

Javier Gómez Delgado  
Jesús García García-Doncel

# PROGRAMACIÓN EN PYTHON

## MÁS ALLÁ DEL CÓDIGO





# **Programación en Python**

Más allá del código

Madrid, 2025

Javier Gómez Delgado

Jesús García García-Doncel

# Programación en Python

Más allá del código



Septiembre, 2025

*Programación en Python: Más allá del código*

Javier Gómez Delgado y Jesús García García-Doncel

Todos los derechos reservados.

Cualquier forma de reproducción, distribución, comunicación pública o transformación de esta obra solo puede ser realizada con la autorización de sus titulares, salvo las excepciones previstas por la ley.

Diríjase a CEDRO (Centro Español de Derechos Reprográficos) si necesita fotocopiar o escanear algún fragmento de esta obra ([www.cedro.org](http://www.cedro.org)).

© 2025, Javier Gómez Delgado y Jesús García García-Doncel

© 2025, ESIC EDITORIAL

Avda. de Valdenigrales, s/n

28223 Pozuelo de Alarcón (Madrid)

Tel.: 91 452 41 00

[www.esic.edu/editorial](http://www.esic.edu/editorial)

@EsicEditorial

ISBN: 978-84-1192-187-9

Depósito Legal: M-16605-2025

Diseño de cubierta: Zita Moreno Puig

Maquetación: Balloon Comunicación

Lectura: Balloon Comunicación

Impresión: Gráficas Dehon

Un libro de



Impreso en España – *Printed in Spain*

*Este libro ha sido impreso con tinta ecológica y papel sostenible.*

# Índice

<b>Capítulo 1.</b>	Introducción y entorno. . . . .	13
1.1	Introducción . . . . .	15
1.2	Instalación. . . . .	16
1.3	Editores y entornos de Python . . . . .	16
<b>Capítulo 2.</b>	Variables, operadores y sentencias input y print. . . . .	23
2.1	Comentarios. . . . .	25
2.2	Variables . . . . .	25
2.3	Tipos de datos . . . . .	28
2.4	Constantes . . . . .	31
2.5	Valores infinitos . . . . .	31
2.6	Tipo NaN (Not a Number) . . . . .	32
2.7	Operaciones aritméticas . . . . .	33
2.8	Operaciones de asignación . . . . .	33
2.9	Operaciones de comparación . . . . .	33
2.10	Operaciones lógicas . . . . .	34
2.11	Operaciones de bit . . . . .	35
2.12	Operaciones de otros módulos . . . . .	36
2.13	Entrada de datos por teclado . . . . .	38
2.14	Función print . . . . .	39
2.15	Ejercicios . . . . .	40
<b>Capítulo 3.</b>	Sentencias de control y bucles . . . . .	43
3.1	Sentencia if/else/elif . . . . .	45
3.2	Operador ternario . . . . .	46
3.3	Match .. case . . . . .	46
3.4	While . . . . .	47

3.5	For .....	48
3.6	Range .....	49
3.7	Break y continue .....	49
3.8	Ejercicios .....	50
<b>Capítulo 4.</b>	<b>Colecciones .....</b>	<b>53</b>
4.1	Rangos .....	55
4.2	Listas .....	55
4.3	Tuplas .....	64
4.4	Conjuntos o set .....	67
4.5	Diccionario .....	69
4.6	Ejercicios .....	73
<b>Capítulo 5.</b>	<b>Cadenas y fechas .....</b>	<b>75</b>
5.1	Codificación .....	77
5.2	Comparaciones .....	78
5.3	Funciones de cadenas .....	78
5.4	Transformaciones .....	82
5.5	Funciones de búsqueda en cadenas .....	83
5.6	Manejo de fechas .....	85
5.7	Ejercicios .....	89
<b>Capítulo 6.</b>	<b>Funciones .....</b>	<b>91</b>
6.1	Definición .....	93
6.2	Tipos .....	94
6.3	Valor None .....	95
6.4	Funciones de argumentos variables .....	96
6.5	Lista o tupla como parámetro .....	97
6.6	Funciones recursivas .....	97
6.7	Alcance de las variables .....	98
6.8	Ejercicios .....	98
<b>Capítulo 7.</b>	<b>Clases .....</b>	<b>101</b>
7.1	Programación orientada a objetos (POO) .....	103
7.2	Clases, atributos .....	104
7.3	Métodos .....	108
7.4	Imprimir un objeto .....	110
7.5	Encapsulación .....	111
7.6	Herencia .....	113
7.7	Clases abstractas .....	116
7.8	Sobrecarga de operadores .....	117

7.9	Polimorfismo . . . . .	120
7.10	Módulos y clases . . . . .	121
7.11	Ejercicios . . . . .	122
<b>Capítulo 8.</b> Excepciones, archivos, XML y JSON . . . . .		125
8.1	Manejo de errores o excepciones . . . . .	127
8.2	Manejo de archivos de tipo texto . . . . .	132
8.3	XML . . . . .	134
8.4	JSON . . . . .	137
8.5	Ejercicios . . . . .	140
<b>Capítulo 9.</b> Bases de datos y archivos de Log. . . . .		143
9.1	Instalación y conexión . . . . .	145
9.2	Definición de la estructura de la BBDD . . . . .	147
9.3	Consultas . . . . .	148
9.4	Inserción, modificación y borrado . . . . .	152
9.5	Archivos de Log . . . . .	154
9.6	Ejercicios . . . . .	157
<b>Capítulo 10.</b> Interfaz de usuario con Tkinter . . . . .		159
10.1	Interfaz de usuario . . . . .	161
10.2	Creación de una ventana . . . . .	161
10.3	Creación de botones . . . . .	162
10.4	Entrada de datos . . . . .	164
10.5	Etiquetas . . . . .	167
10.6	Texto multilínea . . . . .	169
10.7	Colocación de elementos con pack() . . . . .	171
10.8	Colocación de elementos con grid() . . . . .	172
10.9	Colocación de elementos con place() . . . . .	175
10.10	Mensajes . . . . .	176
10.11	Menús . . . . .	178
10.12	Creación de pestañas (tabs) . . . . .	181
10.13	Scroll text . . . . .	183
10.14	Data list o combo box . . . . .	184
10.15	Imágenes . . . . .	185
10.16	Barra de progreso . . . . .	186
10.17	Ejercicios . . . . .	188
<b>Capítulo 11.</b> NumPy . . . . .		191
11.1	Introducción . . . . .	193
11.2	Instalación . . . . .	195

11.3	Creación de arrays . . . . .	195
11.4	Propiedades de los arrays . . . . .	200
11.5	Acceso y segmentación del contenido . . . . .	200
11.6	Modificación de arrays . . . . .	202
11.7	Copia y vista . . . . .	205
11.8	Manipulación de dimensiones . . . . .	205
11.9	Combinación de arrays . . . . .	206
11.10	Operaciones aritméticas, lógicas, ordenaciones y filtrado . . . . .	208
11.11	Funciones estadísticas . . . . .	213
11.12	Álgebra lineal . . . . .	218
11.13	Ejercicios . . . . .	219

## Capítulo 12. Pandas . . . . . 221

12.1	Introducción . . . . .	223
12.2	Instalación . . . . .	223
12.3	Creación de objeto Series . . . . .	224
12.4	Atributos y métodos del objeto Series . . . . .	225
12.5	Trabajando con objetos Series . . . . .	227
12.6	Creación de objetos DataFrames . . . . .	230
12.7	Atributos y métodos del objeto DataFrame . . . . .	232
12.8	Empezando con los DataFrames . . . . .	233
12.9	Recorrer la estructura de un DataFrame . . . . .	237
12.10	Consultas de los elementos de un DataFrame . . . . .	243
12.11	Tratamiento de datos . . . . .	245
12.12	Combinar DataFrames . . . . .	249
12.13	Importar y exportar DataFrames . . . . .	250
12.14	Ejercicios con Series . . . . .	253
12.15	Ejercicios con DataFrame . . . . .	254

## Capítulo 13. Matplotlib . . . . . 255

13.1	Introducción . . . . .	257
13.2	Instalación . . . . .	257
13.3	Dibujando el primer gráfico . . . . .	258
13.4	Dando formato al gráfico . . . . .	259
13.5	Tipos básicos de gráficos . . . . .	261
13.6	Más opciones de formato . . . . .	263
13.7	Dibujando varias series de datos . . . . .	265
13.8	Colocando una leyenda . . . . .	267
13.9	Dibujar subgráficos . . . . .	268
13.10	Más tipos de gráficos . . . . .	273
13.11	Guardar una imagen . . . . .	278
13.12	Ejercicios . . . . .	278



<b>Capítulo 14. Juegos y Pygame</b>	281
14.1 El desarrollo de videojuegos	283
14.2 Introducción a Pygame	283
14.3 Recursos gráficos	284
14.4 El bucle de videojuego	287
14.5 Implementación de la física básica del juego	288
14.6 Gestión de eventos	291
14.7 La clase Sprite	293
14.8 Gestión de colisiones	296
14.9 Uso de fuentes	298
14.10 Efectos de sonido	299
14.11 Caso práctico: construyendo un primer juego	301
14.12 Ejercicios	321
<b>Capítulo 15. APIs y Flask</b>	323
15.1 Llamadas a APIs	325
15.2 Introducción a Flask	327
15.3 Instalación y puesta en marcha	327
15.4 Información de Log	329
15.5 Routing y paso de parámetros	331
15.6 Insomnia y Postman	334
15.7 Templates	336
15.8 Redirecciones	337
15.9 Manejo de errores	338
15.10 JSON	339
15.11 Ejercicios	340
<b>Capítulo 16. Django</b>	343
16.1 Introducción	345
16.2 Instalación y puesta en marcha	346
16.3 Creación de una aplicación	347
16.4 Conexión a la BBDD	349
16.5 Caso práctico	351
16.6 Ejercicios	362
<b>Solucionario</b>	365
<b>Bibliografía</b>	365

# Introducción y entorno

- 1.1. Introducción
- 1.2. Instalación
- 1.3 Editores y entornos de Python

**Objetivos de aprendizaje:**

- Familiarizarse con los objetivos del lenguaje
- Conocer diferentes entornos de programación
- Poder elegir el IDE más adecuado

**Palabras clave:** IDLE, Visual Studio Code, PyCharm, Anaconda, Jupyter

## 1.1 Introducción

Podemos definir Python como un lenguaje de programación de alto nivel, interpretado, orientado a objetos, con semántica dinámica y de propósito general.

Su nombre viene por la afición de su creador (Guido van Rossum) por el grupo de humor británico Monty Python.

Este lenguaje nació a principios de los 90 con los siguientes objetivos:

- Debe ser fácil e intuitivo.
- De código abierto.
- Código comprensible como el inglés.
- Adecuado para cualquier tarea, permitiendo también desarrollos pequeños.

Hoy se considera ya un lenguaje maduro, con conceptos muy modernos, que ha recogido todas las ventajas de otros lenguajes anteriores.

Donde no llega Python es en la programación de bajo nivel, para la creación de controladores o motores gráficos, en donde se necesitan otros lenguajes capaces de comunicarse con dispositivos electrónicos.

Por ahora tampoco ha llegado a los dispositivos móviles, pero esto, posiblemente, se verá algún día.

Existen dos versiones principales de Python, llamadas Python 2 y Python 3. Son versiones incompatibles entre sí, por eso aún siguen existiendo ambas, ya que no es sencillo pasar una gran aplicación de Python 2 a Python 3. Para los nuevos proyectos siempre se debe de utilizar la última versión, Python 3.

### ¿Por qué utilizar Python?

Python es un lenguaje que tiene una sintaxis clara y cercana al lenguaje humano que permite utilizarse a través de varios paradigmas de programación, como orientación a objetos, programación funcional o programación procedural.

La comunidad de desarrolladores que hay detrás es muy activa, proporcionando gran cantidad de tutoriales y recursos. A su vez, proporcionan varias bibliotecas y *frameworks* utilizados para gran cantidad de necesidades.

También Python se puede utilizar en los tres sistemas operativos más extendidos (Windows, macOS y Linux) gracias a que es un lenguaje interpretado y con tipado dinámico.

Gracias a esa sintaxis sencilla, permite a los desarrolladores crear prototipos de forma rápida, siendo mucho más productivos, ya que se escriben menos líneas de código. A su vez, es fácilmente integrable con otros lenguajes como C, C++ o Java.

Por último, Python destaca por sus bibliotecas como PySpark y Dask para trabajar con grandes volúmenes de datos (*big data*) y otras que ayudan para el desarrollo en inteligencia artificial, ciencia de datos, automatización o desarrollo web.

Por otro lado, los puntos débiles son:

- Rendimiento: al ser interpretado, es más lento que lenguajes compilados como C++.
- Uso de memoria: consume más memoria que lenguajes como C.
- Móvil: hoy en día no hay buenas opciones para desarrollo en móviles.

## 1.2 Instalación

Python funciona en cualquier sistema operativo. En Linux es muy probable que ya esté instalado. Para comprobarlo, es tan sencillo como abrir una consola y escribir el comando **python**. El resultado será entrar en el **shell** de Python, en donde te permitirá escribir directamente las sentencias.

[www.python.org](http://www.python.org)  
Página oficial de  
Python

En la página oficial de Python ([python.org](http://python.org)), en la sección de **downloads**, se puede descargar la última versión de Python. El navegador ya conoce el sistema operativo y te redirige a la versión adecuada.

Un usuario de Windows se descargará el archivo **.exe** y ejecutará una instalación personalizada dejando todas las opciones por defecto, excepto la casilla de **Agregar Python 3.x a PATH**, que se tiene que seleccionar.

En macOS se debe descargar e instalar el archivo **.pkg** y, al igual que en Windows, realizar una instalación personalizada dejando todas las opciones por defecto.

Una vez que tenemos instalado Python, se va a necesitar una aplicación que nos permita editar, ejecutar y depurar los desarrollos realizados.

## 1.3 Editores y entornos de Python

### IDLE

**IDLE (Integrated Development and Learning Environment)** es un sencillo editor que se instala junto con la versión de Python, y que nos va a permitir escribir pequeños programas. El acceso a esta aplicación se encuentra entre las aplicaciones que tenemos instaladas en el sistema.

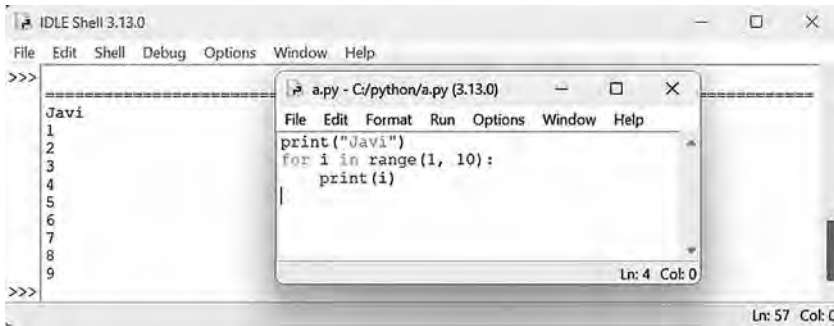


Figura 1.1  
IDLE

Lo primero que vemos es la consola. En la opción de **File**, podemos crear un nuevo archivo y ejecutarlo. El resultado de la ejecución lo veremos sobre la consola.

## Google Colab

**Colaboratory** de Google o más conocido como **Google Colab** (<https://colab.research.google.com/>) es un entorno gratuito de trabajo separado en cuadernos, siendo cada uno de ellos un programa diferente. Para trabajar con esta herramienta es necesario estar registrado en Google, y no hace falta tener instalado Python en nuestro ordenador. El código que nosotros colocamos sobre nuestro cuaderno se ejecutará sobre servidores de Google, a los cuales tenemos que conectarnos pulsando sobre el botón **Conectar** que aparece en la parte superior derecha de la web. Una vez conectado aparece el uso de la RAM y el disco del servidor al que estamos conectados.

En este momento ya podemos escribir código Python y ejecutarlo, y debajo veremos el resultado. Acercando el ratón debajo de la última ejecución se podrá añadir más código o texto, creando nuevas ejecuciones.

Todo este código se va a poder guardar como una copia en Drive o en GitHub para poder distribuirlo o reutilizarlo.

Al tener un servidor a nuestra disposición, también podemos guardar archivos en este para poder utilizarlos en nuestros programas. Para ello se pulsa sobre la carpeta en la barra de iconos de la izquierda.



Figura 1.2  
Google Colab

## Visual Studio Code

Este editor de código e IDE (entorno de desarrollo integrado) creado por Microsoft es el más extendido y utilizado por gran número de desarrolladores para diferentes lenguajes. Se puede descargar desde su página principal (<https://code.visualstudio.com/download>) para Windows, Linux y Mac.

Existe una extensión de Python para **Visual Studio Code** que proporciona indicaciones visuales, como código por colores y la función autocompletar, junto con herramientas de depuración que ayudarán a escribir código de Python mejor y más rápido.

Para instalarla, en **Visual Studio Code**, en la barra de menús, seleccionar Ver/Extensiones para abrir la vista Extensiones.

En esta vista se verán las extensiones instaladas y las extensiones recomendadas.

Escribir Python en el cuadro de búsqueda situado en la parte superior de la vista Extensiones y seleccionar la extensión Python publicada por Microsoft, descrita como **Python language suport...**, normalmente es la primera de la lista. Los detalles sobre esa extensión aparecen en un panel con pestañas a la derecha.

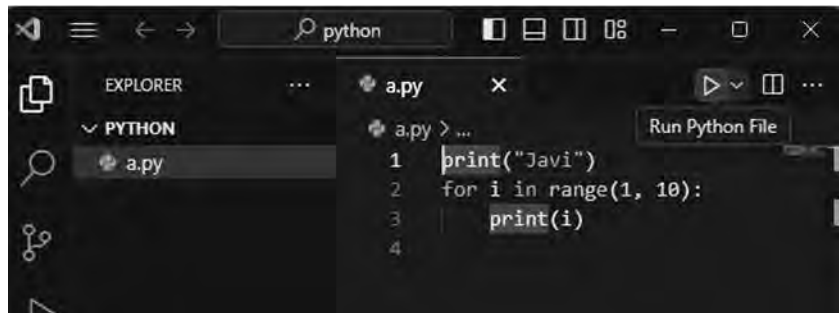
En el panel Extensiones o en el panel principal, seleccionar **Install**.

En este momento ya se podrá crear un nuevo archivo con extensión **.py** y ejecutarlo dentro de este entorno.

También podemos ejecutarlo desde la consola poniendo el comando:

```
py a.py
```

**Figura 1.3**  
Visual Studio Code

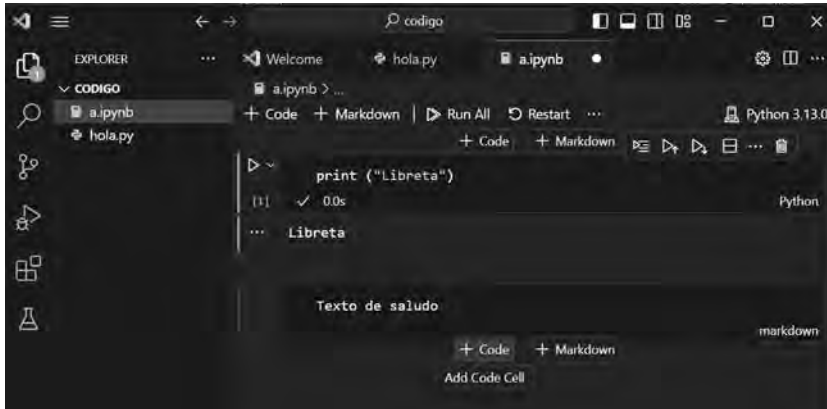


Sobre Visual Studio podemos trabajar con **Jupyter**. Esta es otra extensión que va a permitir trabajar con cuadernos (al igual que se hace con **Google Colab**).

Para trabajar con cuadernos, creamos un nuevo archivo con extensión **.ipynb**

Se verá que cambia la visualización y podremos añadir código y ejecutarlo, y a su vez también podemos añadir texto (**markdown**).

Cuando ejecutamos por primera vez, nos puede preguntar qué versión de Python utilizamos, si tenemos varias instaladas, y pide permiso para instalar un paquete que necesita para sus ejecuciones.



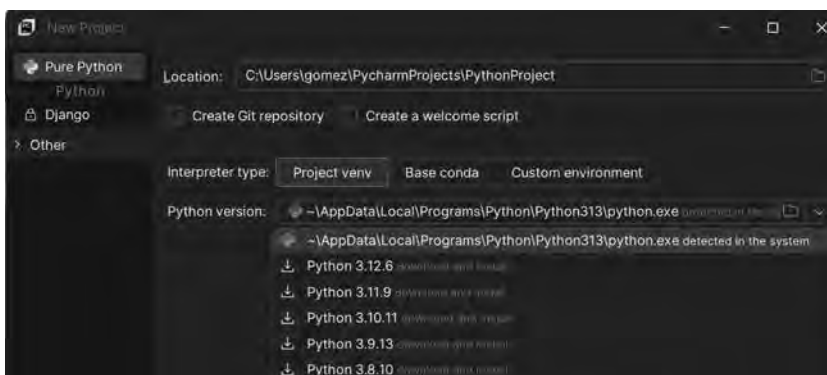
**Figura 1.4**  
Visual Studio con  
Jupyter

## PyCharm

**PyCharm** es un entorno de desarrollo integrado (IDE) de Python, desarrollado por **JetBrains**, que ofrece una amplia gama de herramientas para desarrolladores de Python, todas ellas integradas en el entorno y adaptadas para la creación de programas en Python, web y ciencia de datos. Tiene integración con distintas bases de datos, **Jupyter**, **Git**, **Conda**, **TensorFlow** y algunos más para el manejo de datos. También se integra con **Django**, **Flask** para las webs. Algunas de estas integraciones solo están disponibles en la versión de pago.

Se puede instalar la versión gratuita Pycharm Community descargándola de la siguiente dirección: <https://www.jetbrains.com/pycharm/download/>

Una vez instalada, se ejecuta y se selecciona la opción de crear un nuevo proyecto. En la siguiente pantalla se introduce el nombre del proyecto y se selecciona el intérprete de Python que utilizar, que puede ser el que tengamos ya instalado o cualquier otro para utilizar en este proyecto. En este último caso, lo descargará antes de abrir el nuevo proyecto.



**Figura 1.5**  
Nuevo proyecto en  
PyCharm

Una vez creado el proyecto, se puede crear un archivo Python. Para esto, se da clic derecho en el nombre del proyecto. Esto desplegará un menú contextual, se escoge **New** y **Python File**.

Por último, solicita el nombre del archivo y, a continuación, ya está disponible para escribir el código.

Dar clic en el icono de ejecución (triángulo verde en la parte superior); el resultado de esta ejecución se mostrará en una sección de **PyCharm** que se abre en la parte inferior de la pantalla.

**Figura 1.6**  
PyCharm



Cuando queremos trabajar con librerías que no vienen por defecto en el entorno, como **pandas** para el trabajo de datos, se tendrá que instalar. Para ello, desde la consola se utiliza el comando **pip** y el nombre de la librería que se quiere instalar para luego utilizarla.

```
pip install pandas
```

## Anaconda

Anaconda es una distribución de Python (no es necesario realizar una instalación previa de Python) que viene con las principales herramientas *open source* para realizar trabajos de *data scientist*.

Accediendo a la dirección <https://www.anaconda.com/download> se puede descargar e instalar Anaconda después de registrarse. Hay versiones para Windows, Mac y Linux.

Anaconda tiene incorporadas dos herramientas importantes, **Jupyter Notebook** como herramienta de *notebooks* y **Spyder** como IDE de programación.

El Notebook es una herramienta muy habitual en la ciencia de datos. **Jupyter Notebook** es una interfaz web de código abierto que permite la inclusión de texto, vídeo, audio, imágenes, así como la ejecución de código a través del navegador.

Esa capacidad de incluir código junto con imágenes y texto es lo que lo hace adecuado para el análisis de datos, pues permite llevar un hilo argumental a medida que se va llevando a cabo el estudio, la creación de modelos, la extracción de métricas, etc.

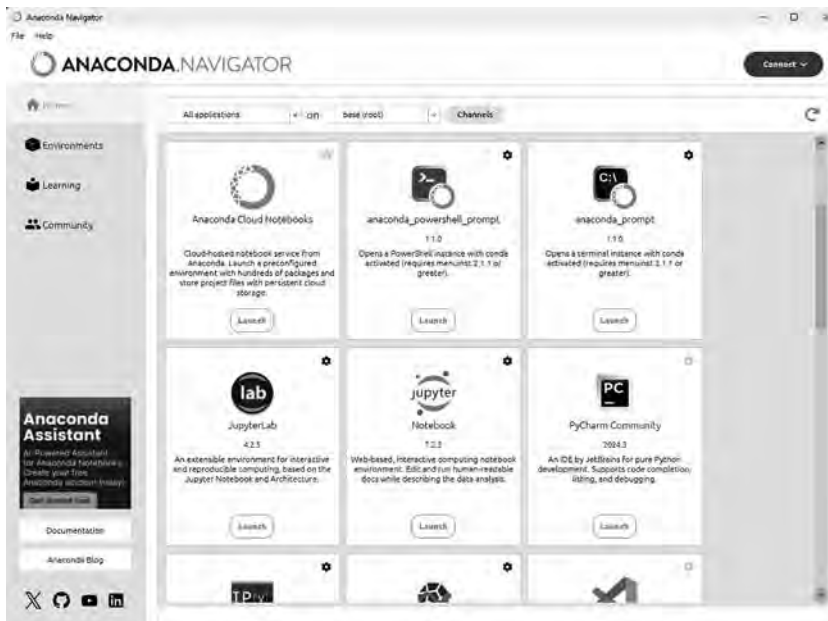
Sin embargo, los *notebooks* tienen una limitación: no permiten de modo fácil la puesta en producción de los programas. Para esto es mejor recurrir a un IDE como **Spyder**, más potente para la parte de desarrollo y menos para el análisis.



Los entornos en Anaconda, como también hace **PyCharm**, permiten tener diferentes proyectos con diferentes versiones de Python y librerías distintas. Lo ideal es tener un proyecto básico y, a partir de ahí, ir añadiendo las librerías necesarias al entorno.

Anaconda también incorpora **Conda**. Este es un gestor de paquetes que permite la creación de entornos y la instalación y actualización de las librerías.

Cuando se ejecuta Anaconda, lo primero que se ve es **Anaconda Navigator**. Aquí es en donde se ven todas las herramientas integradas con la plataforma.



**Figura 1.7**  
Anaconda Navigator

Para crear un entorno es necesario ir a la pestaña **Environments**, y hacer clic en **Create**. A continuación, instalar o actualizar los paquetes que se necesiten.

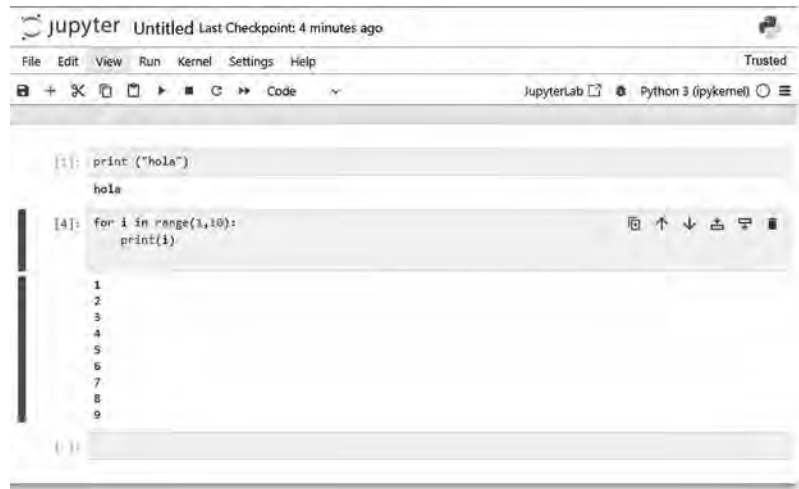
Una vez creado el entorno, se puede pulsar en el triángulo de la derecha y seleccionar la opción **Open Terminal**. De esta manera se accede a un terminal en donde si se escribe el comando **Python**, tenemos un modo de ejecución de lenguaje.

Para empezar a programar, se vuelve a la página de inicio y se lanza la ventana de **Jupyter Notebook**. Puede ser necesario instalarlo en el entorno, pulsando sobre la Home el botón **Install** dentro de **Jupyter Notebook**; a continuación, aparecerá el botón **Launch**.

Cuando arranca abre el navegador con nuestro sistema de ficheros posicionado en nuestro directorio personal. Con **new/folder** podemos crear una carpeta para meter el código. A su vez, se puede crear un nuevo archivo que creará un fichero con extensión **.ipynb**, propio de los cuadernos.

Una vez creado el documento, aparece lo que se denomina una celda, en donde se puede escribir código Python, guardándose ese código y su resultado dentro del cuaderno.

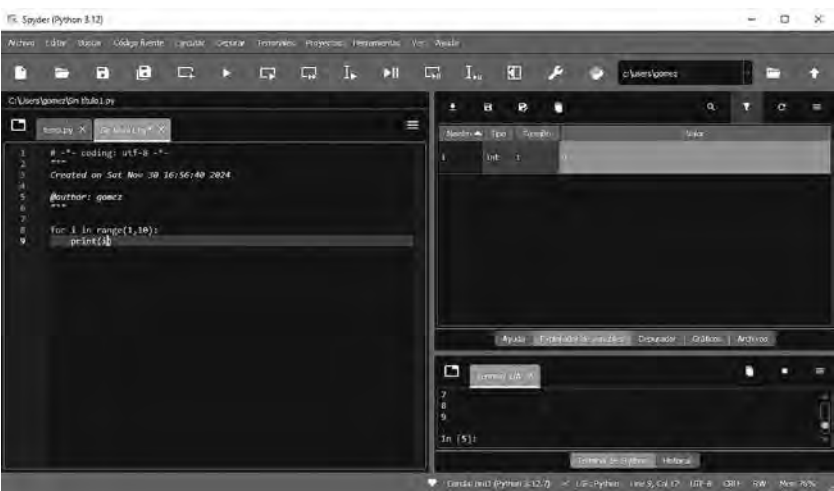
Figura 1.8  
Jupyter Notebook



**Spyder** es un IDE de código abierto que tiene integradas herramientas para el tratamiento de datos.

Para crear un nuevo programa se realiza sobre el menú Archivo/Nuevo archivo. En la parte de la izquierda se escriben las instrucciones del programa y al ejecutarlo (F5) el resultado se visualiza en la consola, y encima de ella los valores de las variables utilizadas.

Figura 1.9  
Spyder



La elección del entorno de desarrollo va a depender de las necesidades que tengamos para nuestro proyecto. Para un proyecto de **Data Analytics** en donde se quiera un entorno gratuito, Anaconda puede ser la mejor solución. En un entorno compacto y ligero para hacer un desarrollo de propósito general se puede optar por la versión gratuita de **PyCharm**. Si estamos acostumbrados en otros lenguajes a utilizar **Visual Studio Code**, esta puede ser la mejor solución.

Lo importante es que el programador se sienta cómodo con el entorno y se adapte bien a sus necesidades.