

# **PROGRAMACIÓN CON CÓDIGO: PYTHON**

## **CONCEPTOS PREVIOS E INSTALACIÓN**

Este documento es de uso único e intransferible para el alumno matriculado en el curso. Cualquier reproducción física o digital del documento sin permiso de los autores vulnera los derechos de propiedad intelectual de los mismos.

## INDICE

<b>INDICE.....</b>	<b>3</b>
<b>1. ¿QUE ES PYTHON? .....</b>	<b>4</b>
1.1 El Zen de Python .....	6
<b>2. ¿QUÉ VOY A APRENDER EN ESTE CURSO? .....</b>	<b>8</b>
<b>3. ¿CÓMO ES UN TEMA O UNIDAD?.....</b>	<b>10</b>
<b>4. ¿CÓMO SE APRENDE A PROGRAMAR?.....</b>	<b>12</b>
<b>5. INSTALANDO PYTHON Y THONNY.....</b>	<b>15</b>
5.1 Windows.....	16
5.2 Mac.....	16
5.3 Linux.....	19
<b>6. USANDO LA IDLE DE PYTHON 3 .....</b>	<b>21</b>
<b>7. USANDO THONNY .....</b>	<b>23</b>
<b>8. EDITOR DE CÓDIGO.....</b>	<b>25</b>
<b>9. ¡A PROGRAMAR!.....</b>	<b>28</b>

## 1. ¿QUE ES PYTHON?

Los ordenadores, teléfonos móviles y tablets, entre muchos otros dispositivos electrónicos, funcionan gracias a instrucciones precisas. Los programadores escriben esas instrucciones mediante el uso de lenguajes de programación. Los lenguajes de programación son formas de expresar lo que deseamos que haga la máquina y, a su vez, la máquina puede traducir esos deseos a un lenguaje que pueda ejecutar. Digamos que un lenguaje de programación es un punto intermedio entre el lenguaje humano y el lenguaje máquina, siendo comprensible por ambos.

Python es un **lenguaje de programación**. Los lenguajes de programación son lenguajes que pueden ser interpretados por máquinas y a su vez pueden ser leídos y escritos por humanos. Evidentemente, es necesario comprender el lenguaje para poder leerlo y escribirlo, de la misma forma que es necesario saber un idioma para hablarlo y entenderlo.

Como lenguaje de programación, Python tiene muchas ventajas frente a otros lenguajes habitualmente usados (C, C++, Java, JavaScript...). La principal de todas es una consecuencia lógica de cómo fue creado.

Python surgió a finales de los ochenta, creado por Guido van Rossum, con la intención de convertirse en un lenguaje de programación con una sintaxis que favoreciese un código legible. Es decir, Python se caracteriza por ser un lenguaje fácil de leerse y escribirse. Los programadores usan a veces la expresión “Python se lo traga todo”, ya que en Python son válidas muchas expresiones que darían error en otros lenguajes.

Esto no quiere decir que Python sea un lenguaje para aprender a programar, para niños o sin potencial. Python es un lenguaje de programación **multipropósito**, podemos utilizarlo para programar aplicaciones, videojuegos, animaciones, páginas web, simples scripts de texto... Es muy usado en el entorno profesional y en el educativo.

Así mismo, Python es un lenguaje pensado para ser interpretado, es decir, poder ser ejecutado directamente mediante un intérprete sin tener que compilarlo y convertirlo a lenguaje máquina. Si no has entendido correctamente este párrafo no te preocupes, no es determinante ahora mismo.

Desde mi experiencia, Python resulta un lenguaje tremendamente útil y sencillo para múltiples aplicaciones y permite centrarse más en **cómo se ejecuta** y menos en **cómo se escribe correctamente**. Mi experiencia con otros lenguajes me dice que se emplea mucho tiempo en aprender la sintaxis, por lo que los cursos sobre los mismos suelen enseñar poco a cómo pensar y programar y se centran en aprender a escribirlo, es decir, picar código (expresión que se usa en programación para definir al hecho de programar en un lenguaje concreto sin hacer cosas realmente interesantes).

Esto no quiere decir que con Python no sea necesario emplear tiempo en escribir código y repetir instrucciones una y otra vez. Practicar y repetir es una clave para interiorizar y mecanizar ciertas instrucciones que acaban acoplándose a la forma de pensar del programador y teniendo como resultado una fluidez y soltura a la hora de programar.

Al aprender Python, el usuario se puede centrar más en ver la utilidad que en aprender cientos de cosas que no aparentan utilidad ninguna. Digamos que facilita un aprendizaje similar al que se da cuando se aprende a hablar: primero aprendes a decir cosas útiles y posteriormente ya vas interiorizando la gramática y cómo se estructura el lenguaje, pero lo primero es decir “hambre”, “sueño”, “pis”...

Python debe su nombre a la admiración que su creador profesaba al grupo cómico inglés Monty Python, de ahí que muchos cursos y recursos en la red utilizan clichés de la filmografía del grupo. Es normal encontrar expresiones como **ní** (Los Caballeros de la Mesa Cuadrada) o **Pijus Magnificus** (La Vida de Bryan).

Resumiendo, Python es un lenguaje creado para facilitar la legibilidad y la calidad del código creado, por lo que profesionalmente es un lenguaje muy demandado y usado. Grandes empresas como Facebook, Google, Instagram o Spotify utilizan Python profesionalmente. Muchas plataformas educativas y empresas de software posicionan Python como la mejor herramienta para aprender en el año 2019, tendencia sin visos de cambiar a corto plazo.

## 1.1 El Zen de Python

Una de las cosas más valiosas de Python es la filosofía de desarrollo que lo acompaña. Dicha filosofía vela por la sencillez, la limpieza, la calidad y atributos similares. Por ello, los desarrolladores de Python se guían por una serie de principios que se pueden resumir en una lista llama El Zen de Python. La lista marca una serie de principios a seguir a la hora de programar en código Python:

1. *Hermoso es mejor que feo.*
2. *Explícito es mejor que implícito.*
3. *Simple es mejor que complejo.*
4. *Complejo es mejor que complicado.*
5. *Sencillo es mejor que anidado.*
6. *Escaso es mejor que denso.*
7. *La legibilidad cuenta.*
8. *Los casos especiales no son lo suficientemente especiales para romper las reglas.*
9. *Lo práctico le gana a la pureza.*
10. *Los errores no debe pasar en silencio.*
11. *A menos que sean silenciados.*
12. *Encara a la ambigüedad, rechazar la tentación de adivinar.*
13. *Debe haber una - y preferiblemente sólo una - manera obvia de hacerlo.*
14. *Aunque esa manera puede no ser obvia en un primer momento a menos que seas holandés.*
15. *Ahora es mejor que nunca.*
16. *Aunque "nunca" es a menudo mejor que "ahora mismo".*
17. *Si la aplicación es difícil de explicar, es una mala idea.*
18. *Si la aplicación es fácil de explicar, puede ser una buena idea.*
19. *Los espacios de nombres son una gran idea ¡hay que hacer más de eso!*

Como podrás ver, Python tiene asociada una filosofía que le quita seriedad a la forma de transmitir las cosas (véase el punto 14 del Zen de Python). Cada punto de los anteriores tiene su explicación y sentido, pero es complejo y complicado entenderlos sin saber nada de programación, así que de momento te pido que recuerdes que existe una filosofía de desarrollo que permite hacer de Python un lenguaje universal y accesible.

## 2. ¿QUÉ VOY A APRENDER EN ESTE CURSO?

En este curso nos vamos a centrar en ver las instrucciones básicas de Python (que comparte con muchos otros lenguajes de programación), enmarcándolas en casos prácticos y útiles. Esto no quiere decir que sean útiles para todo el mundo, es complicado encontrar una aplicación que sea universal para cada concepto, pero seguro que podrás ver que lo que haces puede ser útil en algún contexto.

Por ello, es importante que interiorices lo que se explica a continuación para no sentir angustia ni decepción a lo largo del curso:

- Vas a aprender gradualmente todo lo básico que requiere cualquier lenguaje de programación, pero quizá no nos centraremos tanto en cada una de las opciones de cada uno de los puntos tratados sino más bien en entender su utilidad. Aprenderemos a buscar información de cómo seguir aprendiendo y aplicando cada punto.
- Por suerte, Python es un lenguaje muy usado por programadores, makers y personas que apoyan el mundo del software libre, por lo que internet está colmado de páginas de ayuda y consejos así como de ejemplos resueltos. Es importante que acudas a internet siempre que tengas alguna duda, es la forma de aprender a solucionar aquellas inquietudes que te surgirán tras acabar el curso.
- Algunas unidades van a ser más centradas en un concepto y otras algo más transversales, tocando varios conceptos debido al ejercicio que se resuelve en la misma, pero en todas se trabajarán elementos necesarios para programar.
- A lo largo de los temas se plantearán retos o se pedirá que intentes cosas antes de continuar. La única forma de encontrarse con dificultades y estar preparado para adquirir conocimiento es desde el interés y para ello es necesario intentarlo primero para ver dónde hay dificultades. Es muy recomendable tomarse en serio trabajar las unidades intentando aquellas cosas que se propongan a lo largo de la teoría.
- Los vídeos son breves apoyos para asentar un poco mejor conceptos clave, pero la unidad central es el documento escrito. Intenta trabajarlo a fondo y, tras ello, antes de realizar el test y las actividades, podrás ver los vídeos para asentar. En algunas unidades se comentará el momento en que deberías ver algún vídeo explicativo.



- Aprender realmente a programar requiere mucho tiempo y trabajo. Es una labor bonita y generalmente muy motivadora al principio, pero requiere batallar una y otra vez con ciertos problemas, volver a incurrir en errores y tener **paciencia**. Hoy en día, en el medio tecnológico que vivimos, la paciencia es un don que empieza a escasear y sin ella programar se convierte en un tormento. Disfruta de conseguir solucionar tus propios problemas, disfruta del proceso de aprendizaje, de mejorar tus propios programas...

Una vez terminado el curso serás capaz de usar directamente Python, de crear archivos de Python que se ejecuten, de utilizar archivos de Python y archivos de texto de manera coordinada y de usar muchas de las funcionalidades de Python que han desarrollado terceras personas y han puesto al servicio de la comunidad a través de la red.

### 3. ¿CÓMO ES UN TEMA O UNIDAD?

Cada tema (a excepción de éste primero, por ser un tema introductorio) tiene la misma estructura:

- Un documento de teoría que incluye ejercicios resueltos y a lo largo del cual se plantean preguntas breves y pequeños retos.
- Una serie de vídeos de apoyo en los cuales se explican con detenimiento algunos de los conceptos relevantes del tema tratado.
- Una recopilación de todas las instrucciones de código nuevas trabajadas en la unidad, que se encuentra al final del documento de teoría.
- Uno o dos retos para trabajar sobre los conceptos desarrollados. No es obligatorio entregarlos pero seguramente realizar estos retos marque la diferencia entre asentar realmente o no los contenidos de la unidad. Si tienes problemas con ellos usa el foro o el correo interno para solucionarlos.
- Una serie de preguntas tipo test para ver si tienes claros algunos conceptos clave.

Algunos temas son bastante abiertos y otros son procesos bastante más guiados, pero siempre plantearán retos o preguntas para que tengas que poner de tu parte en el aprendizaje. No se trata de aprender a escribir lenguaje Python, se trata de aprender a resolver problemas programando con lenguaje Python, y ello conlleva desarrollar una forma de pensar estructurada y metódica.

Por último, se ha trabajado el lenguaje de los textos enfocándolo a personas que no saben programar, por ello se ha incurrido en algunos aspectos que podrían considerarse problemáticos a los ojos de un programador experimentado pero que facilitan el aprendizaje:

- Al programar, se usa el castellano. Sí, las instrucciones de código son en inglés, pero todo lo que podemos definir como usuarios (nombres, descripciones...) están en castellano. No es una práctica recomendable porque muchos lenguajes de programación dan errores con caracteres no anglosajones (la ñ, por ejemplo) pero para un principiante facilita la comprensión del código. Además, va en contra de uno de los puntos del Zen de Python, ya que lo habitual y general es programar en inglés, y la excepción de hacerlo en castellano va en contra de lo habitual.

- No hay comentarios en muchos programas. Los comentarios son descripciones que hace el usuario en el programa y que Python no tratará de entender, sirven para aclarar para qué sirve cada parte. La ausencia de los mismos se debe a que quiero que entiendas el código puro, sin explicaciones del mismo. No es buena práctica programar sin comentar porque al final no entiendes ni tú mismo qué hace cada parte del código, así que trata de comentar tus propios programas. Los míos por lo general están huérfanos de comentarios para obligarte a entender el código puro. Con el tiempo te recomiendo que empieces a incluir tus comentarios para posteriormente entender que hacía cada parte de tu programa. Algunos programadores dicen que los códigos deberían ser auto explicativos, de forma que el nombre de los elementos que usemos nos aclaren qué hace cada elemento.
- Se usan descripciones sencillas y no estrictamente correctas. A veces es más complicado entender ciertas expresiones que entender lo que significan, por lo que no he puesto énfasis en llamar a cada cosa con su nombre correcto y preciso, muchas veces se usan palabras que tienen un significado más claro para el usuario no experto en programación, aunque algún avezado programador pueda poner el grito en el cielo. En el futuro seguramente cambies la forma de llamar a ciertas instrucciones, procesos, etc., pero actualmente es mejor centrarse en aprender cómo programar con Python y el pensamiento computacional, que es un tipo de pensamiento enfocado a secuenciar los problemas y abordarlos por fases, algo muy habitual en el mundo de la ingeniería.
- Intenta guardar TODOS los programas que hagas, organiza una carpeta y ve almacenándolos por temas de manera que los tengas accesibles, seguro que podrás reutilizar muchos de ellos. Utiliza un sistema de versiones. Si vas a cambiar algo sustancial en tu programa guárdalo como una nueva versión, de forma que puedas volver a versiones anteriores y esa ingeniosa idea resulta ser catastrófica (cosa que ocurre demasiado a menudo).

## 4. ¿CÓMO SE APRENDE A PROGRAMAR?

Por encima de todo, estoy muy en contra de los cursos que enseñan instrucciones de código con ejemplos absurdos e inútiles. Es cierto que inicialmente es complicado proponer ejemplos útiles pero siempre se puede indicar cómo podría ser útil en un futuro más complejo.

Para aprender a programar hay que aprender a pensar cómo lo hace una máquina, es decir, **pensar computacionalmente**. Esto significa que hay que pensar en etapas y procesos, lo cual siempre tiene dos partes:

- A alto nivel, es decir, sin profundizar mucho, hay que identificar las partes individuales que, en caso de ser resueltas, consiguen resolver el problema. Por poner un ejemplo, si existen humedades en un piso bajo cubierta un ser humano pensará en los pasos a seguir de forma general: Localizar la zona → Acometer una obra para impermeabilizar la cubierta → Comprobar que los resultados son los esperados.
- A bajo nivel, es decir, profundizando ya en el lenguaje de programación, hay que interpretar mentalmente las secuencias que resuelven cada parte del problema que hay que resolver, especialmente en qué orden hay que ejecutarlas. En el ejemplo anterior tendríamos una serie de pasos bien definidos para localizar la zona donde se produce la humedad y los motivos por los que se produce. Posteriormente tendríamos también una secuencia lógica y definida de pasos para poder reparar el problema y por último existen también métodos concretos y definidos para ver que el problema ya no existe.

Pongamos un ejemplo de cómo se resuelve un problema en la vida cotidiana. Problema: no tengo dónde poner todos mis libros. Lo cual quedaría resuelto si consiguiese una estantería, así que voy a construirla.

### **Solución a alto nivel:**

1. Ir a comprar materiales.
2. Construir la estantería.
3. Ubicarla en el sitio idóneo.
4. Colocar los libros.

Una vez he realizado esta estructura (habitualmente mental) tengo que pensar si cada una de las etapas puede ser resuelta de forma individual (por ejemplo, si quiero derrotar a Superman y necesito Kryptonita no tengo que pensar en solucionar el problema si tengo claro que no puedo conseguir Kryptonita). Si una etapa no puede ser resuelta tal como está planteada se puede buscar una alternativa y si existe seguirá siendo un problema con solución.

Este proceso facilita mucho la asimilación del problema y ayuda a enfocarlo por etapas.

### **Solución a bajo nivel:**

1. Iré a comprar los materiales al centro de bricolaje que más cerca tengo.
2. Compraré tablas de madera para las baldas, tablones largos para la estructura y clavos y tornillos para sujetarlo todo.
3. Compraré barniz para proteger la madera.
4. Construiré primero la estructura.
5. Ubicaré la estantería en su lugar antes de colocar las baldas.
6. Colocaré las baldas y las aseguraré antes de utilizarlas.
7. Organizaré los libros por temáticas.
8. Colocaré los libros de forma que cada temática esté en una zona de la estantería...

Puede parecer un proceso simplón o un ejemplo absurdo, pero la realidad es que un programador desarrolla una capacidad innata para pensar en etapas. Es mucho más importante desarrollar esta forma de pensar que aprender un lenguaje concreto. Por ello, es recomendable profundizar en un único lenguaje y no aprender muchos de forma poco inmersiva.

El pensamiento computacional es una competencia muy positiva para la vida, más allá de la programación, pues te permite enfocar los problemas por etapas y con tranquilidad. Tradicionalmente se ha comentado siempre que es una habilidad más propia de la ingeniería pero no deja de ser una habilidad muy útil en múltiples situaciones de la vida.

Otra cosa importante para aprender a programar es entender que sólo la práctica afianza las competencias en un lenguaje y sólo la práctica acaba desarrollando la capacidad de identificar cuando un lenguaje de programación puede ayudarte para resolver un problema.

Es importante para aprender a programar aprender a **cambiar de actividad**. Muchas veces te atascarás en un problema y no encontrarás solución alguna y en esa situación a veces conviene levantarse a beber agua, dar una vuelta, ir a hablar a alguien sobre el problema... de esta forma puedes encontrar una solución que no se te ocurre en el fragor de la batalla. Los estadounidenses denominan a esto “pensar fuera de la caja”, ya que muchas veces desde dentro del problema no ves su solución.

Por último, el mundo necesita programadores, sí, pero cada vez hay más. Lo que el mundo necesitará muy pronto son ideas más que programadores. Es cierto que ahora mismo hacen falta programadores, hay déficit de gente que sepa escribir código, pero no es menos cierto que por aprender a programar no vas a tener la vida resuelta. Hace falta **aprender qué programar**. Intenta desarrollar tu creatividad a la hora de programar, cuando aprendas algo, prueba esas ideas locas que se te ocurran para exprimirlo, busca tus propios ejemplos aplicaciones, trata de jugar y divertirte con la programación por medio de Python.

## 5. INSTALANDO PYTHON Y THONNY

Antes de empezar, voy a darte un consejo. Aunque el presente curso vas a poder realizarlo en Mac, Windows o Linux, sería una idea maravillosa conseguir poder realizarlo en un sistema operativo libre Linux, ya que es donde más sencillo es programar con Python, un lenguaje libre, por lo que la mayoría de los desarrolladores que han ido trabajando en mejorar Python lo han hecho desde un sistema operativo Linux. Quizá puedas hacer una partición en tu disco duro e instalar una distribución de Ubuntu (el sistema operativo basado en Linux más usado).

Ubuntu es un sistema operativo maravilloso, aunque en el mundo de la construcción, arquitectura y similar no está muy extendido porque la mayoría de los programas profesionales (AutoCAD, Presto, Revit...) no tienen versiones para sistemas operativos basados en Linux.

Actualmente coexisten dos versiones de Python, que son usadas y tienen ligeras diferencias en la forma de escribir instrucciones y en la cantidad de recursos de los que disponen.

Por un lado está **Python 2**, que actualmente (junio del 2019) está en su versión 2.7.16. Es una versión muy utilizada, especialmente por la gran cantidad de bibliotecas y recursos creados para ella.

Por otro lado está **Python 3**, que actualmente (junio del 2019) está en su versión 3.7.3. Está siendo cada vez más usada pero todavía existen bibliotecas de Python 2 que no se encuentran para Python 3. Aun así, por ser la última versión y la que a la larga se impondrá como versión única, en este curso vamos a utilizar siempre y únicamente Python 3.

Para usar Python en Mac y Linux tenemos la opción de instalarlo y lo reconocerá el propio sistema operativo, pudiendo ejecutarlo desde Terminal. Si lo queremos utilizar en Windows necesitaremos un programa denominado IDLE (entorno de desarrollo integrado) que nos permite ejecutar programas con lenguaje Python o bien utilizarlo a través del símbolo del sistema. Esta última opción también existe para Mac y Linux.

Además, especialmente en los primeros pasos del aprendizaje de un lenguaje de programación, es buena práctica utilizar algún programa que ayude a entender los programas y ver cómo se ejecutan paso a paso, depurando los errores que se puedan cometer y ayudando a identificar en qué punto del programa se producen.

Por ello en muchas ocasiones vamos a usar un programa llamado Thonny. Thonny es un entorno de desarrollo integrado para Python que está diseñado para principiantes. Admite diferentes formas de ejecutar el código y la evaluación del código paso a paso.

Empecemos por ver cómo instalar Thonny y Python 3 en los diferentes sistemas operativos.

## 5.1 Windows

En el sistema operativo Windows vamos a utilizar la IDLE de Python. Puedes descargarla en el siguiente enlace:

<https://www.python.org/downloads/>

En el mismo encontrarás las últimas versiones de Python listas para descargarse. Descarga la versión más actual de Python 3. Una vez descargada, instala el .exe y ya podrás usar la IDLE de Python. Más adelante se explica cómo usarla.

Para instalar Thonny deberás entrar en el siguiente enlace:

<https://thonny.org/>

En el mismo encontrarás, en la parte superior izquierda, el enlace al instalador de Windows de la última versión de Thonny.

## 5.2 Mac

Para instalar Python en Mac usaremos el siguiente enlace:

<https://www.python.org/downloads/>

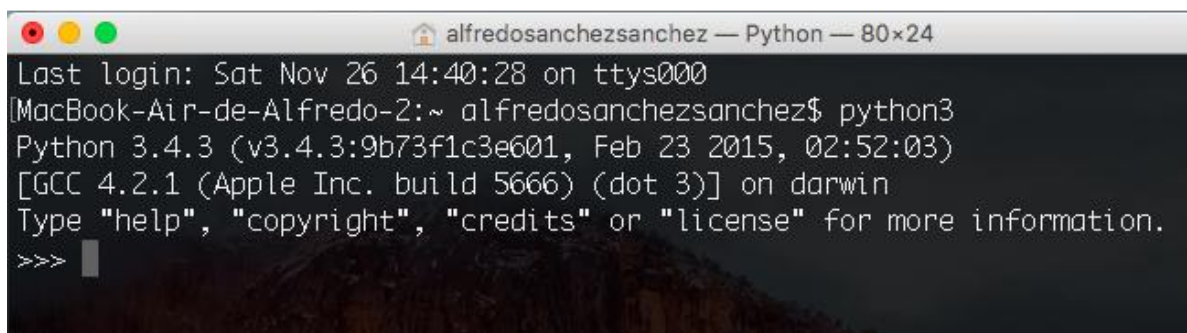
En el mismo podrás encontrar las últimas versiones disponibles de Python para descargarse. Descarga la versión más actual de Python 3:





Al pulsar en el botón de descarga (download) se descargará un archivo .pkg que tendremos que abrir. Es un paquete que contiene todo lo necesario para poder usar Python tanto desde **Terminal** como usando la **IDLE**. Al abrir el .pkg se abrirá un instalador:

Una vez instalado podremos usarlo. Para usarlo **por Terminal** sólo tendrás que abrir una Terminal (Aplicaciones → Utilidades → Terminal) y escribir Python3:



```
alfredosanchezsanchez — Python — 80x24
Last login: Sat Nov 26 14:40:28 on ttys000
[MacBook-Air-de-Alfredo-2:~ alfredosanchezsanchez$ python3
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 23 2015, 02:52:03)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

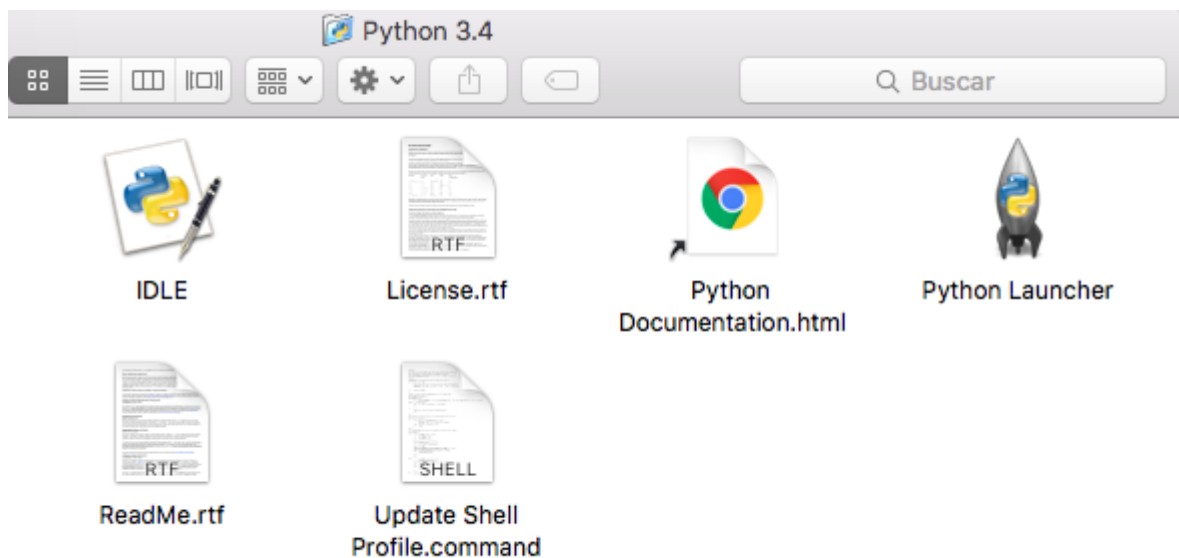
El aspecto de tu Terminal puede variar, ya que se puede configurar tanto su fondo como su letra, tamaño, etc.

Para cargar un programa de Python (terminados en .py) desde Terminal tendremos que ubicarnos en la ubicación donde está el programa y cargarlo desde ahí escribiendo *Python3 nombre\_del\_archivo.py*. Observa la imagen para entender el proceso:

```
[MacBook-Air-de-Alfredo-2:~ alfredosanchezsanchez$ cd Desktop/
[MacBook-Air-de-Alfredo-2:Desktop alfredosanchezsanchez$ python3 mediaArbitraria.py
¿De cuántos valores necesitas hacer la media?:
```

Nota: conviene fijar la Terminal en el Dock, ya que la usarás frecuentemente en este curso.

Para usar Python desde la **IDLE** sólo tendrás que buscar su ubicación una vez instalada, generalmente está dentro de una carpeta llamada Python ubicada en Utilidades.



Pinchando en el icono de la IDLE abriremos el programa:

```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 23 2015, 02:52:03)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> WARNING: The version of Tcl/Tk (8.5.9) in use may be unstable.
Visit http://www.python.org/download/mac/tcltk/ for current information.
```

Más adelante veremos cómo usar la IDLE, ya que se usa de la misma forma para los tres sistemas operativos.

Para instalar Thonny deberás entrar en el siguiente enlace:

<https://thonny.org/>

En el mismo encontrarás, en la parte superior izquierda, el enlace al instalador de Mac de la última versión de Thonny.

### 5.3 Linux

En el caso de Linux nos vamos a centrar en distribuciones GNU/Linux, siendo Ubuntu el sistema operativo más habitual.

En dicho sistema operativo Python viene instalado por defecto, si bien no tiene porqué ser Python 3 la versión instalada. Para obtener Python 3 simplemente tendremos que abrir un Terminal y escribir:

```
sudo apt-get install python3
```

Si ya está instalado Python 3 simplemente nos informará de que esa versión ya está instalada:

```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

escuela@escuela-Lenovo-ideapad-100-15IBY:~$ sudo apt-get install python3
[sudo] password for escuela:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
python3 ya está en su versión más reciente (3.5.1-3).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 270 no actualizados.
escuela@escuela-Lenovo-ideapad-100-15IBY:~$
```

Si la versión que viene por defecto es Python 3 quizá convenga que instales, para el futuro, Python 2.7, ya que si sigues avanzando en la programación con Python puedes encontrar funcionalidades que quieras implementar en Python 2:

```
sudo apt-get install python2.7
```

Por otro lado, para obtener la IDLE de Python 3 tendremos que instalarla igualmente desde Terminal:

```
sudo apt-get install idle
```

Para instalar Thonny deberás entrar en el siguiente enlace:

<https://thonny.org/>

En el mismo encontrarás, en la parte superior izquierda, las instrucciones para instalar desde terminal la última versión de Thonny.

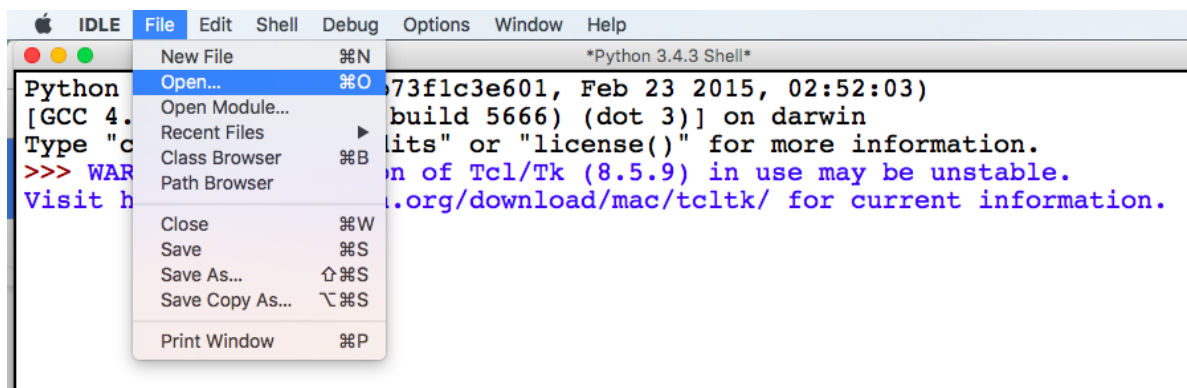
## 6. USANDO LA IDLE DE PYTHON 3

Vamos a ver rápidamente cómo usar la IDLE de Python 3. Dicha IDLE tiene directamente un intérprete de código Python. En el mismo podemos escribir instrucciones y se ejecutarán si tienen sentido para Python.

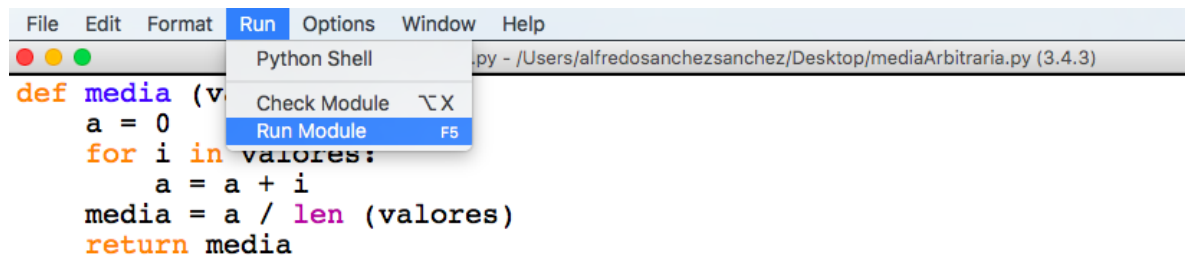
Cualquier proceso que yo explique ejecutándolo por Terminal (uso un Mac) se puede reproducir exactamente igual por Terminal en Linux o en la IDLE de Python 3 en cualquier sistema operativo.

Algunas instrucciones muy avanzadas, que requieren módulos externos, dan problemas con la IDLE, pero llegados a ese punto ya sabrás cómo ejecutar el programa a través de terminal o código del sistema de Windows.

Así mismo, con la IDLE podemos abrir programas guardados con extensión `.py`, simplemente hay que abrirlo desde la pestaña superior *File* o *Archivo*:



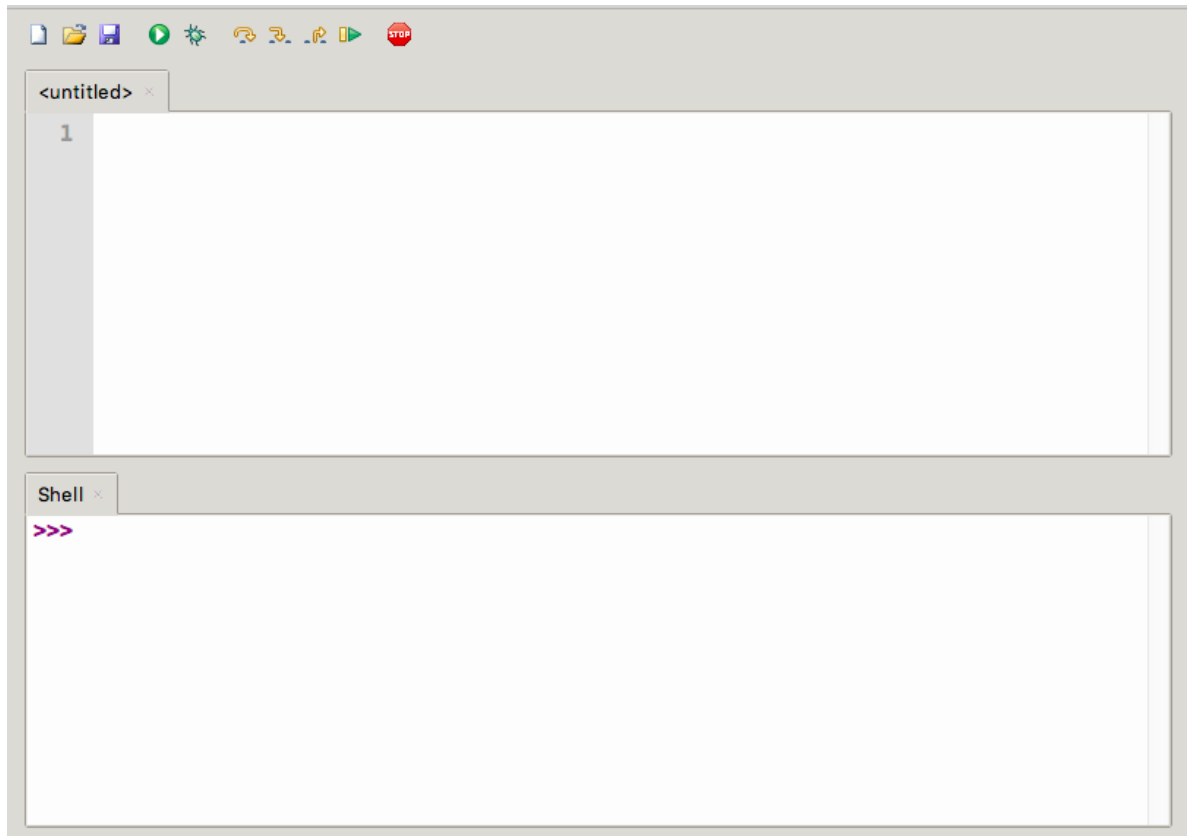
Una vez abierto, para ejecutar el programa sólo tendremos que seleccionar la opción *Run module* de la barra de herramientas *Run*. También sirve presionar F5:



El ejemplo anterior se ha realizado cargando un archivo de Python propio, no intentes reproducirlo, sólo recuerda cómo se podía ejecutar un archivo *.py*.

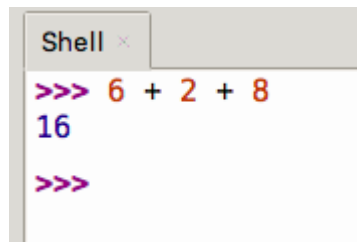
## 7. USANDO THONNY

Thonny es un entorno de desarrollo maravilloso para empezar a programar. Una vez instalado verás que su aspecto es similar a la siguiente imagen:



En la primera sección (<untitled> en la imagen) podemos escribir nuestro código y en la segunda (Shell en la imagen, término para referirse al **intérprete** que ejecutará nuestro código) aparecerá el resultado del mismo al ejecutarlo.

Así mismo, se pueden hacer pruebas de código rápidas en la sección *Shell* de forma que si queremos ver qué hace una línea de código podemos probarla ahí mismo, por ejemplo:



```
Shell x
>>> 6 + 2 + 8
16
>>>
```

A screenshot of a Python Shell window. The window has a title bar with the text "Shell" and a close button. The main area shows a Python prompt ">>>" followed by the expression "6 + 2 + 8" in red text. The result "16" is displayed in blue text on the next line. A second prompt ">>>" is shown on the following line.

Algunos programadores usan directamente un intérprete de Python como calculadora. Huelga decir que en esos casos no se usará Thonny, sino que se usará Terminal.



## 8. EDITOR DE CÓDIGO

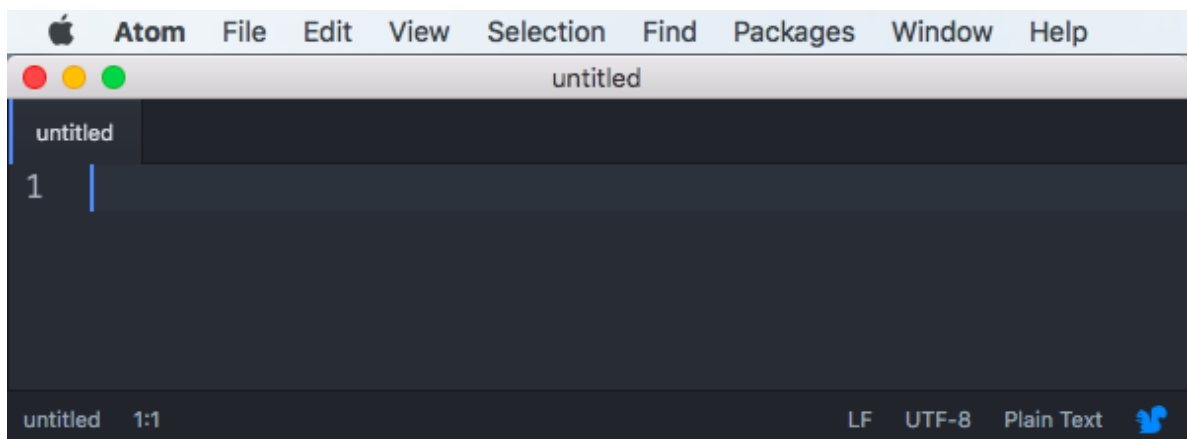
Un editor de código es un editor de texto que reconoce, por la extensión del archivo que carguemos en el mismo, el lenguaje en el que está escrito el código y lo muestra editado resaltando las diferentes partes y favoreciendo tanto la lectura como la escritura rápida de código.

A la hora de escribir programas de Python es muy recomendable usar un editor de código y guardar los archivos, abriéndolos después desde la IDLE, Thonny o Terminal.

Para este curso voy a recomendar un editor de código: **Atom**. En el mismo podemos escribir en muchos lenguajes de programación e incluso archivos de texto al uso. Descárgalo para tu sistema operativo desde este enlace:

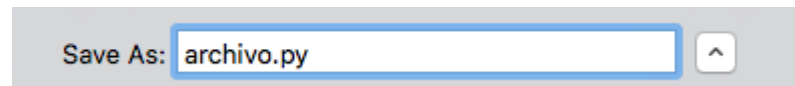
<https://atom.io/>

Una vez lo descargues e instales ábrelo para familiarizarte con él:

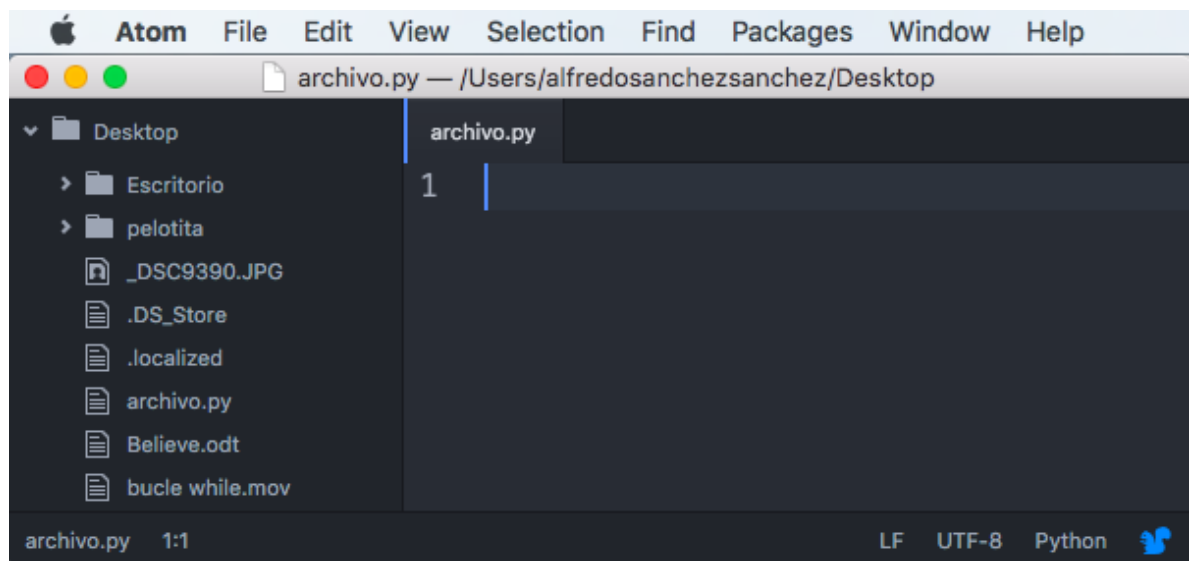


La imagen corresponde a Atom en un Mac, pero se muestra similar en Linux y Windows.

Atom no reconocerá el tipo de código que estamos usando hasta que no guardemos el documento y declaremos su extensión. Al guardarlo hay que indicar **qué tipo de lenguaje contiene** con una extensión. Los archivos de Python terminan en **.py**, por lo que a la hora de guardar un archivo deberemos indicar un nombre y añadir la extensión mencionada:



También podemos elegir la ubicación del archivo a la hora de guardarlo. Una vez lo elijamos, Atom automáticamente mostrará en una barra lateral izquierda todos los archivos que contiene la carpeta donde has guardado el mencionado archivo:



Según avance el curso seguramente comencemos a escribir los programas en Atom los guardemos para posteriormente ejecutarlos en la IDLE, Thonny o Terminal. El uso de Atom facilita la legibilidad de tu código y el autocompletado de instrucciones de Python de forma que evita tener que escribir ciertas partes de la sintaxis (por ejemplo, cuando abrimos paréntesis ya incluye el paréntesis cerrado final y el cursor se sitúa entre ambos para escribir).

No vas a poder ejecutar a través de Atom los programas que escribas, de la misma forma que Word no suele ser el instrumento de muestra final de un escrito, pero ambos facilitan el proceso de obtener algo concluido, en un caso un programa en el otro un documento de texto.

## 9. ¡A PROGRAMAR!

Con esto tienes lo necesario para empezar el curso y poder trabajar. Espero y deseo que a lo largo del curso afrontes con ilusión los retos que el propio aprendizaje te va a deparar.

No tengas miedo en preguntar lo que haga falta pero intenta antes solucionarlo por tu propia iniciativa, es mucho más gratificante. Además, tómate en serio probar cada cosa que se indique antes de seguir, sólo probando, pensando, observando y practicando aprenderás a programar. Yo (el autor) ya sé programar y ver el resultado de mis programaciones no va a aportarte nada, pues salvo en algunos casos que busco el error, mis ejemplos son directos y no tienen fallos. La única manera de aprender es equivocarse, fallar, analizar el error y aprender de ello.

Nos vemos en el tema 2.