Python

Guía para ser un Pythonista

Juan José Lozano Gómez – j2logo.com

1 Introducción

Índice de contenidos

Introducción

Bienvenid@ al curso ***Python – Guía para ser un Pythonista***.

Este no es un curso de Python más, es el curso definitivo para aprender Python. Te lo digo de verdad, porque si no fuera así, me estaría engañando a mí mismo y a toda la gente que me sigue desde hace tiempo. Y yo no quiero eso.

He querido desarrollar un curso, el curso, no solo para que aprendas lo básico del lenguaje, sino para que entiendas qué es realmente Python, para qué lo puedes utilizar y cómo usarlo correctamente en función del problema que pretendas resolver.

Una vez finalizado el curso, te aseguro que estarás en disposición de utilizar el lenguaje para el ámbito al que te quieras dedicar: *desarrollo web*, *análisis de datos*, *machine learning*, *scraping*, *big data*, *scripting*. Estos temas no se tratan en el curso. Sin embargo, *Python – Guía para ser un Pythonista*, es el primer paso, el más importante, para poder afrontar con éxito cualquiera de los retos anteriores.

¿Por qué *Guía para ser un Pythonista*? Porque una cosa es conocer lo básico del lenguaje y, otra muy distinta, saber utilizarlo correctamente, según las guías y recomendaciones oficiales del propio Python. **Como un auténtico Pythonista**. Este, en definitiva, es el propósito del curso. **Convertirte en un Pythonista**.

Por si no me conoces, me voy a presentar muy brevemente. Me llamo Juanjo, conocido como J2LOGO, y seré tu profesor durante el curso. Desarrollo en Python desde 2013. Aprendí el lenguaje por mi cuenta, para programar proyectos web personales con *Django*. Me apasiona la programación y desde que descubrí Python todavía más. Ahora mismo ya no uso *Django*, sino *Flask*, porque me parece un framework que se adapta mucho mejor a mis necesidades.

¿Qué me enamoró de Python? Lo fácil que es el lenguaje, cómo funciona, el ecosistema que hay alrededor, la comunidad, todo lo que puedes hacer con él. Es un todo. A lo largo de mi vida he programado en muchos lenguajes, principalmente en *Java*, *Objective-C*, *C#* y *PHP*. Con ninguno de ellos sentí que tenía el poder. Sin embargo, con Python las sensaciones fueron muy diferentes.

Como todo en la vida es práctica, me llevó un tiempo aprender realmente bien el lenguaje y todo lo que se mueve a su alrededor: el intérprete, las implementaciones, manejo de dependencias, ...

Probablemente, estas cosas no te suenen ahora mismo, pero le vamos a poner remedio. Te voy a enseñar a programar en Python como a mí me hubiera gustado que me hubiesen enseñado.

Yo diría que el curso abarca el 90 % del código Python que normalmente usarás en cualquier aplicación. Los temas van gradualmente de principiante a experto así que, revísalo entero y cuando no sepas o no entiendas cómo hacer algo, siempre puedes volver y consultar un tema o una clase concretos.

Y ya no te entretengo más que seguro que estás con ganas de empezar.

¡Muchas gracias por haber decidido aprender Python conmigo y **BIENVENID@**!

Objetivos del curso

Los objetivos del curso *Python – Guía para ser un Pythonista* son los siguientes:

- Aprender el lenguaje Python, su sintaxis y las instrucciones más importantes.
- Conocer los principales tipos de datos que ofrece el lenguaje y cómo y cuándo usar cada uno de ellos.
- Conocer cómo funciona Python.
- Saber cómo ejecutar un programa escrito en Python.
- Aprender la diferencia entre Script y módulo.
- Conocer cómo crear entornos virtuales e instalar dependencias.
- Descubrir conceptos avanzados del lenguaje.
- Que puedas realizar cualquier programa de consola que te propongas.
- Convertirte en un auténtico Pythonista.
- Ser el punto de partida para que, posteriormente, te puedas dedicar al ámbito que prefieras.

Y todo ello lo conseguirás gracias a la metodología que siempre aplico en mis cursos. El curso está dividido en diferentes módulos o temas, en cada uno de los cuales encontrarás: un apartado teórico,

videotutoriales en los que explico los conceptos de la teoría, una parte práctica y ejercicios, muchos ejercicios.

Temario del curso

El curso está dividido en diferentes bloques, compuestos cada uno de ellos por varios temas. Como te adelanté en la introducción, la dificultad de los temas va aumentando a medida que se avanza en el curso, de manera que la progresión será muy gradual.

A continuación, te detallo el temario del curso:

- **Bloque 1: *Bienvenida***
  - Tema 1: *Introducción*
- **Bloque 2: *Introducción a aspectos básicos del lenguaje***
  - Tema 2: *Características básicas del lenguaje*
  - Tema 3: *Variables. Quiero decir... nombres*
  - Tema 4: *Tipos de datos básicos*
  - Tema 5: *Operadores*
  - Tema 6: *Entrada/Salida*
- **Bloque 3: *Sentencias de control de flujo***
  - Tema 7: *Sentencias de control de flujo*
- **Bloque 4: *Tipos de datos contenedor/colección***
  - Tema 8: *Tipos colección*
  - Tema 9: *Tipos secuencia*
  - Tema 10: *Tipo diccionario*
  - Tema 11: *Tipo conjunto*
  - Tema 12: *Tipos para manejo de bytes*

- **Bloque 5: *División de la lógica de un programa***
  - Tema 13: *Funciones*
  - Tema 14: *Espacios de nombres, módulos y paquetes*
  - Tema 15: *Programación orientada a objetos en Python*
- **Bloque 6: *Otros aspectos básicos del lenguaje***
  - Tema 16: *Ejecución de scripts*
  - Tema 17: *Manejo de errores y excepciones*
  - Tema 18: *Lectura y escritura de Ficheros*

Qué es Python

Principalmente, Python es un lenguaje de programación de propósito general de alto nivel (aunque, como te explicaré más adelante, Python es mucho más que un lenguaje).

La filosofía de Python, su máxima, **es que el código** de un programa escrito en este lenguaje **sea fácil de leer**. Y alrededor de esta filosofía giran todos los elementos del lenguaje.

Las principales características de Python son:

- **Es multiparadigma**: Soporta la programación imperativa, programación orientada a objetos y funcional. **En este curso nos vamos a centrar en la programación imperativa y orientada a objetos.**
- **Es multiplataforma**: Puedes utilizar Python en los principales sistemas operativos: *Windows*, *Linux* y mac*OS*. Además, se puede reutilizar el mismo código en cada una de las plataformas.
- **Es dinámicamente tipado**: El tipo de las variables se decide en tiempo de ejecución.
- **Es fuertemente tipado**: No se puede usar una variable en un contexto fuera de su tipo. Si se

quisiera, habría que hacer una conversión de tipos.

- **Es interpretado**: El código no se compila a lenguaje máquina. El intérprete de Python lee un programa escrito en este lenguaje y lo convierte a lenguaje máquina. Entraré más en detalle en esto en una sección posterior.
- **Es multipropósito.** Puedes usar Python para diferentes campos de la informática, como te explicaré en la siguiente sección.

A pesar de ser un lenguaje muy importante hoy en día, no es nada nuevo. Python fue creado por *Guido Van Rossum* a finales de la década de los 80. Su nombre se debe a la afición que tenía *Guido* por el grupo de humoristas británicos *Monty Python*.

Actualmente, el lenguaje es administrado por la *Python Software Foundation* y se distribuye con una licencia de código abierto, conocida como *Python Software Foundation License*, compatible con la *Licencia pública general de GNU*.

Por qué Python está de moda

Como te he mencionado en la sección anterior, Python tiene ya unas cuantas décadas. Sin embargo, muchos

informes y encuestas ponen de manifiesto que Python es uno de los lenguajes con mayor tasa de crecimiento en estos últimos años. De hecho, es usado por las grandes empresas tecnológicas del momento: *Google*, *Facebook*, *Instagram* o *Netflix*. Pero ¿por qué está tan de actualidad?

Las principales razones son las siguientes:

- Es un lenguaje muy fácil de aprender con una sintaxis clara y sencilla.
- Cuenta con una amplia comunidad de desarrolladores y recursos disponibles.
- Permite dedicarte a diferentes campos de la informática que están en auge.
- Es muy recomendable para implementar scripts y automatizar tareas, prácticas imprescindibles para un Devops.
- Cuenta con dos de los frameworks web preferidos por los desarrolladores y más importantes ahora mismo: *Django* y *Flask*.
- Pero lo que ha hecho despertar su interés y que crezca su popularidad enormemente, ha sido su aplicación en campos como el *machine learning* o el *análisis de datos*, gracias, sobre todo, a la aparición de librerías muy potentes para ello: *NumPy, Pandas, Matplotlib* o *Tensorflow*.

Qué se puede hacer con Python

Llegados a este punto, ya te puedes hacer una idea de los principales usos de Python y de por qué debes aprender Python.

No obstante, por si todavía no lo tienes claro, te detallo a continuación los principales campos de aplicación del lenguaje.

Desarrollo Web

Python cuenta con dos de los frameworks de desarrollo web más importantes en la actualidad: *Django* y *Flask*. Esto hace propicio que aprendas Python si te quieres dedicar a este mundo.

¿Para qué puedes usar Django y Flask? Para desarrollo de aplicaciones web, backends, APIs o microservicios.

Scraping

Seguimos en el mundo web. El *scraping* consiste en extraer datos e información de una web analizando su

contenido. Python cuenta con librerías y frameworks muy potentes para ello, como *Scrapy* o *Beautiful Soup*.

Devops

Devops es un término complicado de definir porque nadie se pone de acuerdo. Lo que sí está claro es que es una combinación de desarrollo de software, integración continua, pruebas, artefactos, puesta en producción, monitorización, etc.

La facilidad de Python para crear scripts lo hacen propicio para esta nueva profesión que está de moda.

Desarrollo de microcontroladores

Con Python es posible desarrollar microcontroladores con una versión reducida del lenguaje conocida como *MicroPython* y que compite directamente con *Arduino*.

¿Qué se puede hacer con *MicroPython*? Controlar pines de tarjetas, hacer parpadear un LED, lecturas digitales, generar señales PWM, controlar servomotores, lecturas de señales analógicas, etc.

Machine Learning, Inteligencia Artificial y Análisis de datos

Si hay unos campos que han hecho despegar a Python en los últimos años, esos han sido los relacionados con la ciencia de datos, la inteligencia artificial y el machine learning. Diferentes todos, pero relacionados entre sí.

Así que, si te quieres dedicar a alguno de ellos, es fundamental que aprendas el lenguaje.

Cuando lo hagas, deberás continuar tu formación estudiando alguna de las siguientes librerías o frameworks: *NumPy, Pandas, Matplotlib* o *Tensor Flow*.

Los anteriores son solo algunos ejemplos de campos de aplicación de Python, pero sin ninguna duda que hay más: scripting, automatización de tareas, aplicaciones de escritorio, ... Y a ti, ¿para qué te gustaría utilizar Python?

Versiones de Python

Bueno, una vez que hemos repasado lo que es Python, por qué es un lenguaje que deberías aprender y los principales campos de aplicación, es hora de ir entrando en acción.

En lo que resta de tema vas a ir preparando el entorno de ejecución y desarrollo junto a mí, para que puedas seguir el curso y comenzar a desarrollar tus programas usando el lenguaje.

Lo primero que debes saber es que existen varias versiones de Python.

Las dos principales son *Python 2* y *Python 3*. Python 2 es una versión obsoleta que ya no tiene mantenimiento, así que olvídate de ella. Mi recomendación es que solo uses Python 2 si heredas una aplicación desarrollada en dicha versión que debes mantener y no tienes intención de migrar.

Respecto a Python 3, también existen varias versiones. Las más recientes son Python 3.6, 3.7, 3.8, 3.9 y 3.10. Utiliza una de estas, ya que incluyen mejoras importantes del lenguaje.

Instalar Python

¡Vamos a instalar Python! Tal y como te indicaba al comienzo, Python está disponible para los principales sistemas operativos: *Windows*, *Linux* y mac*OS*.

A continuación, te mostraré cómo instalar Python en cada uno de ellos.

IMPORTANTE: Para el curso vamos a utilizar la versión 3.7 o 3.8 de Python, dependiendo del sistema operativo.

NOTA: En algunos sistemas operativos (generalmente *Linux* y *macOS*) viene preinstalada una versión de Python. Suele ser la versión 2.7. Como te decía, no uses esa versión. Es posible tener varias versiones instaladas al mismo tiempo en tu equipo.

Sin embargo, antes de instalar Python, comprueba si ya está instalado en tu PC y qué versión es. Te explico cómo lo puedes saber.

Abre un terminal y ejecuta el siguiente comando:

```
$> python -V
```

Si ves que en la pantalla se muestra el mensaje *Python* seguido de una versión, entonces Python ya lo tienes instalado en tu sistema.

Normalmente, el comando `python` está relacionado con una instalación de Python 2.x, por lo que es probable que el mensaje haya sido *Python 2.7.x.*

Para comprobar si hay instalada alguna versión de Python 3, ejecuta el siguiente comando en el terminal:

```
$> python3 -V
```

De nuevo, si ves el mensaje *Python 3.x.x*, significa que tienes una versión de Python 3 instalada.

Si al ejecutar cualquiera de los dos comandos anteriores observas el texto *Python 3.7.x o Python 3.8.x*, entonces ya tienes Python correctamente instalado en tu equipo y puedes saltar a la siguiente sección. Si no es así, continúa por el apartado correspondiente a tu sistema operativo.

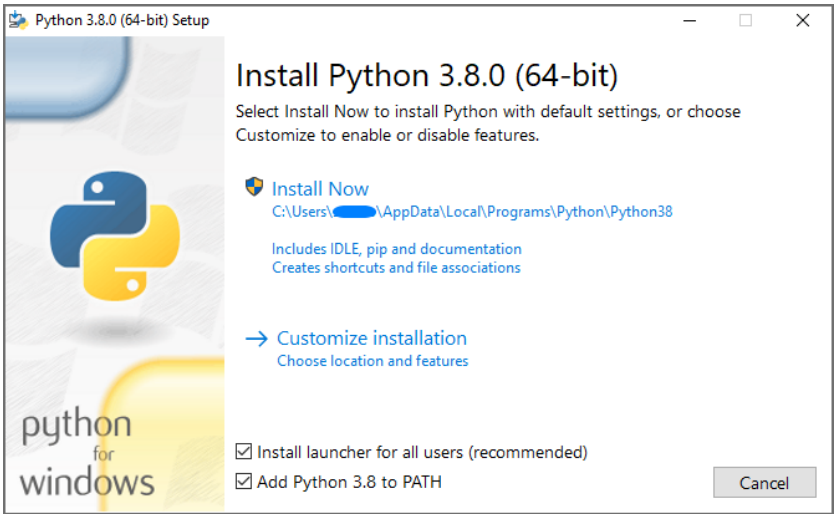
Instalar Python en Windows

Para instalar Python en un entorno *Windows*, descarga el instalador de la última versión (*3.8.x*) desde el sitio web de Python.

Página de descargas de Python.org
https://www.python.org/downloads/

Existen varios instaladores disponibles para la descarga. **Selecciona el instalador completo** correspondiente a la arquitectura de tu ordenador (32 o 64 bits). El instalador completo es aquel que acaba con el nombre *...executable installer*.

Una vez descargado, ejecútalo y sigue los pasos que se indican hasta llegar a la siguiente pantalla:



En esta pantalla tienes varias opciones:

- Realizar una instalación por defecto (*Instalar Ahora*). Esta opción, en principio, no requiere permisos de administrador e instalará Python y los componentes por defecto en tu directorio de usuario.
- Realizar una instalación personalizada. Permite instalar Python para todos los usuarios, seleccionar el directorio de instalación y qué componentes instalar. Es posible que se requieran permisos de administración.
- Marcar las opciones para instalar el *Iniciador de Python* y **añadir Python al PATH**.

IMPORTANTE: Asegúrate de que la opción *Añadir Python al PATH* está marcada. Si no lo haces, es posible que Python no funcione como esperas y tendrás que añadir manualmente algunos directorios a la variable de entorno PATH con posterioridad.

En tu caso, si eres principiante, **selecciona la instalación por defecto** y recuerda marcar las opciones para instalar el *Iniciador de Python* y *añadir Python al PATH*.

NOTA: El iniciador de Python para Windows es una utilidad que ayuda a localizar y ejecutar diferentes versiones de Python que haya disponibles en tu ordenador.

Una vez seleccionadas las opciones oportunas, continúa con la instalación.

Instalar Python en Linux (Ubuntu/Debian)

A continuación, te mostraré cómo instalar Python en una distribución de Linux correspondiente con *Ubuntu/Debian*.

Para instalar Python en *Ubuntu/Debian*, debes instalar los paquetes *python3*, *python3-dev* y *python3-venv*.

El primer paso que debes dar es actualizar los repositorios. Ejecuta lo siguiente en un terminal con un usuario con permisos de *sudo*:

```
$> sudo apt update
$> sudo apt -y upgrade
```

A continuación, procede a instalar los paquetes correspondientes a Python. De nuevo, ejecuta en el terminal lo siguiente:

```
$> sudo apt install python3 python3-dev python3-venv
```

Para comprobar que la instalación se ha realizado correctamente, ejecuta el siguiente comando:

```
$> python3 -V
```

Debe aparecerte el mensaje *Python 3.7.x o Python 3.8.x.*

Por último, tienes que instalar *pip*. *Pip* es un gestor de paquetes y dependencias de Python. Hablaré de él más adelante en el curso. Por ahora no te preocupes si no sabes lo que es, simplemente instálalo del siguiente modo:

```
$> wget https://bootstrap.pypa.io/get-pip.py
$> sudo python3 get-pip.py
```

Comprueba que pip se ha instalado correctamente ejecutando el siguiente comando en la terminal:

```
$> pip3 -V
```

En la salida, debes comprobar que *pip* se encuentra instalado en una ruta dentro de *python3.7 o python3.8*. Si no es así, algo has hecho mal.

Instalar Python en macOS

Existen diferentes modos de instalar Python en *macOS*. Aquí te voy a explicar dos de ellos: a través del gestor de paquetes *Homebrew* o descargando un paquete de la página oficial.

MacOS suele incluir Python 2.7 para sus propios propósitos. Como te decía, ignóralo.

Instalar Python con Homebrew

Homebrew es un gestor de paquetes que permite instalar desde el terminal herramientas y complementos que no vienen de serie con macOS.

En el momento de escribir este curso, el único inconveniente de instalar Python con *Homebrew* es que, en principio, en sus repositorios solo dan soporte hasta la versión 3.7.x. Tratar de instalar Python 3.8 es complicado usando esta herramienta. Así que, si quieres la última versión del lenguaje, mejor consulta la siguiente sección.

Para instalar Python 3.7.x usando *Homebrew*, lo primero que debes hacer es instalar las *herramientas de línea de comandos* de *Xcode*. Para ello, ejecuta lo siguiente desde el terminal:

```
$> xcode-select --install
```

Una vez instaladas, comprueba si ya dispones de *Homebrew* en tu sistema ejecutando lo siguiente:

```
$> brew --version
```

Si te aparece un mensaje con la versión de *Homebrew*, entonces continúa por el siguiente paso. Si no, accede a la página de la herramienta en la que se indica cómo instalarla.

Página de Homebrew
https://brew.sh

Bien, una vez que dispongas de *Homebrew*, ya puedes instalar Python 3 del siguiente modo:

```
$> brew install python
```

El comando anterior instalará la última versión de Python y de *pip* disponible en los repositorios de *Homebrew*.

Instalar Python desde la página oficial

Desde la página oficial de Python es posible instalar la última versión (3.8.x). Accede a la web y descarga el instalador correspondiente.

Página de descargas de Python.org
https://www.python.org/downloads/

Una vez descargado el instalador, ejecútalo y sigue los pasos que te indica.

Por defecto, Python se instalará en la siguiente ruta `/Library/Frameworks/Python.framework`. Además, se añadirá un enlace simbólico al ejecutable de Python en `/usr/local/bin/`.

Al finalizar la instalación con cualquiera de las dos opciones (*Homebrew* o paquete de la página oficial),

comprueba que la instalación se ha realizado correctamente, ejecutando lo siguiente en el terminal:

```
$> python3 -V
```

Debe aparecerte el mensaje *Python 3.7.x o Python 3.8.x*, respectivamente.

Comprueba que *pip* también se ha instalado ejecutando lo siguiente:

```
$> pip3 -V
```

Debes ver un mensaje con la versión de *pip*.

El intérprete de Python

Justo en la sección ***Qué es Python*** indicaba que Python es mucho más que un lenguaje de programación. ¿Qué quería decir con esto?

Voy a tratar de explicártelo lo más fácilmente posible. Quizá esta sección no te la expliquen en otros cursos o no la hayas visto tan en detalle, pero si quieres ser un verdadero Pythonista, debes entender realmente qué es Python y cómo funciona un programa escrito en este lenguaje.

Es probable que no entiendas muchos conceptos de los que aquí te explique si eres nuev@ en Python. Si es así, no dudes en volver a esta sección cuando hayas acabado el curso.

Bueno, como te iba diciendo, **Python, el lenguaje, es una especificación**. Una interfaz que indica qué elementos componen el lenguaje, lo que deberían hacer y cómo deberían comportarse.

Por otro lado, hemos visto que **Python es un lenguaje interpretado**. Esto significa que hace falta un **intérprete** que entienda un programa escrito en Python para que finalmente pueda ser ejecutado en una

computadora. Técnicamente, el intérprete es una capa de software que funciona entre tu programa y el hardware de tu ordenador para ejecutar tu código.

Como Python, el lenguaje, es una interfaz (indica qué se debe hacer), dicha interfaz necesita de una implementación (cómo se hace). La implementación por defecto de Python, la principal y la que probablemente uses, es conocida como *CPython*. Esta implementación, que viene a ser el propio intérprete, es básicamente un programa escrito en el lenguaje *C*.

En definitiva, Python es la combinación del lenguaje y su implementación, es decir, el intérprete. De hecho, cuando instalas Python de la página oficial, entre otras cosas, lo que haces es instalar el intérprete *CPython*.

El nombre *CPython* se utiliza para distinguirlo de otras implementaciones y para diferenciar la implementación del propio lenguaje de programación Python.

¿Qué otras implementaciones del lenguaje existen? Las alternativas a la implementación "oficial" más conocidas son las siguientes:

- *Jython*, un intérprete implementado en Java.
- *IronPython*, un intérprete implementado en C#.

- *PyPy*, un intérprete implementado en *RPython* (un subconjunto de Python estáticamente tipado).

En este curso nos centraremos exclusivamente en *CPython*, de manera que cuando hable de Python, me estaré refiriendo al lenguaje y, en su defecto, a esta implementación.

¿Cómo funciona el intérprete de Python?

Cuando ejecutas un programa escrito en Python, el intérprete genera un código intermedio conocido como *bytecode*. Tras esto, el intérprete entra en un bucle en el que va leyendo, interpretando y ejecutando el *bytecode* sobre la marcha.

La generación del *bytecode* es una especie de compilación, pero no te confundas. En un lenguaje compilado, un compilador genera *código máquina* que puede ser ejecutado directamente por el procesador.

Sin embargo, el *bytecode* es una representación interna de un programa Python producida por el intérprete. Generalmente, la primera vez que se ejecuta un programa, el *bytecode* se almacena en un archivo de caché con extensión `.pyc`. Esto permite que las sucesivas ejecuciones del programa sean más rápidas, puesto que no es necesario volver a generar el *bytecode*.

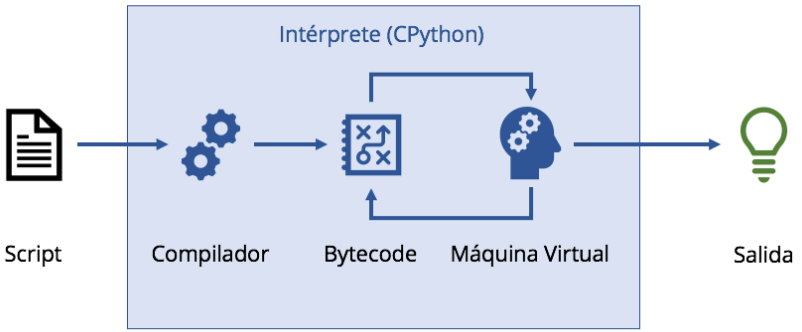
Por tanto, si te lo estabas preguntando, no, ***CPython* no traduce un programa a código *C***, sino a *bytecode*.

Finalmente, las fases que sigue el intérprete para ejecutar un programa en Python son las siguientes:

- Comienza leyendo las instrucciones del programa de manera secuencial.
- El código se compila a *bytecode*. Realmente, el *bytecode* se genera solo para los módulos y no para los scripts ejecutables (hablaremos de esto en el tema siguiente).
- El *bytecode* se envía para su ejecución.
- En este momento entra en acción la *Máquina Virtual de Python*. Esta máquina virtual es el motor de tiempo de ejecución de Python. Es un bucle que itera sobre las instrucciones en *bytecode* y las ejecuta una por una.

El ciclo anterior se conoce como el *Modelo de Ejecución de Python* y lo puedes ver en el siguiente diagrama:

Modelo de Ejecución de Python



Como puedes observar, el intérprete en sí está compuesto por distintos elementos. Los más importantes son el *compilador* y la *máquina virtual*.

NOTA: Según la documentación oficial de Python, no se espera que el *bytecode* funcione entre diferentes máquinas virtuales de Python, ni que sea estable entre las versiones de Python.

Pues ahora sí que puedes decir que entiendes cómo se ejecuta un programa en Python. Eso que todavía no

hemos visto ninguno, jaja. No te preocupes, estamos sentando las bases para los próximos temas.

Y bien, una vez que ya sabes lo que es el intérprete, solamente me queda decirte que este puede ejecutar código Python de dos formas diferentes:

- Como un script o módulo.
- Como un fragmento de código escrito en una sesión interactiva.

Pero esto lo veremos en el tema siguiente.

¿Dónde vamos a trabajar?

Pues hemos llegado a la última sección del tema de introducción (introductorio pero intenso, jaja).

En esta sección simplemente te voy a dar unas recomendaciones de qué aplicaciones puedes usar para desarrollar en Python.

Si eres nuev@ en esto de la programación, debes saber que para programar se utilizan editores específicos para ello, del mismo modo que para escribir un documento se utiliza, por ejemplo, *Microsoft Office* o *Libre Office*.

Si eres nuev@ en Python y no conoces ninguno, te diré que tienes decenas de alternativas. No obstante, para mí, las mejores actualmente son *PyCharm* y *Visual Studio Code*. Por tanto, te recomiendo cualquiera de los dos para realizar este curso.

Si ya has programado con Python y tienes un editor de preferencia, sigue con él si así lo deseas.

A continuación, te mostraré las características de cada uno, qué ventajas tienen y una configuración básica para comenzar a trabajar.

PyCharm

Sin duda alguna, desde mi punto de vista es el mejor de todos si quieres programar en Python. Yo es el que uso personalmente y el que utilizaré durante el curso, pero como te decía, tú puedes usar el que prefieras.

Ediciones de PyCharm

PyCharm está disponible para *Windows*, *Linux* y *macOS*.

Existen tres ediciones del IDE: *Community*, *Edu* y *Professional*.

Para comenzar, puedes optar por la versión *Community*. Es de código abierto y gratuita y tiene opciones y herramientas más que suficientes para ayudarte durante cada etapa del desarrollo.

Requisitos mínimos

El único "pero" es que para que vaya fluido necesitas tener un PC con, al menos, 8 GB de memoria RAM.

Este es su principal punto en contra. Su potencia consume algunos recursos, más que *Visual Studio Code*.

Si tu PC no cumple ese requisito, te recomiendo que no lo utilices.

Características principales

PyCharm es un IDE todoterreno que gana en muchos aspectos a otros IDEs y editores de texto, especialmente su inspector de código.

Las principales características de este son:

- **Editor inteligente.** Sin duda, el mejor editor para Python. Sus principales funcionalidades: Resaltado de sintaxis, sangría automática y formateo de código, finalización de código, formateador de código, resaltado de errores sobre la marcha, análisis del código a medida que se escribe y detector de código duplicado.
- **Depurador gráfico en local.**
- **Refactorización**.
- **Navegación inteligente por el código.**
- **Integración con sistemas de control de versiones:** *Git, Mercurial, CVS, Subersion, GitHub*.
- **Ejecución de un plan de pruebas.**
- **Integración con documentación:** *reStructuredText y Google*.
- **Integración con *PyQt y PyGTK*.**

- **Gestión de paquetes**.
- **Integración nativa con *Virtualenv/Buildout*.**
- **Consola de Python.**
- **Soporte para *XML, HTML, YAML, JSON, RelaxNG.***
- **Terminal local.**

No te agobies si ahora mismo no entiendes nada, solo quería mostrarte de lo que es capaz.

En principio, no tienes que instalar nada adicional para comenzar a trabajar con *PyCharm*. Otra ventaja. Viene con todo lo que necesitas para ponerte manos a la obra.

Si te decides por este IDE, puedes descargarlo desde su página web (recuerda descargar la edición *Community*):

Página de descargas de PyCharm
https://www.jetbrains.com/es-es/pycharm/download/

Visual Studio Code

Por su parte, Visual Studio Code es un editor de código desarrollado por Microsoft que se puede utilizar para programar en diferentes lenguajes de programación.

Sin embargo, en caso de que quieras usarlo para desarrollar en Python, es conveniente que instales una serie de plugins y librerías adicionales que lo dotan de nuevas funcionalidades y características para trabajar adecuadamente con el lenguaje.

Requisitos mínimos

En la página del editor se definen los siguientes requisitos mínimos que debe tener un PC:

- Un procesador de 1.6 GHz
- 1 GB de memoria RAM

Características principales

A continuación, te enumero las características principales del editor:

- **Multiplataforma:** Existe una versión de *VS Code* para los principales sistemas operativos (Windows, Linux y macOS).
- **Completado de código (IntelliSense).**
- **Depurador.**
- **Características de edición avanzadas.**
- **Navegación por código y refactorización**
- **Sistema de plugins (extensiones) para añadir nuevas características y funcionalidades al editor.**

Configuración para trabajar con Python

Como te he indicado, para desarrollar en Python es aconsejable que instales una serie de extensiones y librerías que doten al editor de características adicionales a las que ofrece por defecto. Son las siguientes:

- **Extensiones**:
  - *Python extension for Visual Studio Code*
- **Librerías**:
  - Pylint: *Analizador de código estático*

En caso de que decidas usar Visual Studio Code, puedes descargarlo desde su página web:

Página de descargas de VS Code
https://code.visualstudio.com/

Con esto llegamos al final de este capítulo. En el siguiente tema te mostraré cómo usar cada uno de los IDEs, cómo crear tus programas en Python y cómo puedes ejecutarlos.