

Curso de Programación en Python

Tema 1. Introducción

Índice

Esquema

Ideas clave

- 1.1. Introducción y objetivos
- 1.2. ¿Qué es Python?
- 1.3. Instalación
- 1.4. Herramientas

A fondo

- Historia de Python
- Guía de estilos para el desarrollo en Python
- Instalación de Python
- Documentación Anaconda
- Documentación Jupyter Notebook

Test

Introducción a Python		
¿Qué es Python?	Instalación	Entornos de desarrollo
Historia de Python	Distribución básica de Python	Modo interactivo
Ventajas de Python	<ul style="list-style-type: none">▪ Descarga▪ Instalación▪ Instalación de nuevos módulos	IPython
Versiones de Python	Distribución Anaconda	Editores de texto plano
	<ul style="list-style-type: none">▪ Descarga▪ Instalación▪ Instalación de nuevos módulos	Entornos de desarrollo avanzados
		Jupyter Notebook
		<ul style="list-style-type: none">▪ Navegador de archivos▪ Vista del <i>notebook</i>

1.1. Introducción y objetivos

Python se ha convertido en uno de los lenguajes de programación más populares debido a su potencia y su facilidad para aprenderlo. En este tema pondremos en contexto qué es Python y qué características han hecho que este lenguaje sea uno de los más utilizados, sobre todo en campos de inteligencia artificial o análisis de datos.

Además, en este tema prepararemos nuestros equipos para poder desarrollar en Python. Para ello, explicaremos cómo instalar las dos distribuciones más populares dentro de Python y cómo instalar nuevos módulos en estas distribuciones para aumentar el número de funcionalidades. Por último, describiremos diferentes herramientas disponibles para desarrollar en este lenguaje y nos centraremos en Jupyter Notebook, que será el entorno de desarrollo que usaremos a lo largo del curso. Los objetivos que trataremos son:

- ▶ Contextualizar Python desde su historia y sus características.
- ▶ Comprender el problema de las versiones que ha existido hasta este año.
- ▶ Conocer los pasos para instalar Python en nuestro equipo.
- ▶ Conocer las distintas herramientas existentes para programar en Python.
- ▶ Comprender el entorno de desarrollo de Jupyter Notebook.

Evaluación:

Para obtener el certificado de superación del curso, deberás alcanzar una nota final mínima de 8.

Actividades a realizar:

- ▶ 4 actividades autocorregibles (2 puntos cada una) con intentos ilimitados.
- ▶ 8 test autocorregibles (0,25 puntos cada uno) con intentos ilimitados.

1.2. ¿Qué es Python?

Python es un lenguaje de propósito general que en los últimos años se ha ido haciendo cada vez más popular en áreas como *data science* o la inteligencia artificial.

Contexto de Python

Para entender un poco de dónde viene y por qué se ha vuelto tan popular, vamos a repasar su historia y las características más importantes.

Historia

Python fue creado en 1989 por Guido van Rossum. Guido empezó a implementar este lenguaje como pasatiempo con el objetivo de que fuera fácil de usar y aprender, pero, a la vez, que fuese un lenguaje potente.



Figura 1. Guido van Rossum, creador del lenguaje de programación Python. Fuente: <http://perlgeekbook.com/tag/guido-van-rossum/>

Van Rossum le dio el nombre de «Python» en homenaje al grupo Monty Python del que era fan. Aunque su desarrollo empezó en 1989, la primera versión no se publicaría hasta principios de 1991. Como hemos dicho, el objetivo principal de este lenguaje era que fuese fácil de entender, es decir, que el código que se escribiese con Python fuese más legible que con otros lenguajes de la época como C++ o Java.

Para lograr este objetivo, el desarrollador Tim Peters creó el Zen de Python, que define un conjunto de reglas que representan la filosofía de Python. Todos los usuarios pueden acceder a este conjunto de reglas a través de un *easter egg* que se introdujo en Python. Para verlo, solo tenemos que ejecutar la instrucción `import this` en una terminal con Python.

Siguiendo la filosofía propuesta en el Zen de Python, se creó una guía de estilo que se encuentra descrita en el *Python Enhancement Proposal*, abreviado como PEP, versión 8. Esta guía contiene las reglas de estilo que se aplicaron a la hora de desarrollar Python para que se sigan en el desarrollo de nuevas aplicaciones.

PEP 8 -- Style Guide for Python Code

PEP:	8
Title:	Style Guide for Python Code
Author:	Guido van Rossum <guido at python.org>, Barry Warsaw <barry at python.org>, Nick Coghlan <ncoghlan at gmail.com>
Status:	Active
Type:	Process
Created:	05-Jul-2001
Post-History:	05-Jul-2001, 01-Aug-2013

Figura 2. Información de la guía de estilos de Python PEP8. Fuente: <https://www.python.org/dev/peps/pep-0008/>

Toda esta filosofía está muy bien, pero Python no es tan popular solo por sus reglas de estilo y su filosofía. En la siguiente sección veremos las ventajas más importantes de este lenguaje de programación.

Ventajas de Python

Python tiene varias propiedades que lo han convertido en un lenguaje muy potente y fácil de aprender. Estas propiedades son las siguientes:

- **Tipado dinámico:** Python no necesita que definamos el tipo de las variables cuando las inicializamos como pasa, por ejemplo, en Java o C. Cuando inicializamos una variable Python le asigna el tipo del valor que le estamos asignando. Incluso, durante la ejecución, una misma variable podría contener valores con distintos tipos. Esta propiedad hace que sea más sencillo aprender a usarlo, aunque hace que sea más difícil detectar errores asociados con los tipos de datos.

- ▶ **Lenguaje multiparadigma:** Python permite aplicar diferentes paradigmas de programación como son la programación orientada a objetos, como Java o C++, programación imperativa, como C, o programación funcional, como Haskell.
- ▶ **Interpretado/*scripts*:** otra ventaja es que podemos ejecutar Python de forma interpretada o usando *scripts*. Es decir, puedo abrir una consola de Python y escribir y ejecutar las instrucciones una a una o, por otro lado, puedo crear un fichero que almacene todo el programa.
- ▶ **Extensible:** por último, Python cuenta con una gran cantidad de módulos y librerías que podemos instalar para incluir nuevas funcionalidades. Sin embargo, como Python está implementado usando C++, podemos crear nuevos módulos en C++ e incluirlos en Python haciendo que el lenguaje sea extensible a nuevos módulos.

Versiones en Python

Uno de los problemas que tenía una persona que se inicializaba en Python era saber qué versión tenía que instalar, ya que hasta finales de diciembre de 2019 existían dos versiones activas.

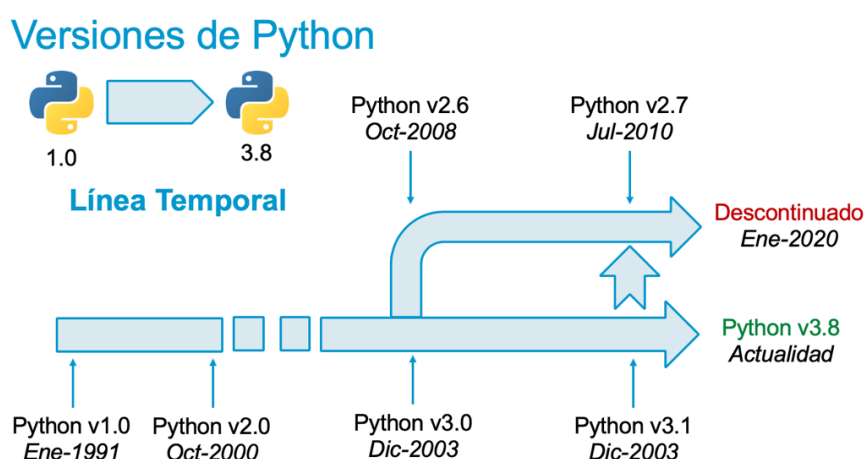


Figura 3. Línea temporal de las versiones de Python.

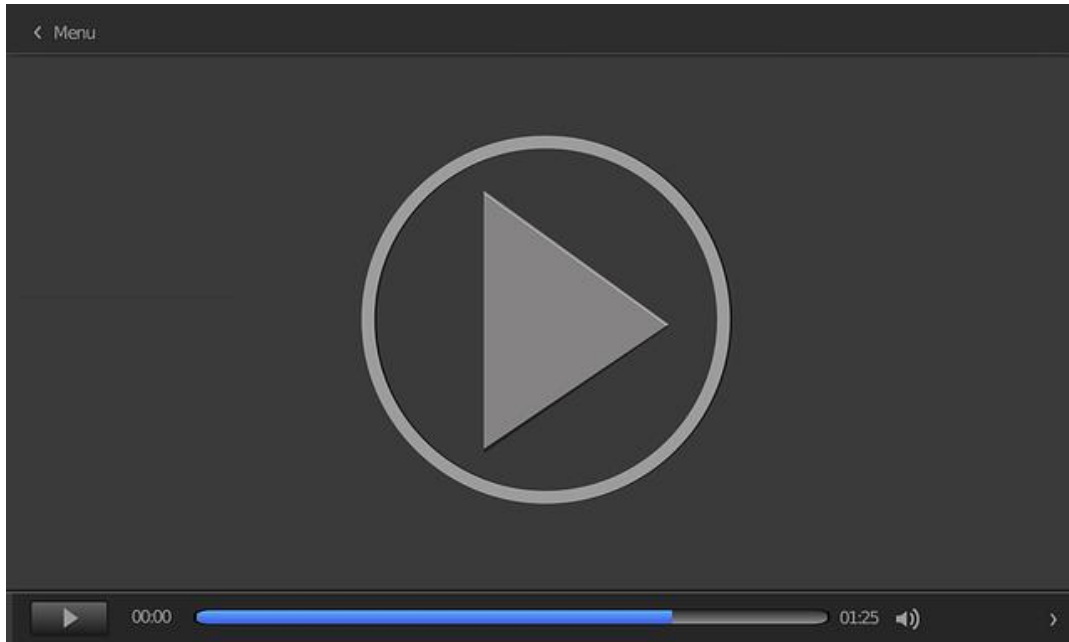
Como explicamos al principio, la primera versión de Python se publicó en 1991. Desde entonces se han publicado varias versiones que han seguido siendo más o menos retrocompatibles. Es decir, podía usar esas versiones en programas que había creado en versiones anteriores.

Sin embargo, a finales de 2008 se iba a publicar la versión 3.0. Esta nueva versión era un cambio radical con respecto a las predecesoras y esto hacía que no fuera compatible con las versiones anteriores de Python. Por este motivo, se publicó la versión 2.6 como soporte para los desarrollos de la versión 2. En la versión 2.6 se incluyeron nuevas funcionalidades de la versión 3, pero adaptadas a la versión 2. Desde ese momento Python contaba con 2 versiones y las novedades las tenían que duplicar en ambas. Por ejemplo, en 2010 publicaron la versión 3.1 y la 2.7 que incluía las nuevas funcionalidades, pero adaptadas a la versión 2.

Sin embargo, desde el equipo de Python siempre han explicado que las versiones 2.6 o 2.7 eran un parche y que no iban a ser versiones funcionales en un futuro. Este hecho se hizo oficial en enero de 2020 cuando se decidió que la versión 2.7 quedaba descontinuada y que a partir de entonces solo se publicarían nuevas funcionalidades para la versión 3.

En este curso trabajaremos con la versión 3.8 de Python, ya que en el momento de publicar este vídeo es la versión más reciente de Python dentro de la rama que publicará nuevas funcionalidades.

A continuación veremos el vídeo titulado *¿Qué es Python?*



Accede al vídeo:

<https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=da2e37bf-378e-4e89-98cd-af4e00c4f115>

1.3. Instalación

Distribución básica Python

La primera opción que podemos instalar es una distribución básica de Python. Esta distribución solo incluye los módulos principales que hay en Python. A continuación, veremos los pasos para instalarlo.

Descargar Python

La primera opción de instalación es únicamente los módulos principales del lenguaje. En esta opción tendremos que instalar nuevos módulos si queremos ampliar las funcionalidades de Python. Para poder descargarnos esta versión, deberemos acceder a la sección de descargas de la página principal de Python.

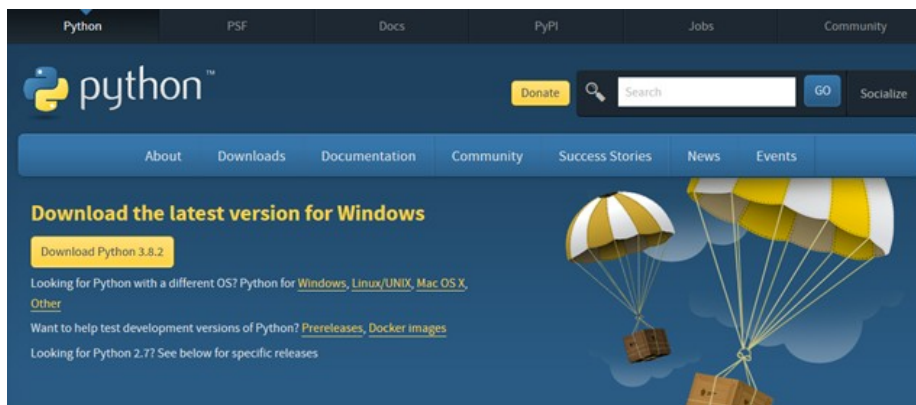


Figura 4. Página principal de descargas de Python. Fuente: <https://www.python.org/downloads/>

En esta página tenemos una opción para instalar la última versión estable de Python. En nuestro caso, es la versión 3.8. Para descargarlo, hacemos clic en el botón *download Python 3.X* y automáticamente empezará la descarga del paquete de instalación.

Instalación Python

Una vez que tenemos el paquete de instalación descargado, podemos iniciar el proceso de instalación. Para ello, ejecutaremos el fichero *python-3.X.exe* que hemos ejecutado y, a continuación, se nos abrirá el asistente de configuración.



Figura 5. Asistente de configuración de la instalación de Python.

La primera ventana del asistente nos muestra dos formas para instalar Python. La primera de ellas, *install now*, nos permite instalar Python con la configuración por defecto. En cambio, *customize installation* permite cambiar algunos parámetros como, por ejemplo, dónde vamos a instalar Python. Por último, tenemos dos opciones que tenemos que comprobar que están marcadas para, en primer lugar, instalar Python a todos los usuarios de un equipo y, en segundo lugar, almacenar Python en el PATH del sistema operativo. Para nuestra instalación elegiremos la opción *install now*.

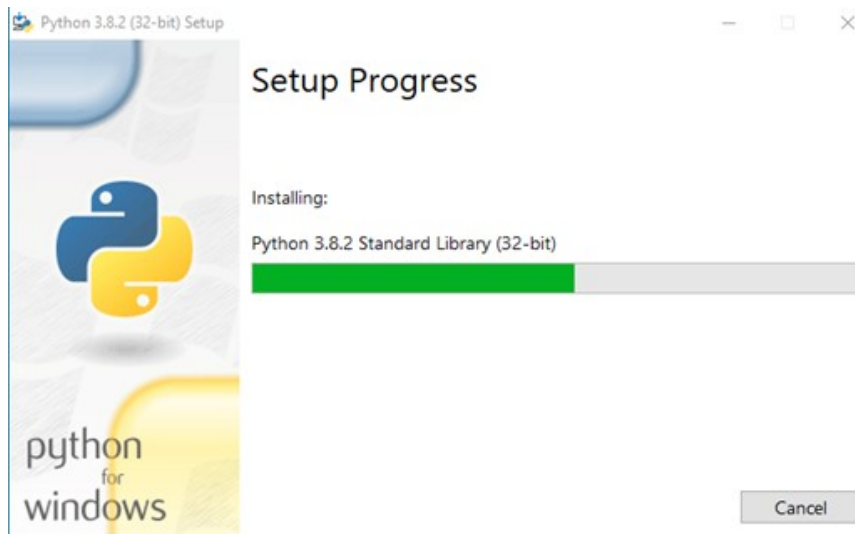


Figura 6. Progreso de la instalación de Python.

A continuación, aparecerá una ventana como la Figura 6, en la que se mostrará el progreso de la instalación de Python. Pasados unos minutos, el asistente nos informará de que la instalación ha terminado con éxito y podremos cerrar el asistente con el botón *close*.

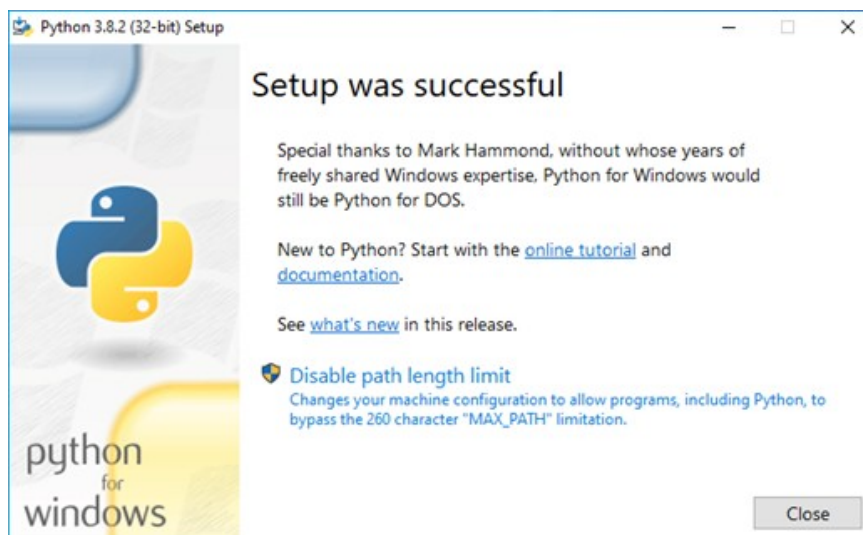
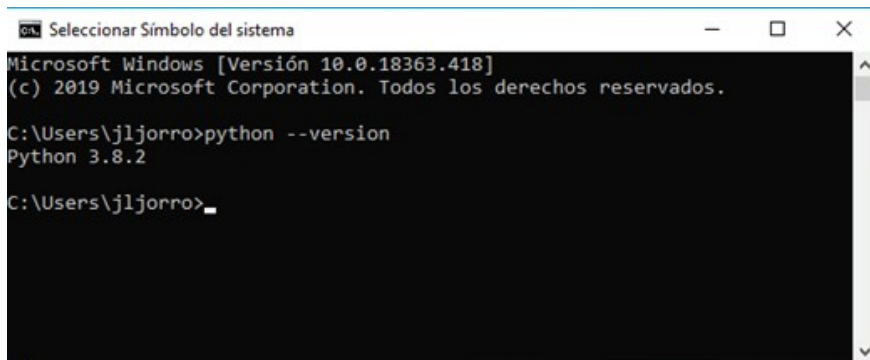


Figura 7. Confirmación de que la instalación se realizó correctamente.

También podemos comprobar que la instalación se realizó correctamente. Una forma de comprobarlo es preguntando al sistema por la versión que tenemos instalada de Python. Para ello abrimos el símbolo del sistema y ejecutamos la instrucción `python --version`. Esto nos debería mostrar la versión de Python que hemos instalado.



```
Microsoft Windows [Versión 10.0.18363.418]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\j1jorro>python --version
Python 3.8.2

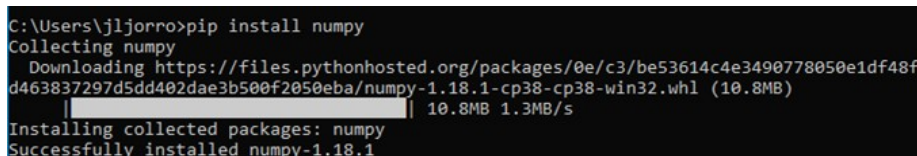
C:\Users\j1jorro>
```

Figura 8. Comprobación de la instalación desde el símbolo del sistema.

Instalación de nuevos módulos

La versión que hemos instalado solo contiene los módulos básicos de Python. Por este motivo suele ser necesario instalar los nuevos módulos que queremos incluir en nuestros programas. Para instalar nuevos módulos, Python incluye el paquete de instalación para Python (pip).

Si queremos instalar un nuevo módulo en Python, tenemos que ejecutar la instrucción `pip install nombre_modulo`. Por ejemplo, si queremos instalar el módulo *numpy*, abriremos el símbolo del sistema y ejecutaremos la instrucción de instalación.



```
C:\Users\j1jorro>pip install numpy
Collecting numpy
  Downloading https://files.pythonhosted.org/packages/0e/c3/be53614c4e3490778050e1df48f
d463837297d5dd402dae3b50f2050eba/numpy-1.18.1-cp38-cp38-win32.whl (10.8MB)
    | 10.8MB 1.3MB/s
Installing collected packages: numpy
Successfully installed numpy-1.18.1
```

Figura 9. Comprobación de la instalación desde el símbolo del sistema.

De esta forma podremos instalar todos los módulos que se puedan necesitar en el futuro. Sin embargo, hay otra distribución que tiene algunos módulos preinstalados, sobre todo aquellos asociados a la *data science*. En el siguiente capítulo veremos cómo instalar esta distribución.

Distribución Anaconda

Otra distribución con la que podemos instalar Python es Anaconda. Esta distribución, aparte de instalar la versión básica de Python, incluye otros módulos importantes dentro del análisis de datos, como son *numpy* o *pandas*. Esta es la distribución que usaremos en el curso.

Descargar Anaconda

Otra distribución de Python es la que nos proporciona Anaconda. Esta distribución nos instala los módulos principales de Python y otros módulos importantes en el análisis de datos como son *numpy* y *pandas*. Esta es la distribución que usaremos durante el curso. Para descargarnos Anaconda, nos dirigimos a la sección de descargas de su página y seleccionamos la versión 3.7.

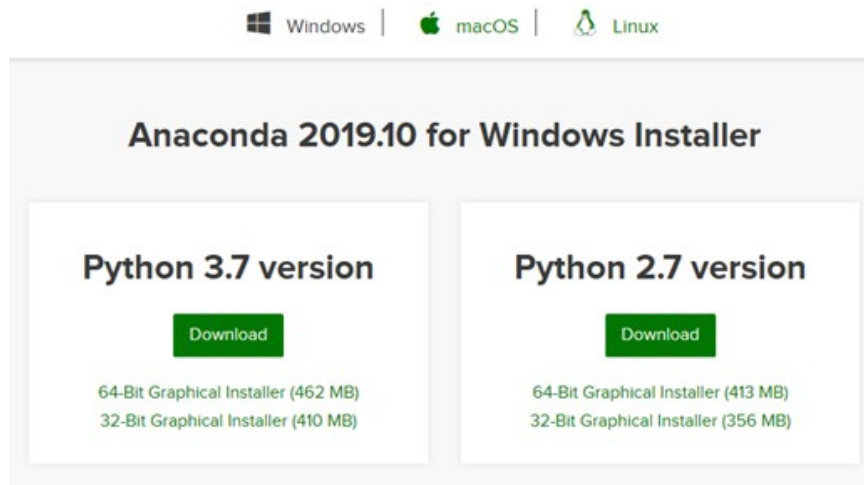


Figura 10. Página principal para descargar distintas versiones de Anaconda. Fuente:

<https://www.anaconda.com/products/individual#download-section>

Una vez descargada la versión de Anaconda, procederemos a su instalación en nuestro equipo.

Instalación Anaconda

Para iniciar la instalación de Anaconda en nuestro equipo, ejecutamos el fichero *Anaconda3-XXX.exe*. Hecho esto, se nos abrirá el asistente de instalación que nos guiará en los pasos de configuración de Anaconda.

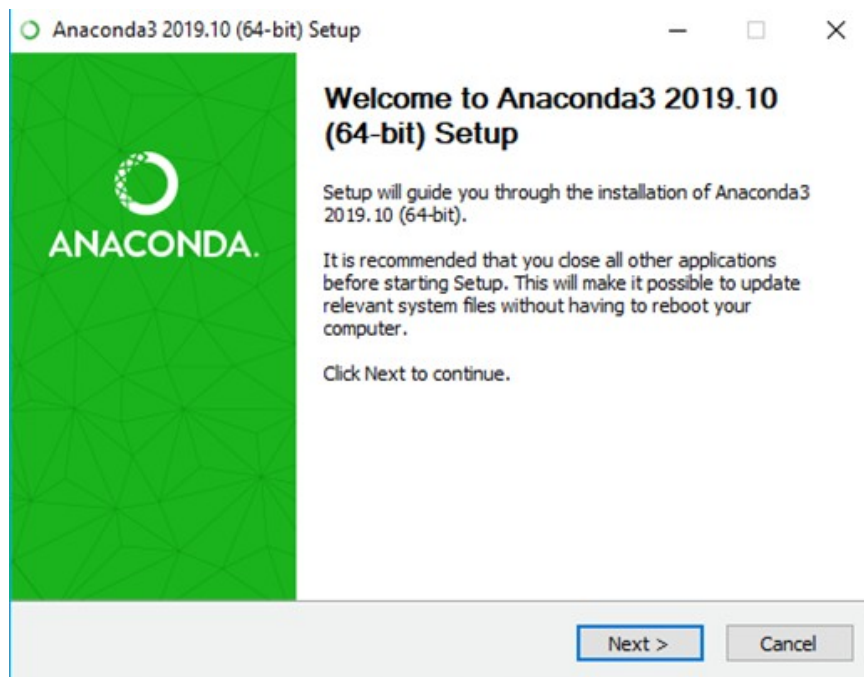


Figura 11. Ventana bienvenida instalación Anaconda.

Seleccionamos el botón *Next >* y nos mostrará el acuerdo de licencia del *software*. Para continuar con la instalación, es necesario aceptar los términos de la licencia con el botón *I Agree*.

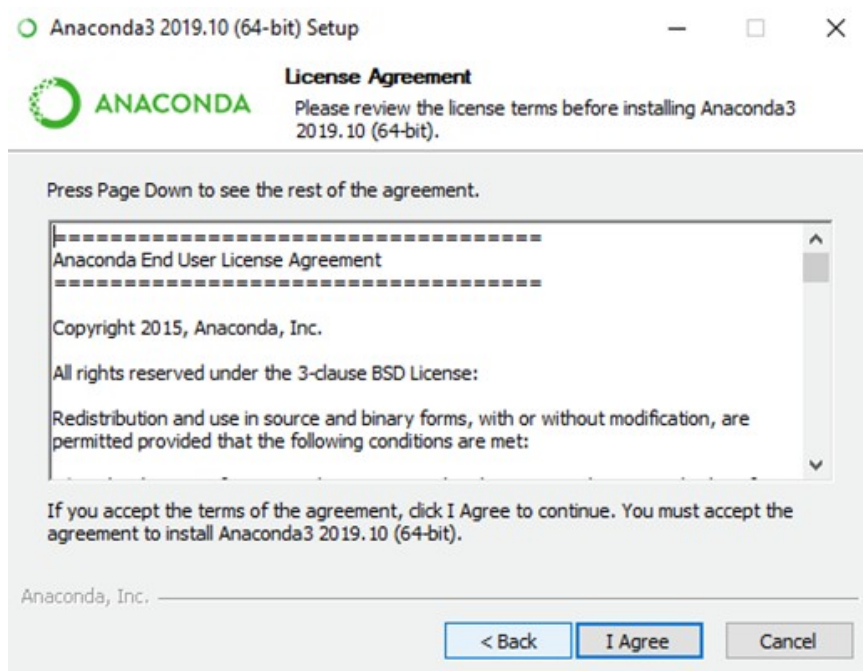


Figura 12. Acuerdo de licencia de Anaconda.

En el siguiente paso, el asistente nos preguntará el tipo de instalación que queremos hacer. La primera opción es hacer una instalación únicamente para el usuario que se está ejecutando en ese momento. La segunda opción permite instalar Anaconda a todos los usuarios que comparten un mismo equipo, aunque para ello necesita permisos de administración. Seleccionamos una de las dos opciones y continuamos con el botón *Next >*.

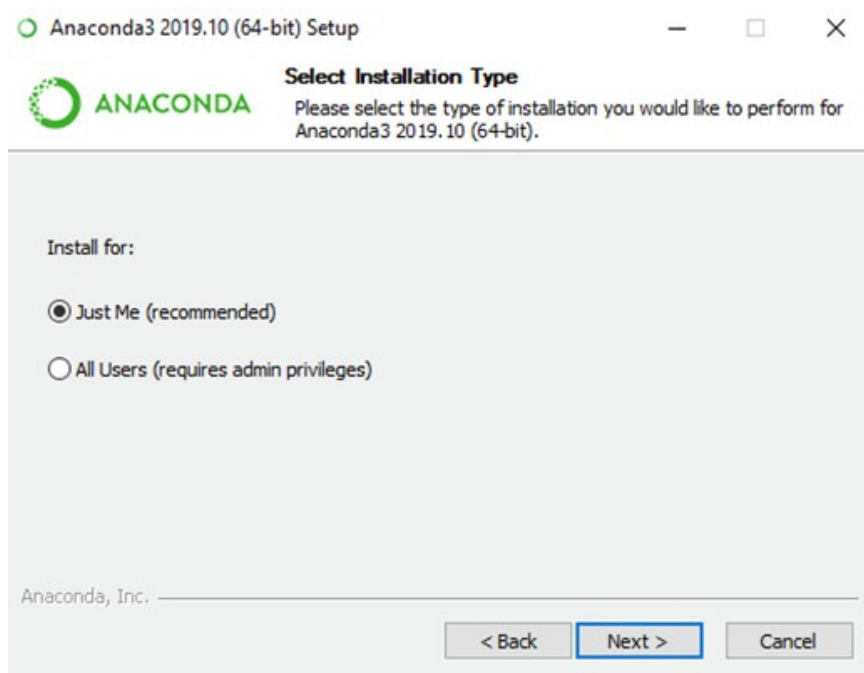


Figura 13. Selección del tipo de instalación de Anaconda.

Una vez decidido qué usuarios pueden acceder a Anaconda, toca elegir en qué carpeta deseamos que se instale el *software*. En nuestro caso dejaremos la carpeta que viene por defecto. En este paso hay que prestar atención al espacio libre que tenemos en el sistema, ya que Anaconda ocupa 2,9 GB. Si hay espacio suficiente y hemos elegido la carpeta, continuamos la instalación pulsando *Next >*.

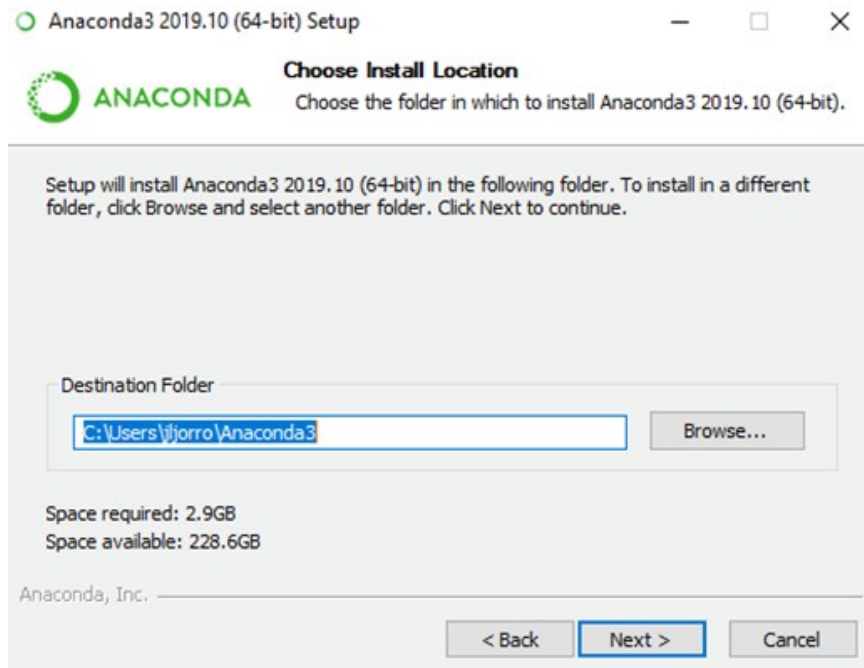


Figura 14. Selección de la localización de instalación de Anaconda.

El siguiente paso es muy importante. Tenemos que asegurarnos de que las dos opciones avanzadas que se nos muestran están seleccionadas. La primera de ellas incluye Anaconda en el PATH del sistema. Esto nos permitirá acceder a Anaconda desde el símbolo del sistema. La segunda opción hace que la versión de Python instalada en Anaconda sea la versión por defecto del sistema, en nuestro caso la versión 3.7. Una vez seleccionadas ambas opciones, continuamos la instalación con el botón *Install*.

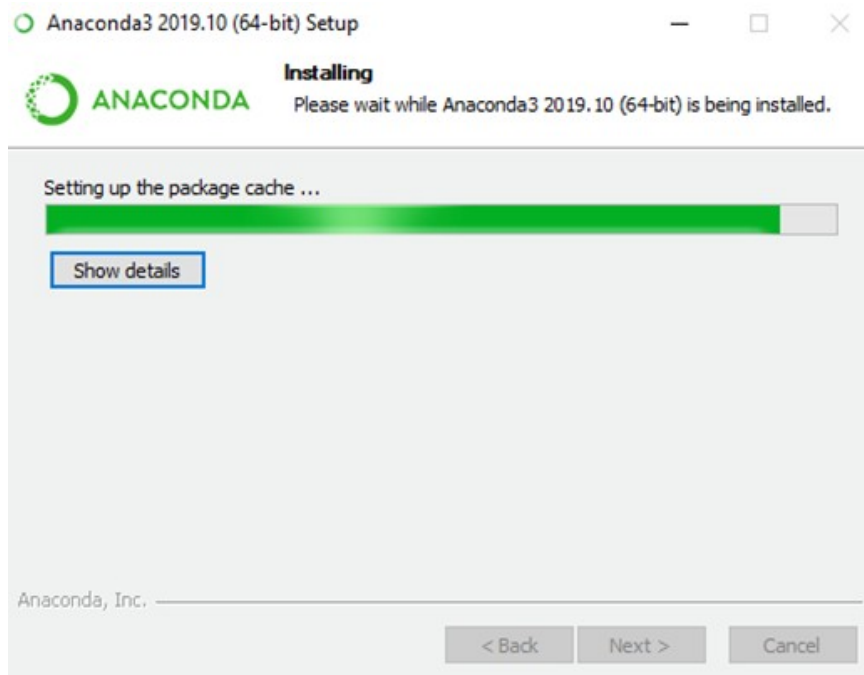


Figura 15. Progreso de instalación de Anaconda.

En ese momento, el asistente nos mostrará una barra de progreso de la instalación. Pasado un tiempo, el sistema nos indicará que la instalación se ha completado. En ese momento pulsamos el botón *Next >* para finalizar la instalación. En las ventanas siguientes nos mostrarán algunos mensajes de editores y opciones que podemos utilizar con Anaconda. Seguiremos dando al botón *Next >* hasta llegar al último paso donde pulsaremos el botón *Finish*.

Para comprobar que la instalación se ha realizado correctamente, nos dirigiremos al botón de inicio de Windows y entre los programas tendremos una carpeta con el nombre «Anaconda3».

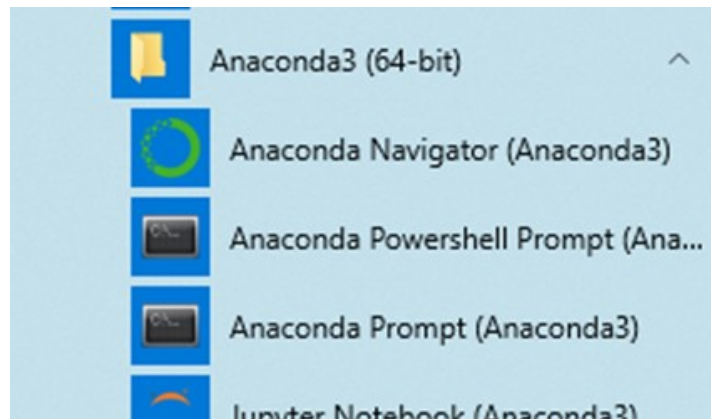


Figura 16. Localización de Anaconda en el menú de inicio de Windows.

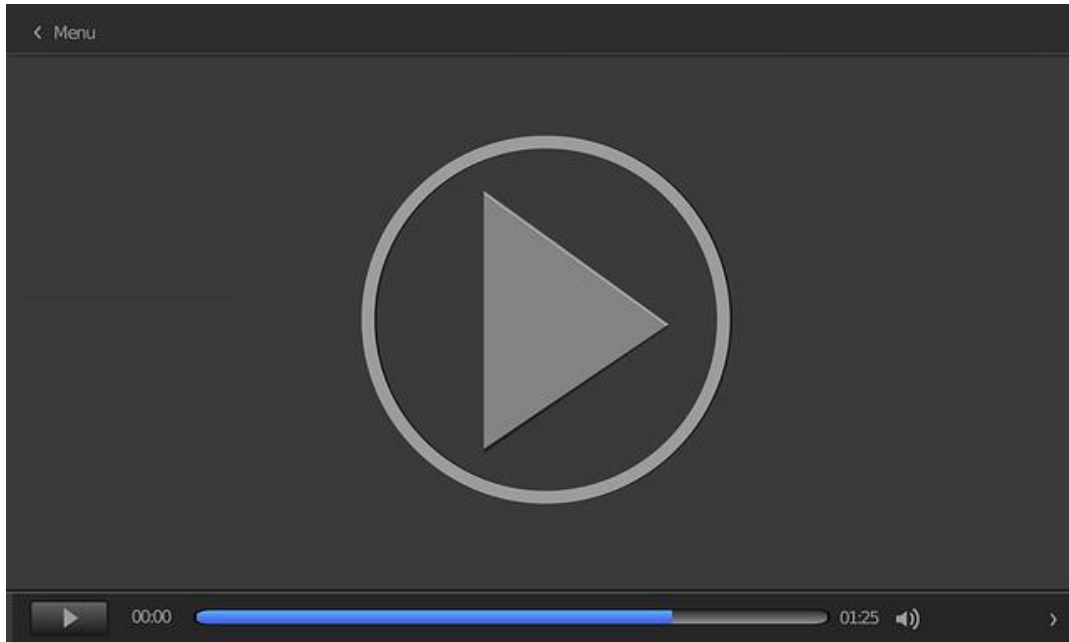
Instalación de nuevos módulos

Aunque la distribución de Anaconda incluye nuevas funcionalidades a los módulos básicos de Python, no están incluidos todos los posibles. Por este motivo también puede ser necesario instalar nuevos módulos en nuestra distribución. Para instalar estos nuevos módulos, podemos utilizar la opción de instalación que tenemos desde el símbolo del sistema: `conda install nombre_modulo`. Por ejemplo, vamos a instalar el módulo *scikit-learn* en nuestro sistema.

```
C:\Users\jljorro>conda install scikit-learn
Collecting package metadata (current_repodata.json): done
Solving environment: done
```

Figura 17. Comprobación de la instalación desde el Símbolo del sistema.

A continuación veremos el vídeo titulado *Instalación de Python*.



Accede al vídeo:

<https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=4d8d3e92-f6c7-4e1a-a898-af4e00c5ee2a>

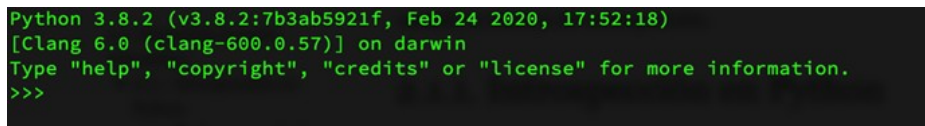
1.4. Herramientas

Entornos de desarrollo

Como ya hemos explicado, Python es un lenguaje que podemos ejecutarlo de dos formas principalmente: usando el intérprete o usando *scripts*. En este apartado veremos los diferentes entornos de desarrollo que encontramos para programar en Python. Además, explicaremos en detalle el funcionamiento de Jupyter Notebook, ya que será la herramienta con la que trabajemos en este curso.

Modo interactivo

Python es un lenguaje interpretado, es decir, Python es capaz de ir ejecutando las instrucciones según las vamos introduciendo. Por este motivo, la instalación de Python incluye el intérprete en el que podemos ejecutar instrucciones. Para iniciar este intérprete, solo tenemos que ejecutar la instrucción `python` en nuestra consola de comandos.



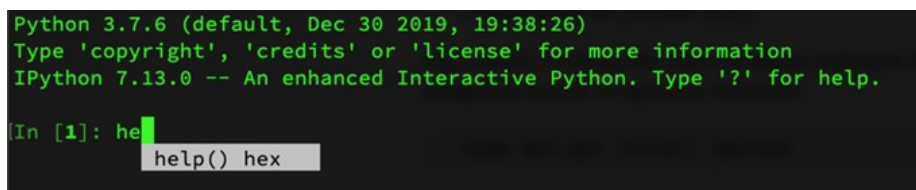
```
Python 3.8.2 (v3.8.2:7b3ab5921f, Feb 24 2020, 17:52:18)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Figura 18. Intérprete de Python para la versión 3.8.

Usando este modo podemos conocer algunos elementos del lenguaje Python como sus clases o funciones. Además, podemos consultar la documentación del lenguaje desde el propio intérprete. Os animamos a que uséis el intérprete de Python para probar los conceptos básicos que vemos durante el curso. Para salir del intérprete solo debemos ejecutar el comando `exit()`.

IPython

Para mejorar el intérprete de Python, se puede instalar el paquete IPython (las instrucciones se encuentran en <https://ipython.org/install.html>). IPython añade más funcionalidades al intérprete de Python, como son el resaltado de errores, completado automático de variables o módulos a través del tabulador, etc. Una vez instalado, para iniciar este intérprete solo debemos ejecutar la instrucción `ipython`.



```
Python 3.7.6 (default, Dec 30 2019, 19:38:26)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.13.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: he
        help() hex
```

Figura 19. Intérprete IPython ejecutando el completado automático de nombres usando el tabulador.

Editores de texto plano

Las dos opciones anteriores nos permiten ejecutar pequeñas instrucciones en Python y poder consultar la documentación de objetos y módulos. Sin embargo, para crear programas más complejos es necesario escribir *scripts* que contienen más instrucciones o diferentes bloques de código. Una de las primeras opciones que se pueden utilizar para implementar estos *scripts* es la utilización de editores de texto plano. Existen muchos tipos de editores de texto para todos los sistemas operativos, algunos ejemplos pueden ser:

- ▶ Nano o vim (sistemas UNIX).
- ▶ Bloc de notas de Windows.
- ▶ Sublime Text 3.
- ▶ Atom.

- ▶ Notepad++ (solo Windows).
- ▶ Visual Code.

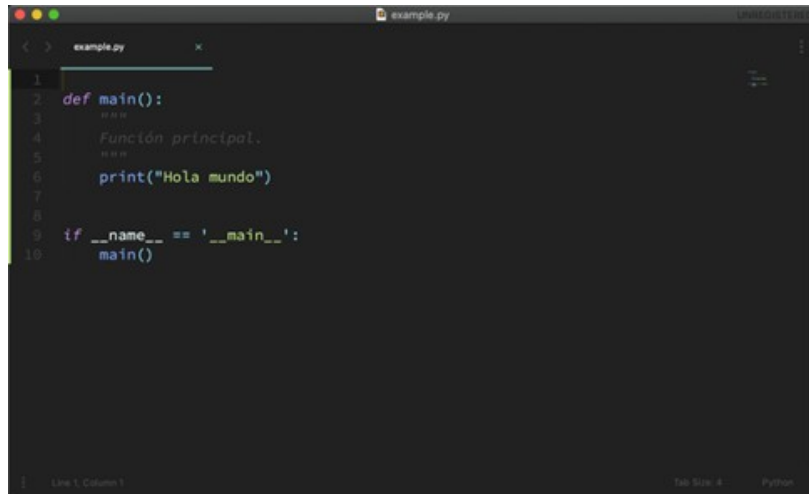


Figura 20. Ejemplo de *script* de Python implementado en el editor Sublime Text.

Entornos de desarrollo avanzados

Por último, existen diferentes entornos de desarrollo avanzados orientados a Python. Estos entornos están orientados a grandes proyectos en Python y se incluyen muchas más funcionalidades como son la gestión de repositorios. Algunos de estos entornos de desarrollo pueden ejecutar en un mismo proyecto *scripts* y *notebooks*, como es el caso de PyCharm. Los entornos de desarrollo más utilizados son:

- ▶ PyCharm.
- ▶ Eclipse PyDev.
- ▶ Spyder.

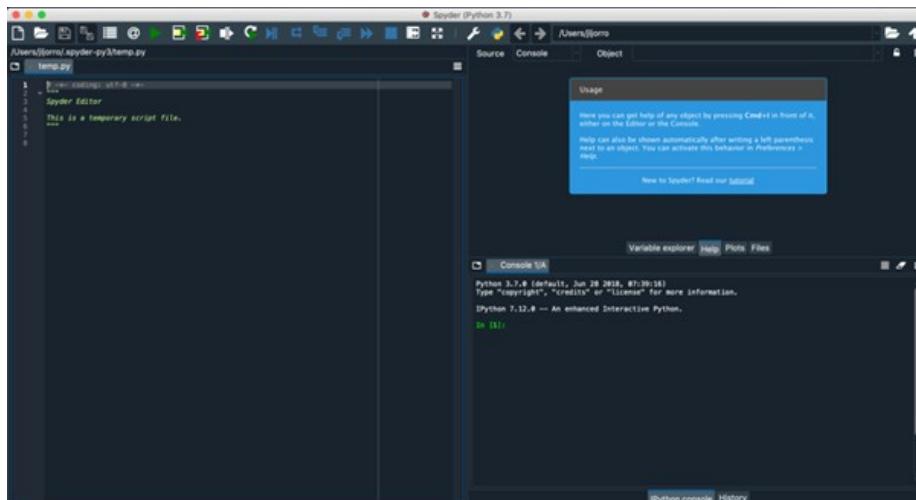


Figura 21. Ventana principal del entorno de desarrollo Spyder.

Jupyter Notebook

Jupyter Notebook es una aplicación web incluida en la distribución Anaconda. Esta aplicación web es una extensión de IPython, donde se añaden funcionalidades y se mejora la interfaz gráfica. La principal característica que tiene Jupyter Notebook es la creación de celdas con objetivos específicos. Estos objetivos específicos de cada celda pueden ser: ejecutar código de Python, incluir texto en *markdown* o visualizar gráficos. Es la aplicación más utilizada en el campo de la ciencia de datos.

Para abrir Jupyter Notebook en nuestro equipo, ejecutaremos la aplicación Anaconda Navigator. Esta aplicación nos mostrará distintas herramientas que podemos ejecutar. Desde ahí pulsamos en el botón *Launch* de Jupyter Notebook.

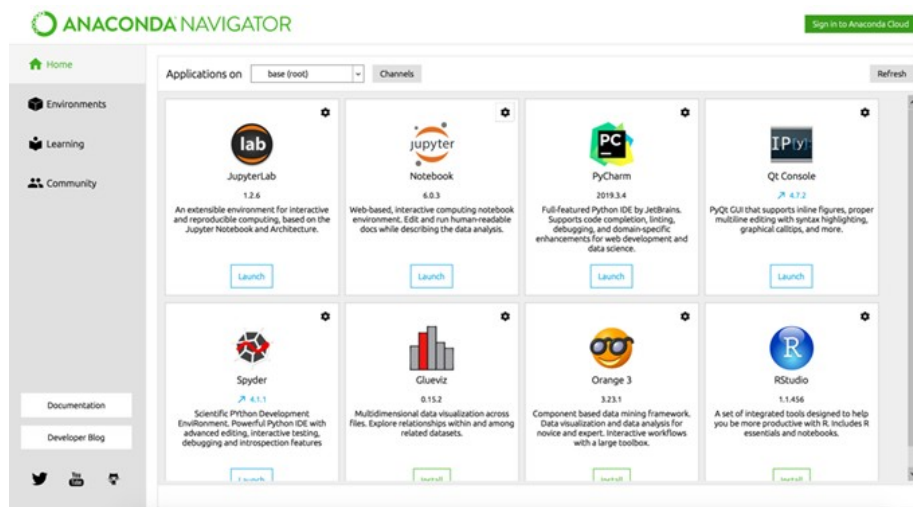


Figura 22. Vista de la ventana de aplicaciones de Anaconda Navigator.

Esto hará que nuestro navegador web predeterminado abra Jupyter Notebook como una aplicación web. La primera ventana que veremos será el navegador de archivos de Jupyter que explicaremos a continuación.

Navegador de archivos

La primera ventana que nos muestra Jupyter Notebook es el navegador de archivos. A través de esta ventana podemos movernos en nuestro sistema de ficheros, crear nuevas carpetas o archivos y renombrar ficheros.



Figura 23. Vista del navegador de archivos de Jupyter Notebook.

En primer lugar, para navegar entre las carpetas, solo tendremos que hacer clic en el nombre de la carpeta a la que queremos acceder. Si lo que queremos es volver a la carpeta superior, solo debemos hacer clic en la carpeta que tiene como nombre dos puntos (..).

Para crear un nuevo fichero, pulsamos sobre el botón *New*. Una vez hecho esto, se nos desplegarán varias opciones para crear un fichero: *notebook* (con la versión correspondiente de Python) u otro tipo de fichero como, por ejemplo, un fichero de texto o una carpeta.

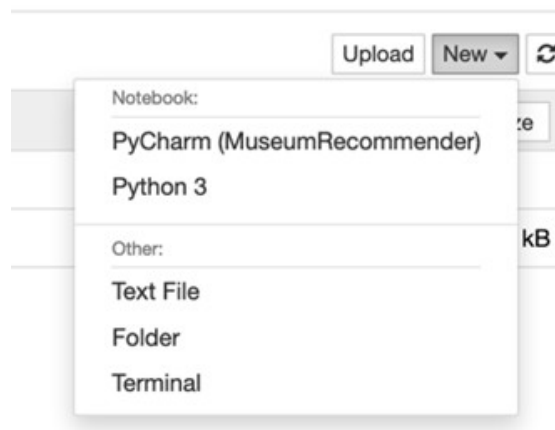


Figura 24. Menú con los tipos de fichero que se pueden crear.

Esto hará que se cree un fichero del tipo seleccionado en la carpeta en la que nos encontremos en ese momento. Vamos a crear un fichero de tipo *notebook* y, a continuación, explicaremos la ventana que nos muestra Jupyter dentro de un *notebook*.

Vista del *notebook*

Cuando creamos un *notebook* o abrimos uno que ya existe veremos una pantalla similar a la que se muestra en la Figura 24. En la parte inferior veremos todas las celdas de nuestro *notebook*. Cada una de estas celdas pueden ser de los siguientes tipos:

- ▶ Código en Python.
- ▶ Texto con *markdown*.
- ▶ Formato *raw* en la que se muestra el texto con el formato de consola.
- ▶ Formato *heading* que crea una celda con formato título.

Además, existe otro tipo de celda que se da cuando una celda de código en Python devuelve un resultado como, por ejemplo, si utilizamos la instrucción `print`. Estas celdas se crearán de forma automática.

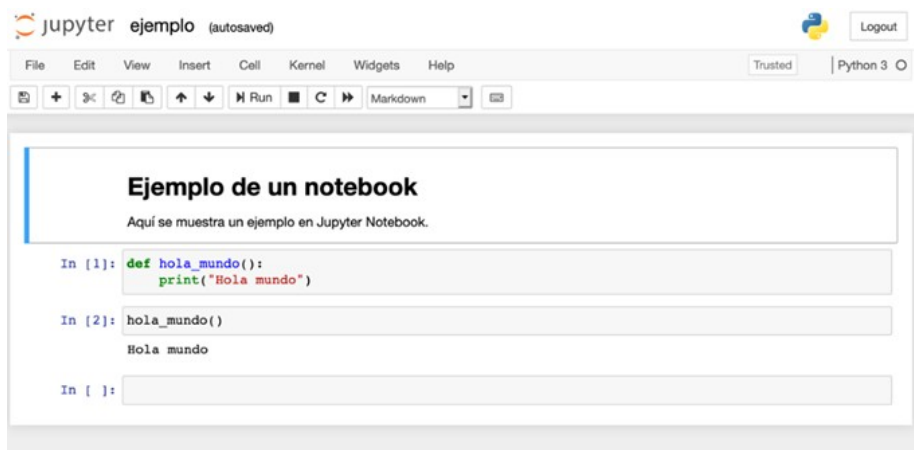


Figura 25. Ejemplo de un *notebook* en Jupyter.

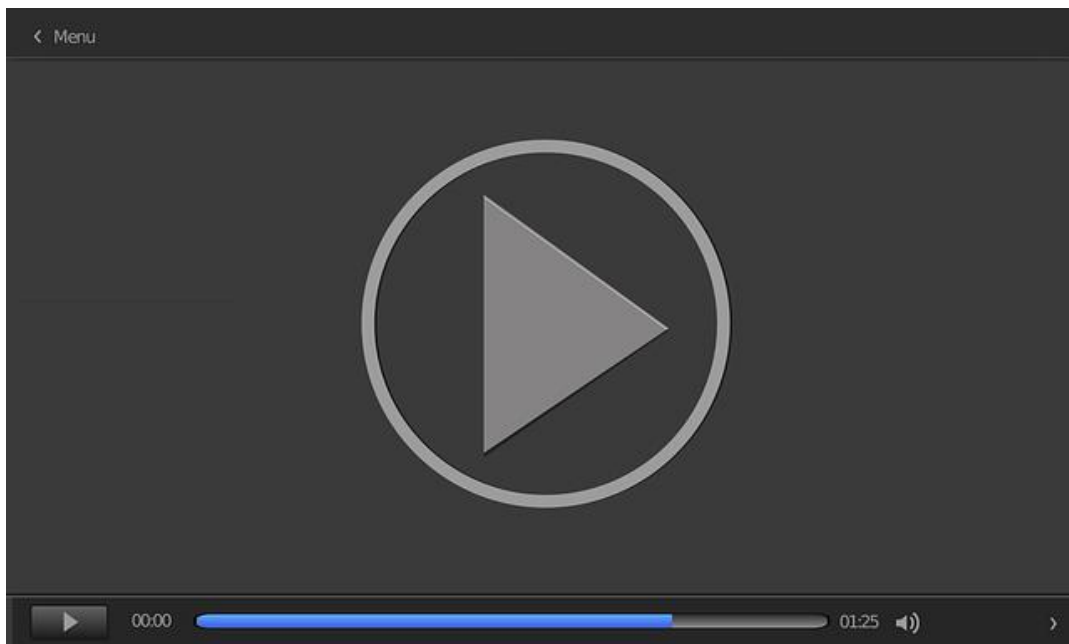
Justo encima de las celdas, tenemos un conjunto de botones que nos permiten interactuar con las celdas. A continuación, explicaremos cada uno de estos botones siguiendo el orden de izquierda a la derecha:

- ▶ **Guardar:** almacena en el fichero todas las celdas y guarda el estado de ejecución en el que se quedó el *notebook*.
- ▶ **Nueva celda:** crea una nueva celda inmediatamente después de la celda que tenemos seleccionada.
- ▶ **Cortar:** permite cortar una celda para pegarla en otro punto del *notebook*.
- ▶ **Copiar:** permite copiar una celda para poder pegarla en otro punto del *notebook*.
- ▶ **Pegar:** pegamos la celda que hemos copiado o cortado previamente inmediatamente después de la celda que tenemos seleccionada.
- ▶ **Bajar una celda:** desplaza la celda seleccionada una posición hacia abajo.
- ▶ **Subir una celda:** desplaza la celda seleccionada una posición hacia arriba.
- ▶ **Ejecutar celda:** ejecuta el contenido que hay en la celda seleccionada. Si esa celda es de tipo código, ejecutará las instrucciones y devolverá la salida en una celda de salida. Por otro lado, si la celda es de tipo texto, le asignará un formato HTML.
- ▶ **Stop:** para la ejecución del *kernel* de Python. Para poder seguir ejecutando nuevas celdas, es necesario reiniciar el *kernel* de Python.
- ▶ **Reiniciar *kernel*:** reinicia la ejecución del *kernel*, eliminando de la memoria toda la información del *notebook* que tuviese almacenada.
- ▶ **Reiniciar el *kernel* y ejecutar todas las celdas:** reinicia el *kernel* de Python como el botón anterior y, después, ejecuta todas las celdas del *notebook*.
- ▶ **Selección del tipo de celda:** permite seleccionar el tipo de celda que tenemos seleccionado. Los tipos son los que hemos descrito anteriormente.

En la parte superior se encuentra el menú que incluye muchas más funciones. A continuación, explicamos algunas de las funciones más útiles y dónde se encuentran:

- ▶ Crear un nuevo *notebook* [*File >> New Notebook*].
- ▶ Descargar *notebook* en otro formato (HTML, PDF...) [*File >> Downloads*].
- ▶ Cerrar el *notebook* y apagar el *kernel* [*File >> Close and Halt*].
- ▶ Insertar celdas encima o debajo de la celda seleccionada [*Insert*].
- ▶ Referencia de Python y librerías utilizadas en Jupyter [*Help*].

A continuación veremos el vídeo titulado *Herramientas para programar en Python*.



Accede al vídeo:

<https://unir.cloud.panopto.eu/Panopto/Pages/Embed.aspx?id=d7449370-1b0f-4090-8f56-af4e00c6ac22>

Historia de Python

Wikipedia. (13 de junio de 2020). Historia de Python [Página web].
https://es.wikipedia.org/wiki/Historia_de_Python

En esta página se hace un repaso más a fondo de la historia de Python, las influencias que ha recibido de otros lenguajes y una cronología de las versiones de Python con las características que se iban incluyendo en cada una de las versiones.

Guía de estilos para el desarrollo en Python

Python. (2013). PEP 8 - Style guide for Python code [Página web].
<https://www.python.org/dev/peps/pep-0008/>

La página oficial donde se incluye la guía de estilo para el desarrollo de proyectos en Python. Esta guía contiene los estándares de estilo que se siguieron en el desarrollo del propio lenguaje de programación.

Instalación de Python

Python. (24 de septiembre de 2020). Python setup and usage [Página web].
<https://docs.python.org/3/using/index.html>

Sección de la documentación de Python que está orientada a la instalación y configuración de Python para diferentes plataformas. Además, incluye la descripción de algunas herramientas que facilitan el desarrollo en Python.

Documentación Anaconda

Anaconda. (2020). Anaconda individual edition [Página web].
<https://docs.anaconda.com/anaconda/>

Documentación de la distribución Anaconda que incluye las instrucciones de instalación, la guía de uso y una referencia con los términos y dudas más frecuentes.

Documentación Jupyter Notebook

Jupyter Notebook. (2015). The Jupyter Notebook [Página web]. <https://jupyter-notebook.readthedocs.io/en/stable/>

Documentación oficial de la aplicación Jupyter Notebook que usaremos a lo largo del curso para explicar los conceptos de Python.

1. Python se creó con el objetivo de:
 - A. Ser más potente que otros lenguajes de la época.
 - B. Crear un lenguaje fácil de usar y de aprender.
 - C. Programar usando interfaces visuales.
 - D. Poder crear aplicaciones de servidor más fácilmente.

2. ¿Cuál de las siguientes propiedades tiene Python?
 - A. El tipo de las variables se hace de forma dinámica.
 - B. Se pueden programar en diferentes paradigmas como programación orientada a objetos o programación funcional.
 - C. Podemos extender las funcionalidades de Python creando módulos en C++.
 - D. Todas las anteriores.

3. La versión de Python que debemos usar es:
 - A. Tenemos que usar la versión 3.8 o posterior.
 - B. Podemos usar cualquier versión de Python, ya que no cambian mucho.
 - C. Tenemos que usar la versión 2.7.
 - D. Depende de las características de nuestro equipo.

4. ¿Cuál de las siguientes herramientas no podemos usar para programar en Python?
 - A. Editor de texto plano como Atom.
 - B. La consola del sistema.
 - C. Aplicaciones de ofimática como Word.
 - D. Entornos de desarrollo avanzado como PyCharm.

5. ¿Cómo puedo instalar Jupyter Notebook en mi equipo?
- A. Instalando la distribución Anaconda.
 - B. Instalando la distribución básica de Python.
 - C. Instalando un editor de texto plano.
 - D. A través de una suscripción.
6. ¿Qué diferencia existe entre las distribuciones de Python?
- A. La distribución básica de Python incluye más módulo que Anaconda.
 - B. La distribución de Anaconda incluye más módulo que la versión básica.
 - C. Anaconda no permite instalar más módulos.
 - D. La versión básica de Python no permite instalar más módulos.
7. ¿Qué es Jupyter Notebook?
- A. Un editor de texto plano.
 - B. Un comando de la consola para ejecutar instrucciones de Python.
 - C. Un entorno que permite crear celdas con diferentes funcionalidades (escribir código, documentar, etc.).
 - D. Otra distribución para instalar Python en nuestro equipo.
8. ¿Qué tipo de celdas podemos usar en Jupyter Notebook?
- A. Celdas con código de Python.
 - B. Celdas con texto basado en *markdown*.
 - C. Texto con formato consola.
 - D. Todas las anteriores.

9. ¿Qué tipo de ficheros no podemos crear con el navegador de archivos de Jupyter Notebook?

- A. *Notebooks*.
- B. Carpetas.
- C. Ficheros de Texto.
- D. Ficheros PDF.

10. En Jupyter Notebook:

- A. Las celdas de salida se crean automáticamente.
- B. Tenemos que ejecutar las instrucciones en la consola de Python.
- C. No se pueden exportar los *notebooks* a otros formatos como PDF o HTML.
- D. No se puede reiniciar el *kernel* para borrar la memoria.