrame`

Re-arrange a dictionary to become a pandas dataframe. It performs a type conversion of a dictionary (e.g. a dictionary that represents some kind of valid time intervals), returning a pandas.DataFrame.

Code examples:

When correct parameters are provided, it **returns** a `pandas.DataFrame` object:

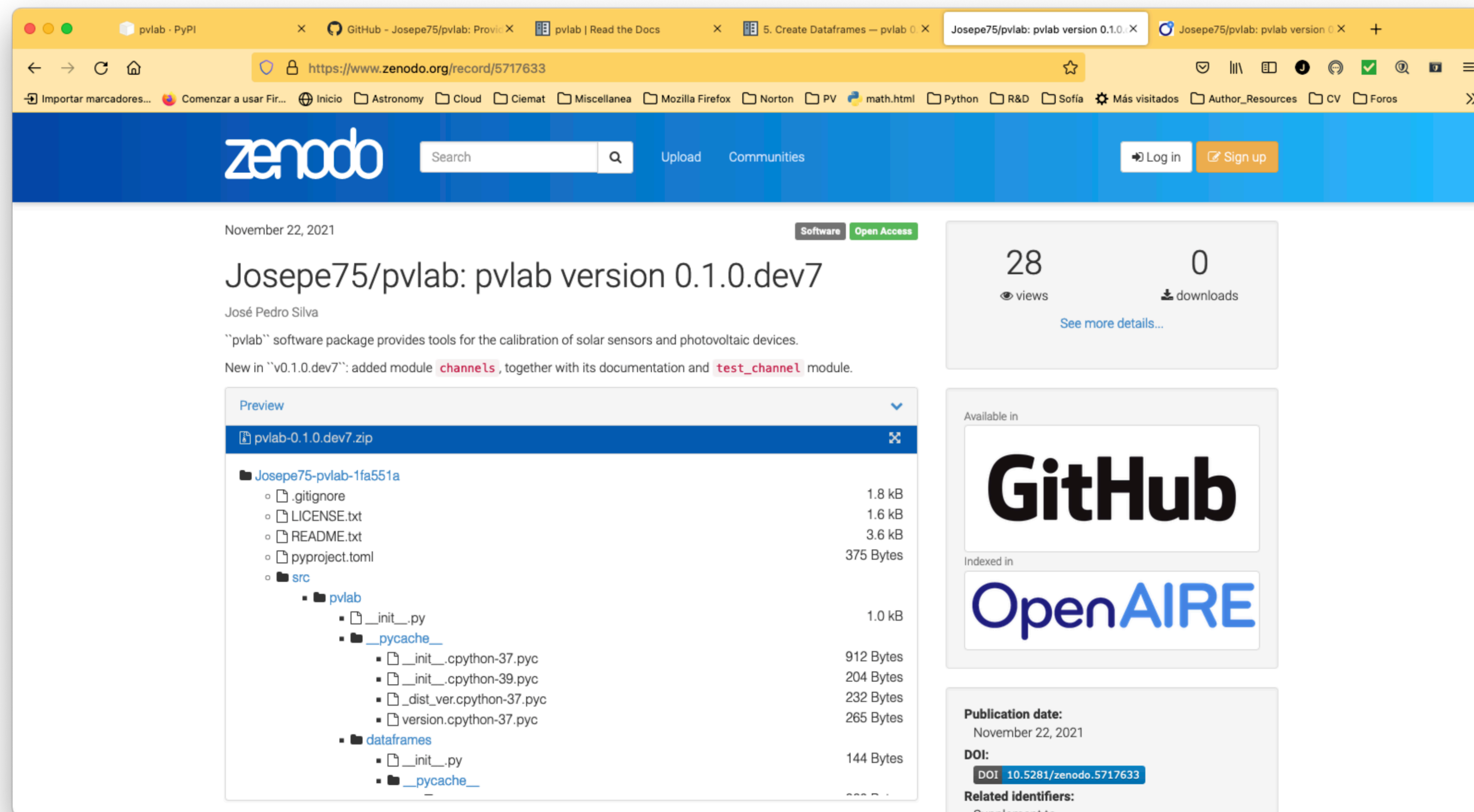
Example 1: correct use of function `pvlab.dataframes.dfdicts`.

```
>>> from pvlab.dataframes.dfdicts import dict_to_df
>>> dates = {'START_1': (2021,5,5,8,1,0), 'END_1': (2021,5,6,22,52,0)}
>>> columns = ['%Y', '%m', '%d', '%H', '%M', '%S']
```

v: latest

Recursos para el Desarrollo de Software

Distribución y Publicación



Ubicación de pvlab en el repositorio Zenodo.

Recursos para el Desarrollo de Software

Distribución y Publicación

The screenshot shows a web browser window with multiple tabs open, including 'pvlab · PyPI', 'GitHub - Josepe75/pvlab: Provide...', 'pvlab | Read the Docs', '5. Create Dataframes — pvlab', and 'Josepe75/pvlab: pvlab version 0.1.0.'. The address bar shows the URL 'https://explore.openaire.eu/search/software?pid=10.5281%2Fzenodo.5717633'. The browser's bookmark bar contains various folders like 'Astronomy', 'Cloud', 'Ciemat', 'Miscellanea', 'Mozilla Firefox', 'Norton', 'PV', 'math.html', 'Python', 'R&D', 'Sofia', 'Más visitados', 'Author_Resources', 'CV', and 'Foros'. The OpenAIRE Explore page displays the software 'Josepe75/pvlab: pvlab version 0.1.0.dev7' by 'Silva, José Pedro;'. It includes buttons for 'OPEN ACCESS' and 'ENGLISH', and states 'Published: 22 Nov 2021' and 'Publisher: Zenodo'. A sidebar on the right offers options: 'LINK THIS SOFTWARE TO...', 'CITE THIS SOFTWARE', 'ADD TO ORCID', and 'ADD ANNOTATION'. The 'SUMMARY' section contains an abstract: '``pvlab`` software package provides tools for the calibration of solar sensors and photovoltaic devices. New in ``v0.1.0.dev7``: added module channels, together with its documentation and test_channel module.' Below the abstract are 'Persistent Identifiers' with DOI links: 'DOI: 10.5281/zenodo.5717632' and '10.5281/zenodo.5717633'. The 'Download from' section lists two links: 'https://dx.doi.org/10.5281/zen...' and 'https://dx.doi.org/10.5281/zen...', both pointing to 'Software . 2021' from 'Provider: Datacite'. A link 'View all 2 versions' is also present.

Indexado del proyecto pvlab en OpenAire.

Recursos para el Desarrollo de Software

Distribución y Publicación

El salto a la publicación en abierto...

1. Mejora de código —> **módulos python (.py)**

- a) Estructura, claridad (*type hints*), PEPs
- b) Funcionalidad, captura de excepciones
- c) Descripción, ejemplos
- d) Test de ejemplos (p.ej. *doctest*)
- e) Test de código (p.ej. *unittest*)

2. Proy. documental —> **documentación (.tex, .pdf, ...)**

- a) Aplicación (p.ej. *sphinx*)
- b) Configuración proyecto (*conf.py*, *index.rst*)
- c) README, LICENSE, TOC, archivos *.rst*, *.md*
- d) Compilación, tests

3. Distribución —> **paquetes software (.tar, .whl)**

- a) Aplicación, configuración
- b) Generación *source* y *wheel* distributions
- c) Tests instalación local (*from src*)

4. Publicación (**PyPI**) —> **Proyecto en www.pypi.org**

- a) Aplicación (p.ej. *twine*, *flit*, ...)
- b) Registro (credenciales *keyring*), configuración
- c) Prueba de **instalación** (p.ej. *pip*, *conda*, etc.)

5. Publicación (**GitHub**) —> **Proyecto en www.github.com**

- a) Registro en **Servidor Control Versión** (*Git*, *GitHub*, ...)
- b) Subir la *source distribution*

6. Publicación (**Zenodo**) —> **Proyecto en www.zenodo.org**

- a) Generación de una entrega (*release*) en GitHub.
- b) Publicación en **repositorio abierto e indexado (DOI)**.

7. Publicación (**ReadtheDocs**) —> **Doc en www.readthedocs.org**

- a) Registro, enlace al proyecto público GitHub.
- b) Compilación proyecto documental
- c) **Publicación de la documentación** en internet.
- d) Pruebas, recompilaciones

Apéndices

Enlaces de Interés

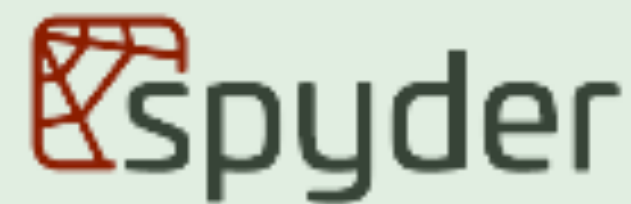
Recursos para la programación científica

Some built-in and python libraries

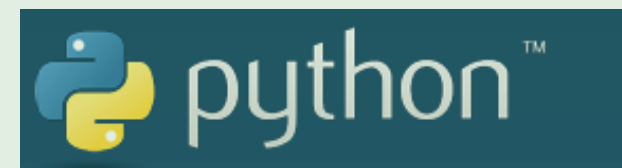
Some third-party libraries



Environment Anaconda



IDE Spyder



Python Programming Language



Jupyter Notebooks

...



doctest - Test interactive Python examples



math - Mathematical functions



pip - Installing Packages



typing - Support for type hints



re - Regular expression operations



unittest - Unit testing framework



venv - Creating Virtual Environments



Numerical Computing Tools



Algorithms for scientific computing



Pandas Data analysis



Document Generator Sphinx

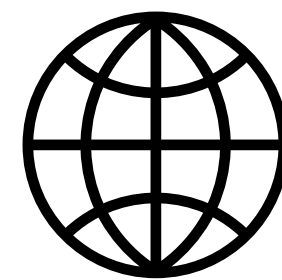
Enlaces de Interés

Recursos para la distribución

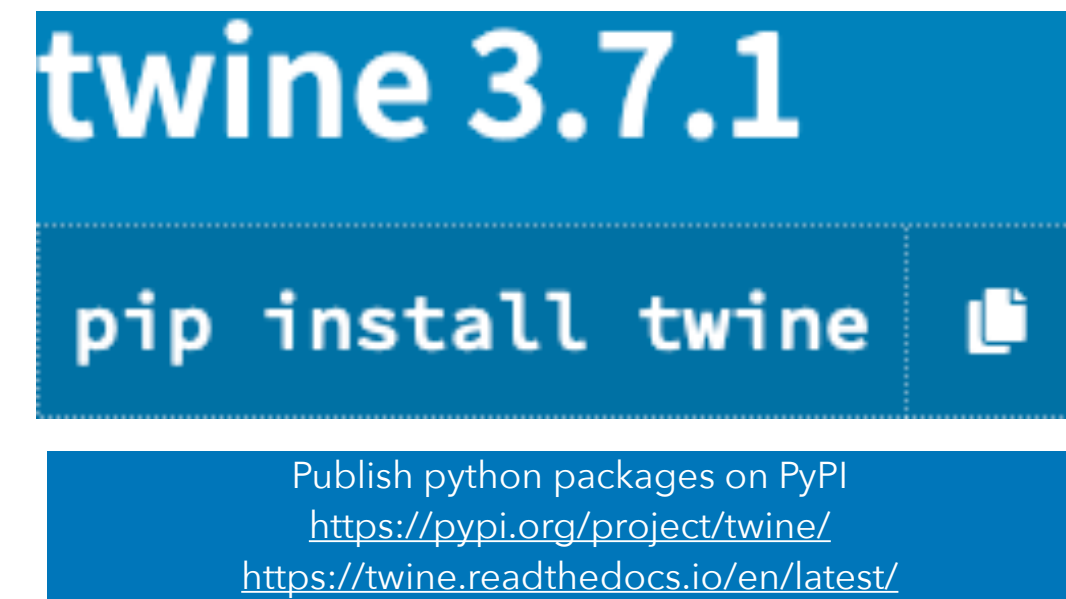


Enlaces de Interés

Recursos para la distribución

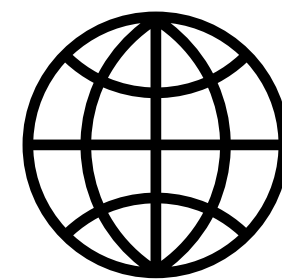


List Classifiers

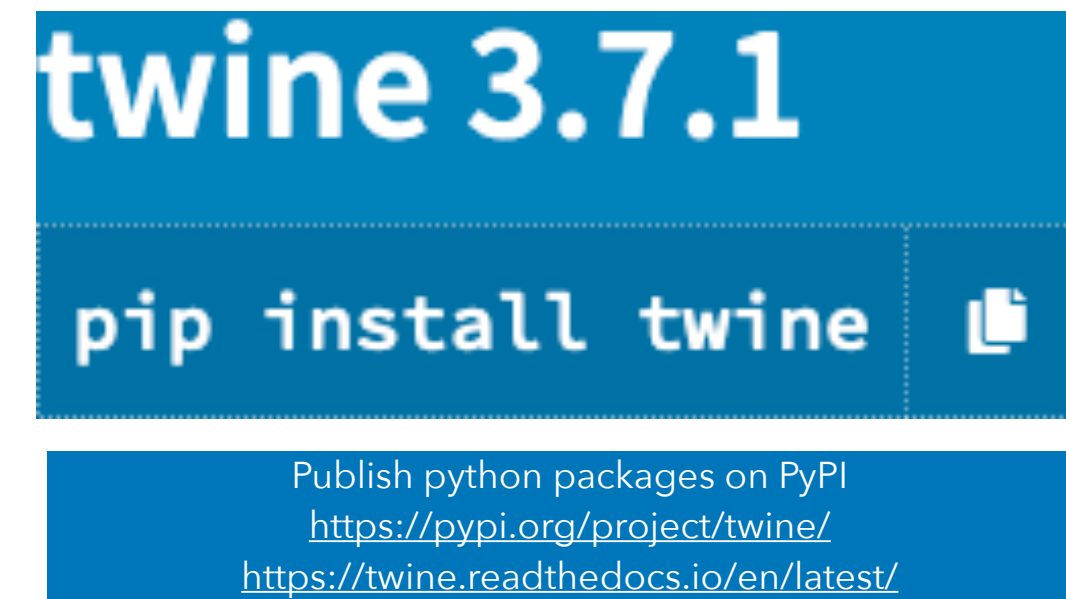


Enlaces de Interés

Recursos para la distribución

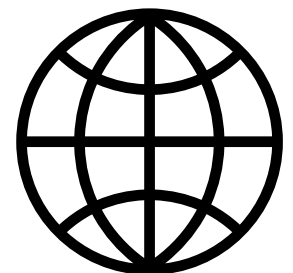


List Classifiers

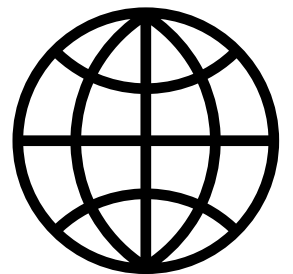


Enlaces de Interés

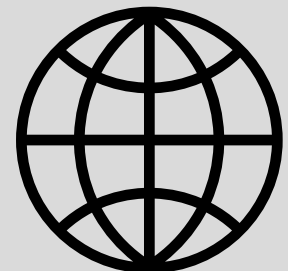
Recursos para la documentación y publicación



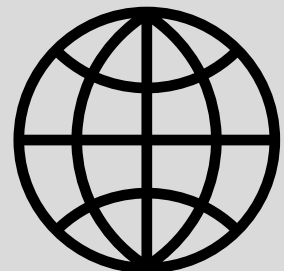
reStructuredText Markup Specification



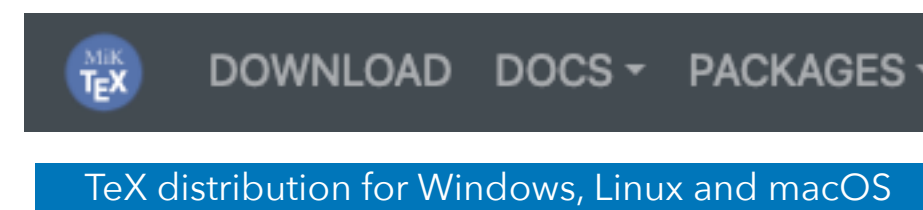
reStructuredText Directives



UTF - 8 encoding



List Classifiers



Markdown markup language

Enlaces de Interés

Repositorios



ZENODO: desarrollado por CERN Data Centre & Invenio.



GitHub: Repositorio/Plataforma de Desarrollo de Software (**CVS:** *Control Version Software*).



PYPI: Repositorio original paquetes instalables python (Python Package Index).



OpenAire: Indexa contenidos (p.ej. software de zenodo) y relaciona con Proyectos Europeos, comunidades y resultados de investigación (publicaciones).



ReadtheDocs: Genera y aloja la documentación de un proyecto a partir del contenido alojado en GitHub.

Enlaces de Interés

Repositorios



ZENODO: desarrollado por CERN Data Centre & Invenio.



GitHub: Repositorio/Plataforma de Desarrollo de Software (**CVS:** *Control Version Software*).



PYPI: Repositorio original paquetes instalables python (Python Package Index).



OpenAire: Indexa contenidos (p.ej. software de zenodo) y relaciona con Proyectos Europeos, comunidades y resultados de investigación (publicaciones).



ReadtheDocs: Genera y aloja la documentación de un proyecto a partir del contenido alojado en GitHub.

Enlaces de Interés

Licencias



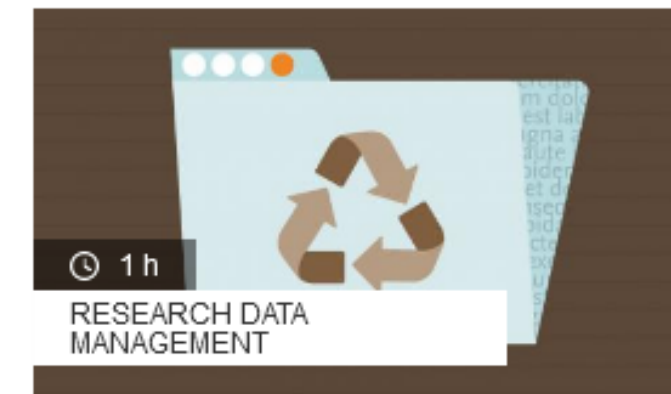
Licencias: *Open Source Initiative.*



Noticia: Licencia adoptada por la Comisión Europea para la información que publica online.



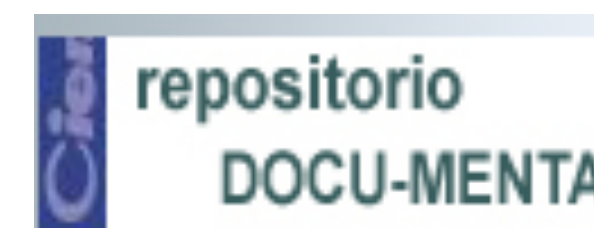
Licencias: *Creative Commons.*



Recursos: make your data reusable (Elsevier Research Academy)



Info: Política sobre acceso abierto Documenta - Ciemat.



Info: Licencias en Documenta - Ciemat (FAQ#8).



Licencia Creative Commons Attribution - NoCommercial - NoDerivatives

Enlaces de Interés

Licencias



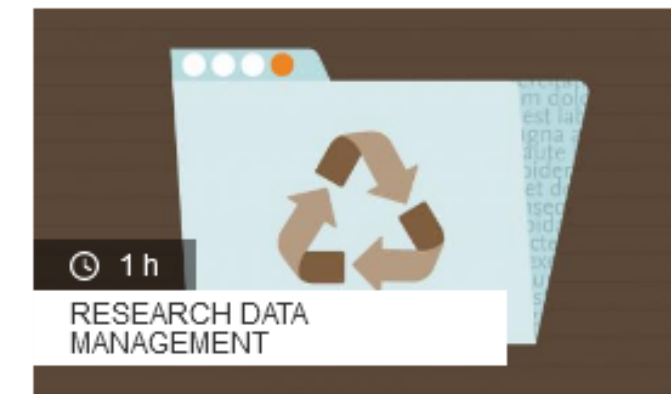
Licencias: *Open Source Initiative.*



Noticia: Licencia adoptada por la Comisión Europea para la información que publica online.



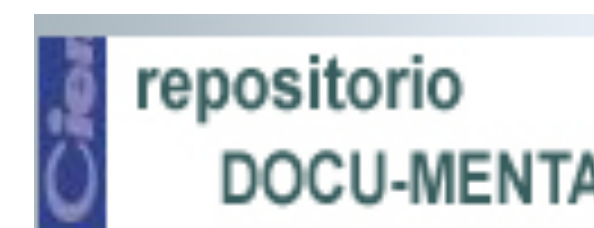
Licencias: *Creative Commons.*



Recursos: make your data reusable (Elsevier Research Academy)



Info: Política sobre acceso abierto Documenta - Ciemat.



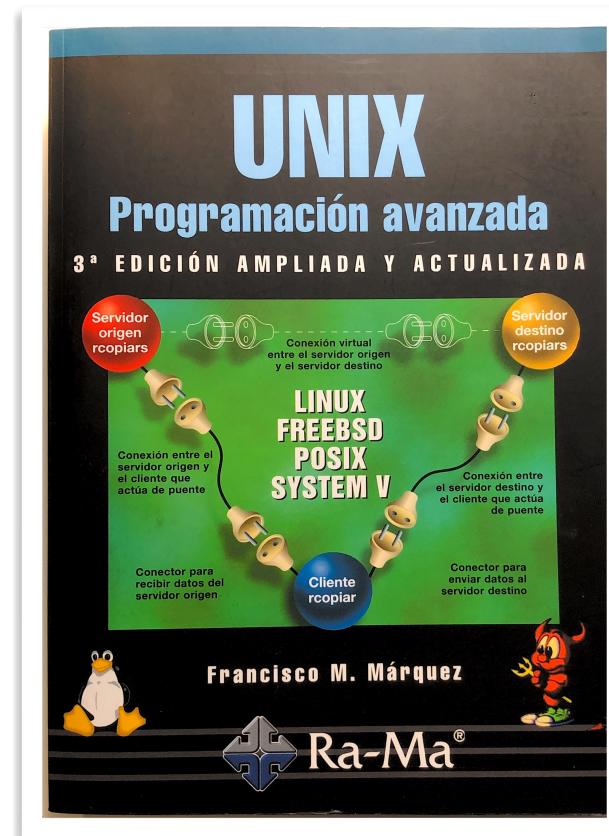
Info: Licencias en Documenta - Ciemat (FAQ#8).



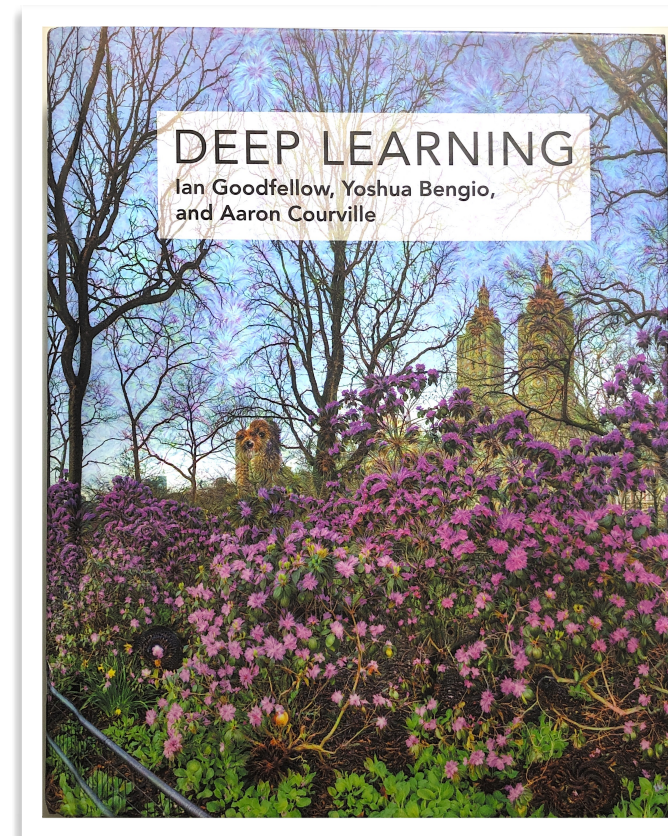
Licencia Creative Commons Attribution - NoCommercial - NoDerivatives

Bibliografía

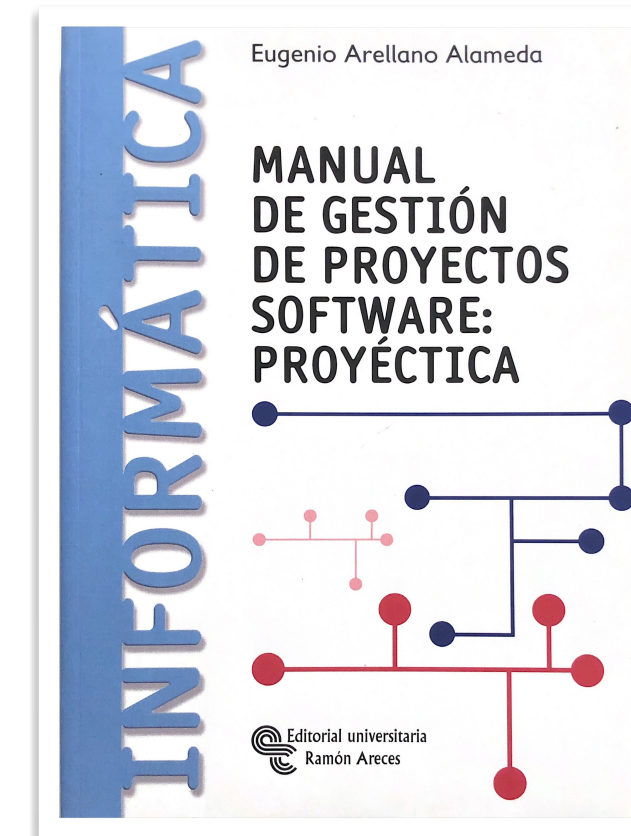
Algunos compañeros de viaje ...



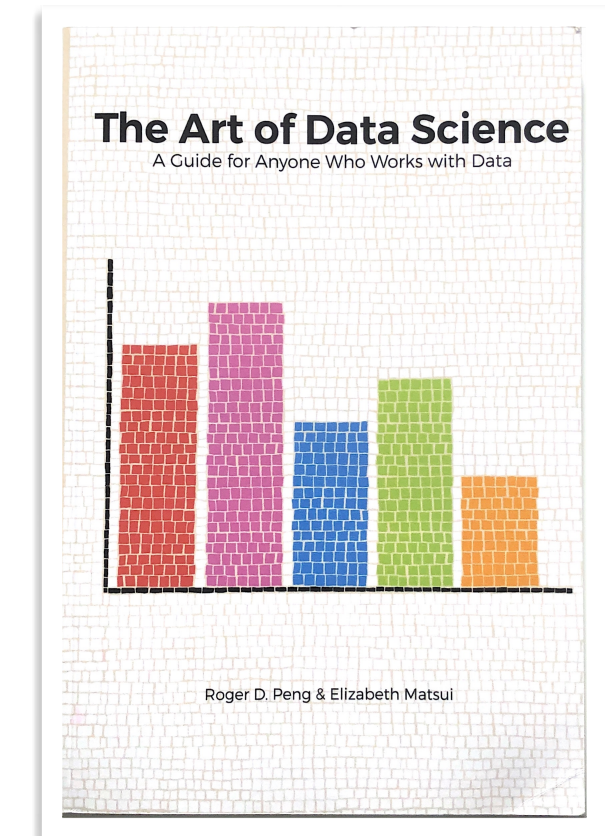
ISBN: 978-84-789-7603-4



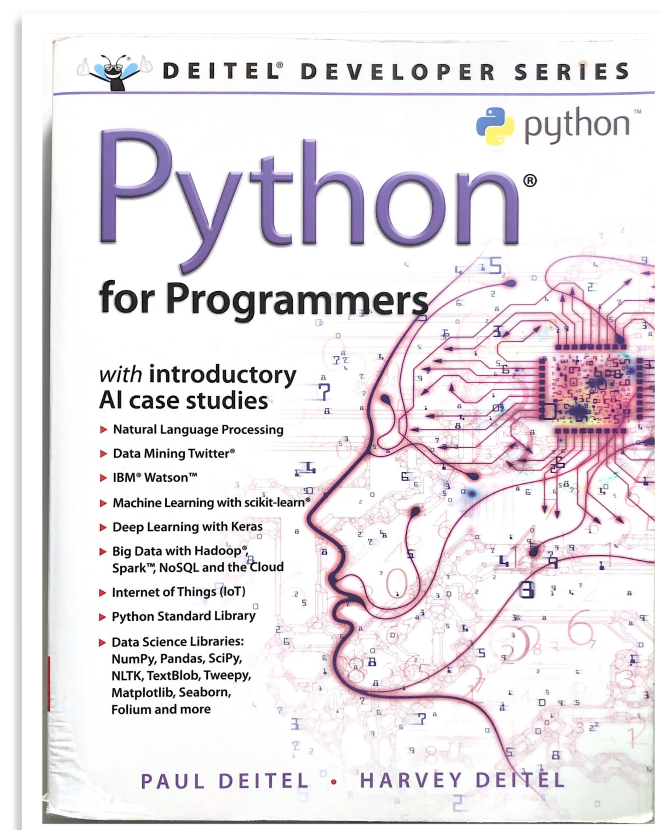
ISBN: 978-02-620-3561-3



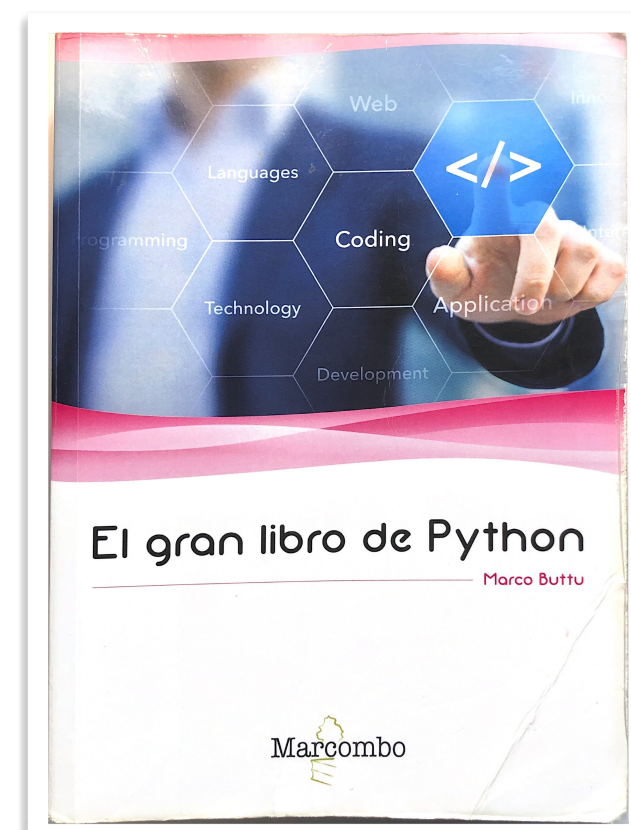
ISBN: 978-84-8004-762-3



ISBN: 978-1-365-06146-2



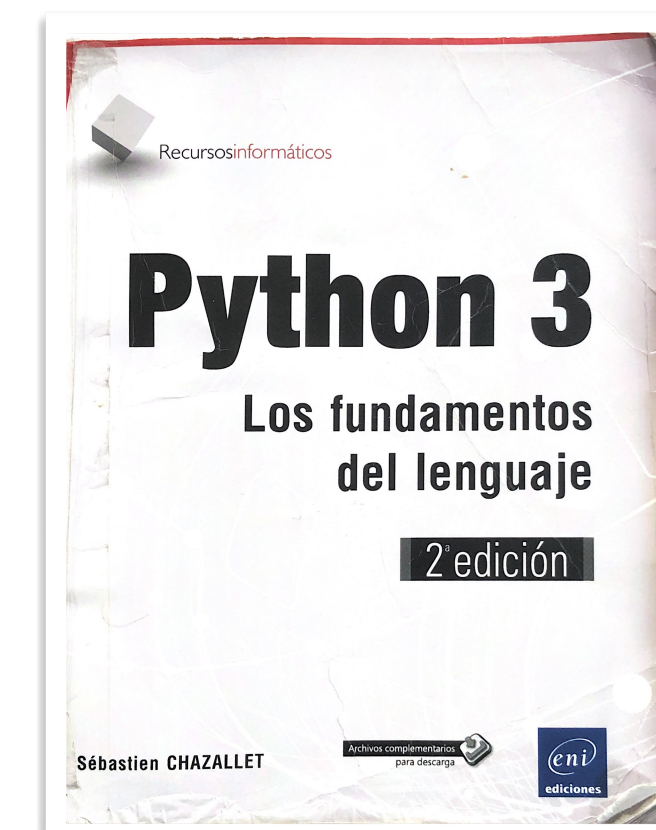
ISBN: 978-0-13-522433-5



ISBN: 978-84-267-2290-4



ISBN: 978-84-415-2513-9



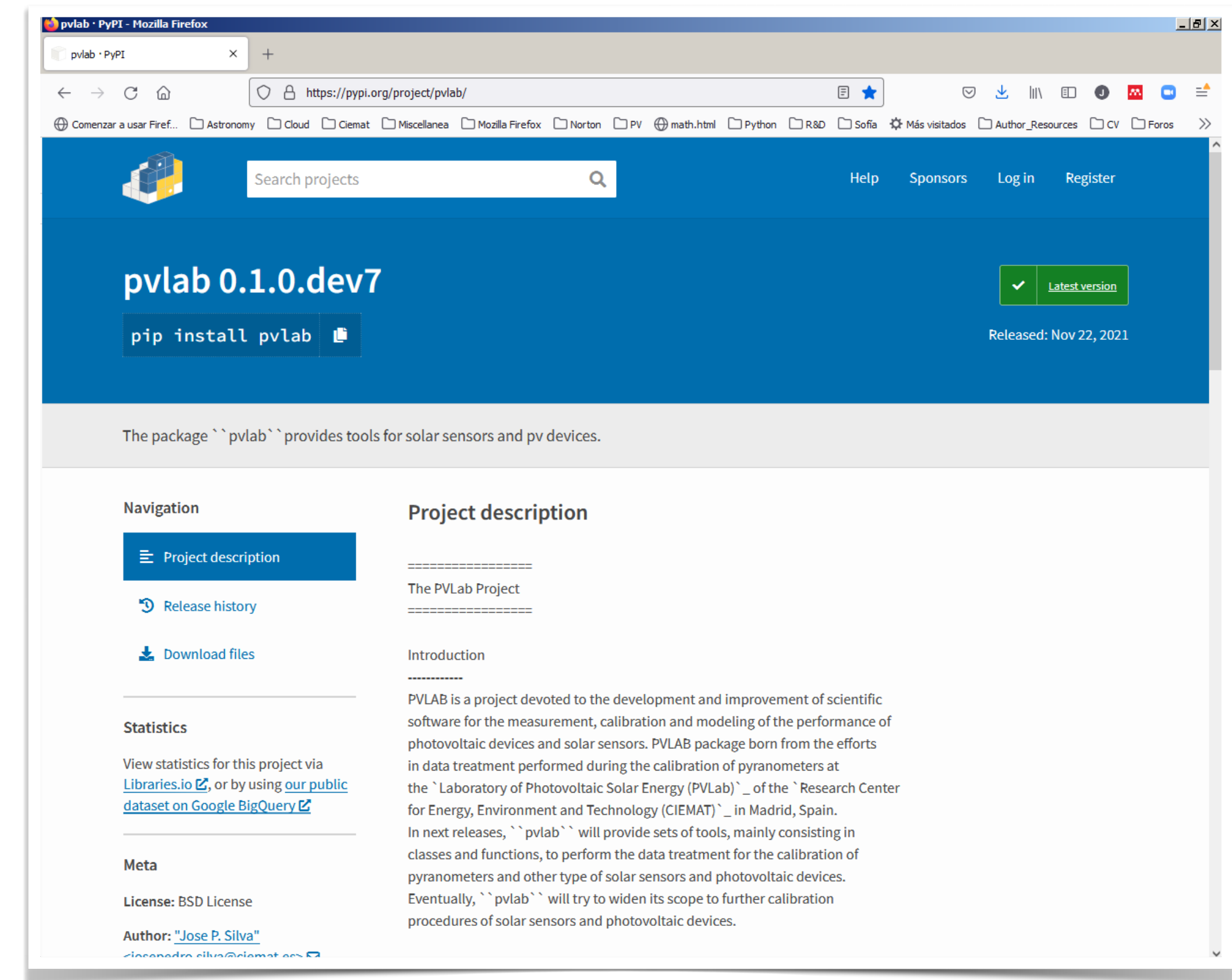
ISBN: 978-2-409-00614-2

Información Adicional

Aspectos por los que ha habido que pasar, pero inabordables en una primera exposición...

- Análisis del **modelo de objetos, estructuras y algoritmos** en *python*.
- Uso de **librerías de cálculo numérico** (p.ej. *numpy*, *pandas*).
- Aspectos de la **elaboración del proyecto documental**, como los lenguajes de tipado de texto (.rst, .md, ...).
- Gestión de **dependencias** (Instalación/desinstalación/actualización) en diferentes entornos.
- Elaboración de **puntos de acceso a la aplicación** o *entry points* (p.ej. *console scripts*).
- Aspectos de la **distribución y publicación del software** (aplicaciones disponibles, criterios, opciones de configuración, registros, autenticación en plataformas y repositorios, etc.).
- Una revisión detallada de las **licencias de uso** (licencias CC y OSI - *approved*, fundamentalmente), así como del esquema *FAIR*.
- Una revisión detallada de las **publicaciones existentes** en el ámbito del desarrollo de software, sus índices de impacto ("*metrics*") y sus diferentes orientaciones ("*Aims and Scope*").
- Etc.

... y otras muchas herramientas a tratar en el futuro (ver el Capítulo ***Tendencias*** en la Parte III: ***Recursos Adicionales***).



The background of the image shows two jellyfish swimming in a deep blue environment. The jellyfish have translucent, bell-shaped bodies with a pinkish-purple hue and long, thin, trailing tentacles. The text 'GRACIAS POR SU ATENCIÓN' is superimposed in the center in a bold, orange-red font.

GRACIAS POR SU ATENCIÓN