

## Visual Studio Code

Existen varios **IDEs (Integrated Development Environment)** de programación en Python. Para aprender a programar normalmente se utilizan los Notebooks de Jupyter, donde podemos integrar celdas de código Python, junto con celdas de markdown, de manera que podamos documentar y explicar mejor cada línea de código. Ahora bien, si lo que queremos es desarrollar un programa de Python en un **entorno de desarrollo**, un programa con bastantes más líneas de código de lo que solemos hacer en un Notebook, tendremos que desarrollar en entornos tipo **Visual Studio Code**.

Visual Studio Code es un IDE de Microsoft que nos sirve para crear nuestros primeros programas de Python.

### ¿Qué diferencia hay respecto al Notebook?

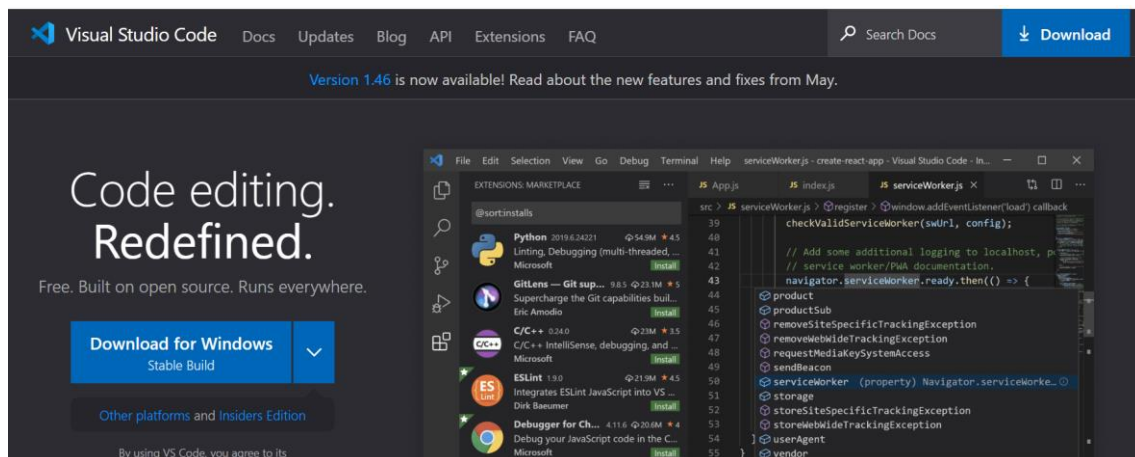
- Scripts vs celdas
- Debugging
- Mayor integración con git y conexiones a servidores.
- Intellisense

### - Instalación

Podréis instalarlo desde esta web:

<https://code.visualstudio.com>

Si te fijas en el desplegable, tienes la versión para cada sistema operativo.



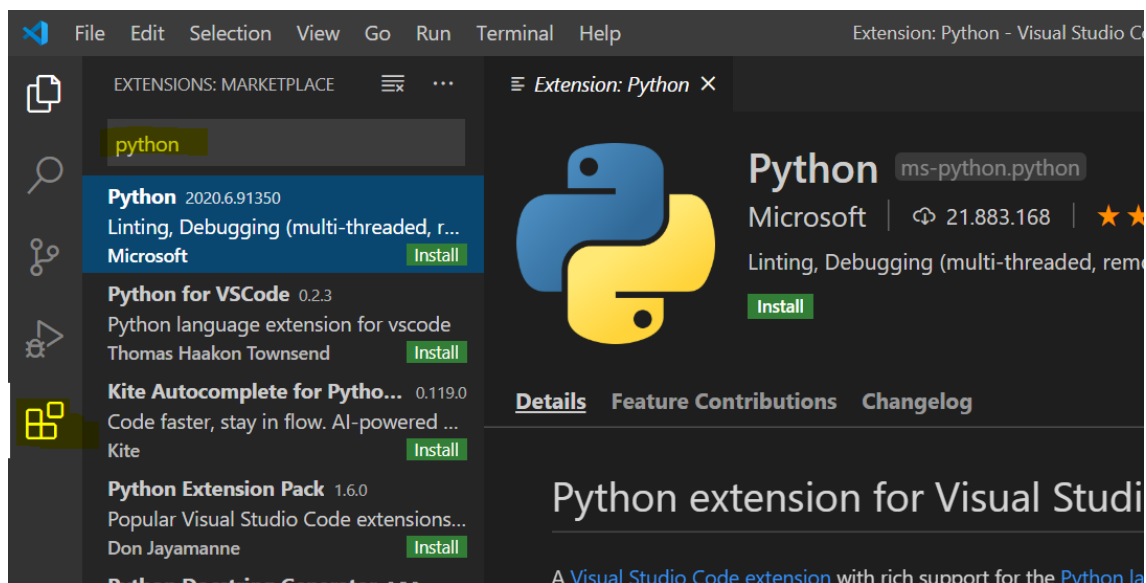
Una vez descargas el archivo .exe, ejecútalo y sigue paso a paso la instalación con la configuración que viene **por defecto**.

### - Intérprete de Python

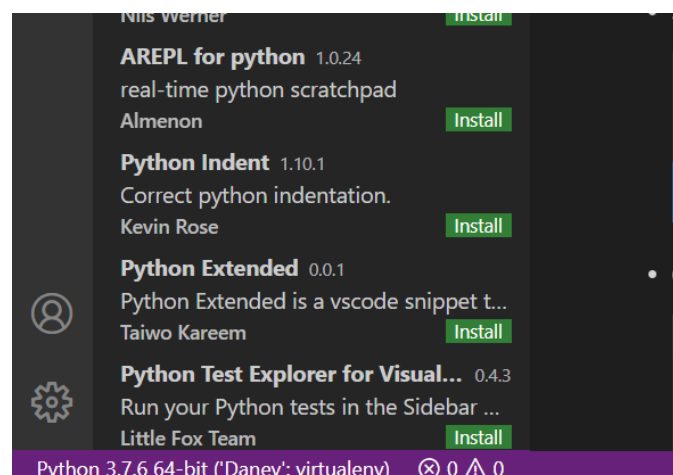
**Visual Studio Code no es más que un editor de texto** que se utiliza para programar en diferentes lenguajes, aunque viene prácticamente vacío. Si llevase todo lo necesario para

programar en múltiples lenguajes, sería un editor muy pesado de instalar y manejar. Probablemente el desarrollador que utiliza Java, no usa R o Python, por lo que VSCode viene vacío y es el propio desarrollador el que configura el entorno mediante **las extensiones**.

Por tanto, ve a las extensiones (imagen de abajo) e introduce *python* en el buscador. Selecciona la primera, cuyo dueño es Microsoft y tiene unas 21 MM de descargas a día de hoy (02/07/2020).



No tarda mucho en instalarse. Para comprobar que ya lo tienes, debería aparecer abajo a la izquierda un indicador de que no hay errores ni warnings, y que ha encontrado el **intérprete de Python**.

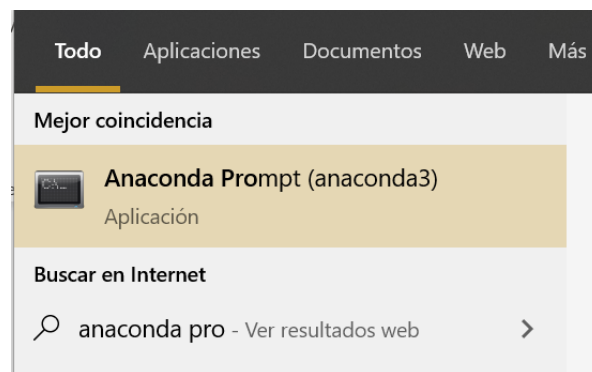


Siempre vamos a trabajar con **entornos de desarrollo como Jupyter Lab, o VSCode**, que no son más que editores de texto, que entienden de sintaxis, saben que un *for* es un bucle, y lo señalan con otro color y demás, pero no entienden de la ejecución del programa. Para ello está **el intérprete de Python**, en nuestro caso, es la versión 3.7. El intérprete o bien lo descargamos

desde [la web de Python](#), o se nos descarga automáticamente con plataformas como **Anaconda, donde viene integrado en la instalación.**

En el caso de VSCode, viene totalmente vacío, **tenemos que instalar la extensión de Python** (lo que acabamos de hacer), **y por otro lado habrá que linkarlo al intérprete de Python.** Si no lo tienes instalado, acude a la web de Python, pero si ya se instaló con Anaconda, no deberías tener problema. Ahora bien, VSCode automáticamente hace un mapeo en tu ordenador para buscar el intérprete. En el caso de la imagen de arriba, vemos que ha encontrado un intérprete (el de la instalación de Anaconda), pero podría ocurrir lo contrario, que no lo encuentre o no exista ninguno en el ordenador.

¿Cómo solventamos esto? Buscamos el intérprete de Python de nuestro ordenador, y le decimos a VSCode donde está. Para ello, abre **un Prompt de Anaconda:**



Ejecuta un:

> where Python

```
(base) C:\Users\Daney>where python
C:\Users\Daney\anaconda3\python.exe

(base) C:\Users\Daney>_
```

Perfecto, mi intérprete está en “C:\Users\Daney\anaconda3\python.exe”. Volvemos al VSCode hacemos clic abajo a la izquierda, e introducimos la ruta donde se encuentra el intérprete. Siempre va a ser un archivo “python.exe”.

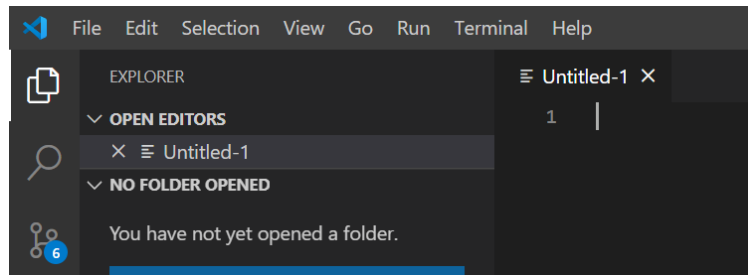
Ya tenemos el entorno listo y configurado para empezar a trabajar 😊

## - Primer script en Python

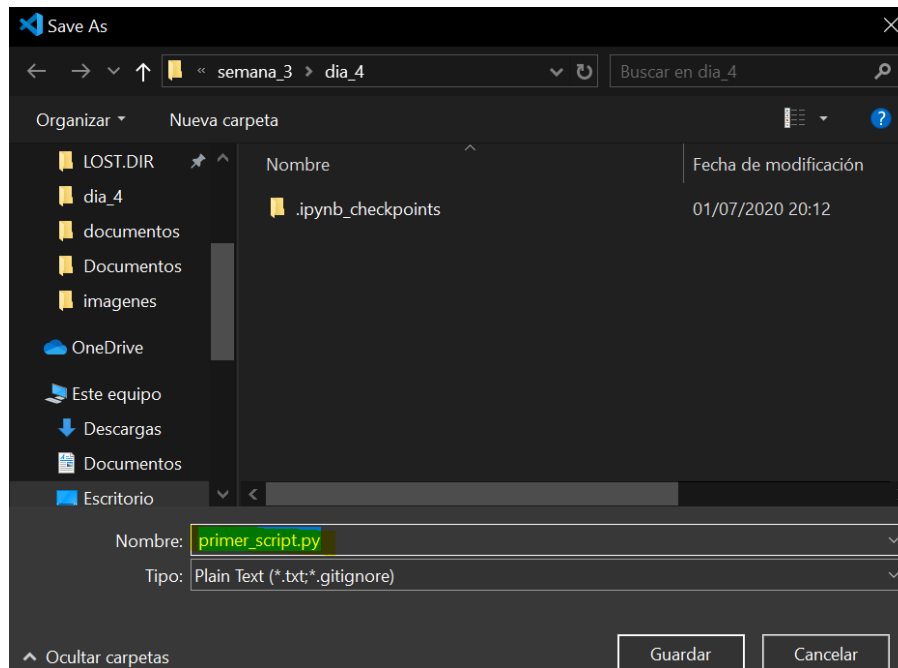
Ya tenemos todo configurado y podemos empezar a programar.

### Crea tu primer archivo

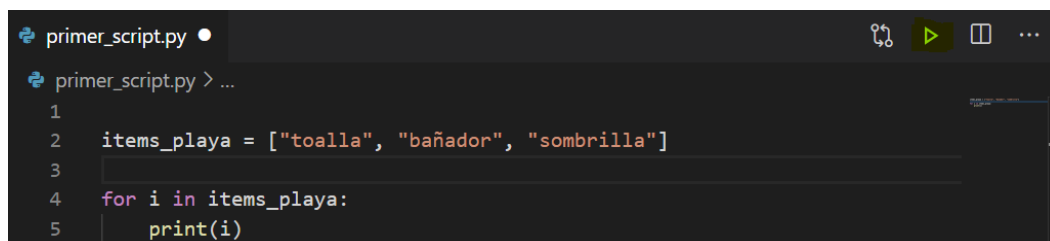
File -> New File. Por defecto creará un archivo llamado “Untitled.txt”.



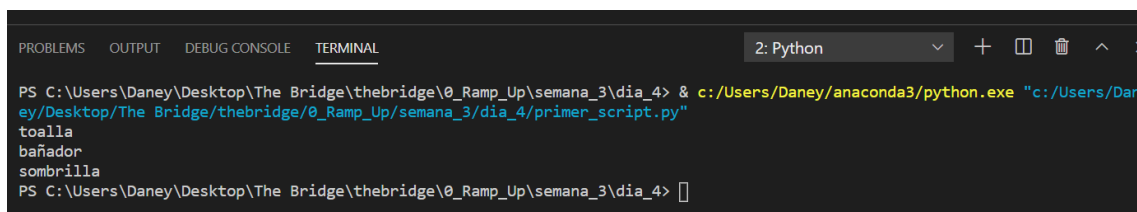
Si queremos que lo interprete como un archivo de Python, tendremos que guardarlo con la extensión de archivo de Python (.py):



Fíjate que ahora, además de cambiar el nombre del archivo, aparece al lado el símbolo de Python. Vamos a crear unas líneas simples de código. ¿Cómo se ejecuta? El botón de *Play* de arriba.



Verás que aparece una ventana abajo que tiene esta pinta:



Vemos que ha ejecutado, y no ha habido ningún error. Además podemos ver los outputs del programa, que en este caso, es simplemente un *print* dentro de un *for*.

Fíjate en la sentencia que ha escrito. Una ruta en amarillo con el intérprete de Python, y otra línea en azul con tu script de Python. Así es como se ejecutan las cosas de una manera productiva. Se le indica donde tiene el intérprete, y donde está el código. Fin. VSCode nos hace la vida más fácil con su interfaz de usuario amigable, pero en el fondo lo que hace es esa sentencia, prueba a abrir un terminal de tu sistema operativo y a ejecutar esa sentencia, a ver qué sale.

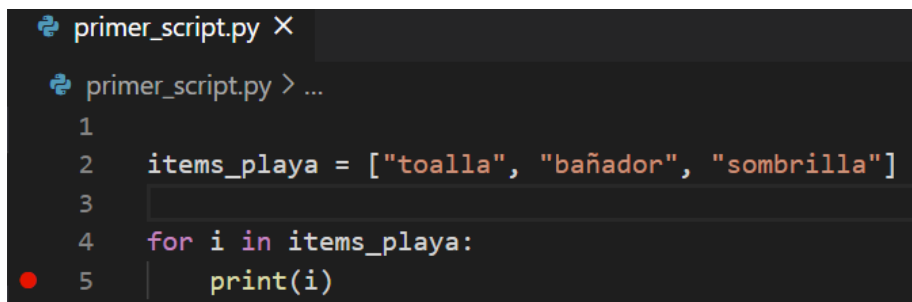
Dándole al *play* es la manera que tenemos de ejecutar el script entero, pero ¿y si lo que queremos es ejecutar unas líneas como hacíamos con los Notebooks? También tenemos la opción. Seleccionamos las líneas a ejecutar y después shift + enter.

## - Debugging

Cuando ejecutamos código y no sabemos muy bien qué está haciendo, y no solo eso, sino que puede tener errores y a simple vista no los vemos, existe una funcionalidad llamada *debugging* o depuración, que **permite ejecutar el código paso a paso, viendo en todo momento el estado de las variables en ejecución.**

Para ello, tendremos que poner *breakpoints* en el código. Un breakpoint no es más que un punto del código señalado por nosotros que indica que se tiene que parar la ejecución justo en ese lugar. Si tenemos la sospecha de que un bucle no funciona bien, y no sabemos por qué, podremos colocar un *breakpoint* dentro del bucle para ver paso a paso su ejecución.

Los *breakpoints* se colocan haciendo click en el margen izquierdo del script:

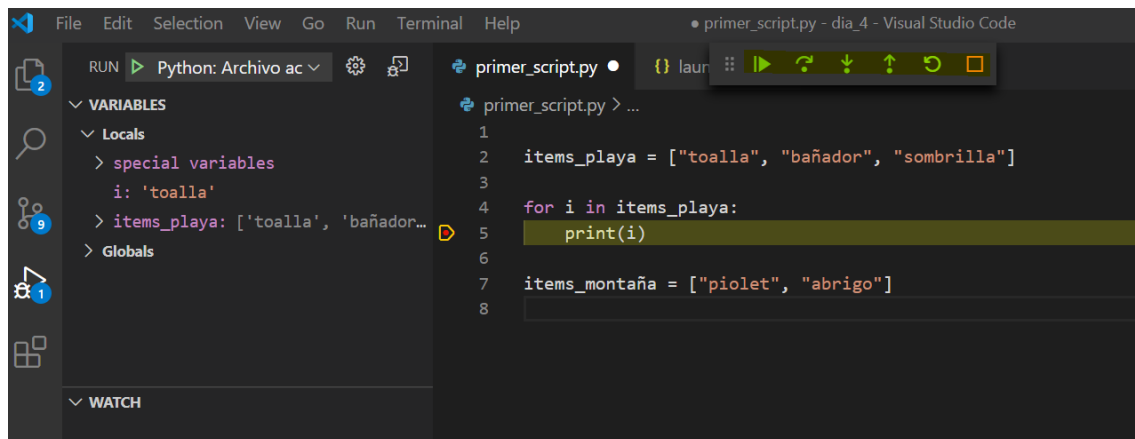


```
primer_script.py X
primer_script.py > ...
1
2 items_playa = ["toalla", "bañador", "sombrilla"]
3
4 for i in items_playa:
5     print(i)
```

## Run vs Debug

Cuando le damos a correr, simplemente se ejecuta el código, con sus inputs, outputs, bucles y demás. Pero cuando le damos a *debug*, también corre el script, pero se va a parar en los breakpoints, lo que nos va a dar mucho juego a la hora de ir hacia adelante y hacia atrás en la ejecución.

Añade unos ítems más en otra lista después del for y depura tu código. Te va a aparecer algo parecido a esto:



### ¿Qué vemos en el debugging?

- A la izquierda vemos **las variables** declaradas. De momento “ítems\_playa”, y la variable de ejecución del bucle “i”, que es equivalente al primer ítem.
- En el script, tienes un **indicador en el breakpoint** y un menú arriba. Con ese menú podrás ir moviéndote en la ejecución, yendo al siguiente paso, el paso anterior, el siguiente *breakpoint*, o parando el debugging.