

Objetivos del Tutorial:

1. Aprende a instalar Python y comienza a programar
2. Aprende los conceptos básicos de Python
3. Explora a dónde puedes ir a continuación en tu viaje de Python

Este tutorial es un recorrido rápido por Python, el objetivo no es que te conviertas en un experto al final de este, sino más bien para sentar las bases para que empieces a abordar tus propios desafíos en Python y para comenzar tu aprendizaje. Después de este tutorial, espero que entiendas los conceptos básicos de python, sepas cómo y dónde puede escribir código, saber dónde puedes obtener más ayuda cuando estás atascado y sentir entusiasmo por la idea de aprendiendo más.

Pasos:

- 1 Por qué aprender Python
 - 2 Cómo instalar Python con Anaconda
 - 3 Escribiendo código Python en Spyder
 - 4 Variables e impresión
 - 5 Matemáticas simples y operadores
 - 6 Bucles
 - 7 Funciones
 - 8 Declaraciones if-else
 - 9 Importaciones (paquetes)
 - 10 Encontrar ayuda en línea
 - 11 Próximos pasos
-

¿Por qué Python?

Python es uno de los lenguajes de programación más utilizados y de los que más se habla que existen. Fue diseñado ser fácil de leer y escribir desde el momento en que fue diseñado. Por esta razón, es un gran lenguaje para aprender si eres nuevo en la programación. Sin embargo, Python tiene muchos otros beneficios que incluyen:

- Es gratuito y de código abierto, nunca tendrás que pagar para usar Python.
- Es ampliamente utilizado y tiene una gran base de usuarios, por lo tanto, siempre podrás encontrar ayuda en línea
- Es el lenguaje más utilizado para el aprendizaje automático y la ciencia de datos
- Es multipropósito y puedes hacer cualquier cosa, desde crear gráficos bonitos hasta entrenar redes neuronales. redes, ejecutar simulaciones, raspado web, alojamiento web, crear juegos y mucho más más

- Python tiene muchas bibliotecas geniales y gratuitas (complementos) que puedes usar sin tener que escribir todo tu código desde cero
- Python es una gran herramienta para estadísticas, matemáticas, procesamiento de datos y modelado.

Esperamos que estés convencido de que Python es una gran herramienta para añadir a tu repertorio.

2. Instalando Python con Anaconda

Así que ahora queremos saber cómo instalar Python y empezar a escribir algo de código Python impresionante. gobernar el mundo.

Hay una distribución llamada Anaconda que es muy popular entre los usuarios de Python y es muy poderosa. herramienta, vamos a instalar esto. Como mencioné, Anaconda es una 'distribución' de Python, esto básicamente significa es un paquete de Python que viene con el lenguaje Python y algunos extras. Cuando instalas Anaconda también instalarás Spyder (una aplicación que puedes usar para escribir código Python) y algunas otras cosas de las que no necesitas preocuparte aún.

Para este tutorial, estaremos trabajando con Spyder dentro de la distribución Anaconda, esto sin duda será suficiente para comenzar y llevarte muy lejos en el universo de Python. Spyder es un IDE integrado entorno de desarrollo (si eres un aficionado de R, puedes pensar en Spyder como RStudio para Python), es básicamente el programa que usarás para escribir todo tu código. Si eres completamente nuevo en la codificación puedes pensar sobre Spyder como si fuera Microsoft Word para escribir código Python. En Microsoft Word, creas un nuevo documento y escribes tus ideas en inglés (o en cualquier otro idioma maravilloso), si cometes un error tipográfico entonces Microsoft Office te pedirá que corrijas tu error. En Spyder creas un nuevo 'script' y escribes bajo tus ideas en Python, si cometes un error tipográfico, Spyder te pedirá que lo corrijas.

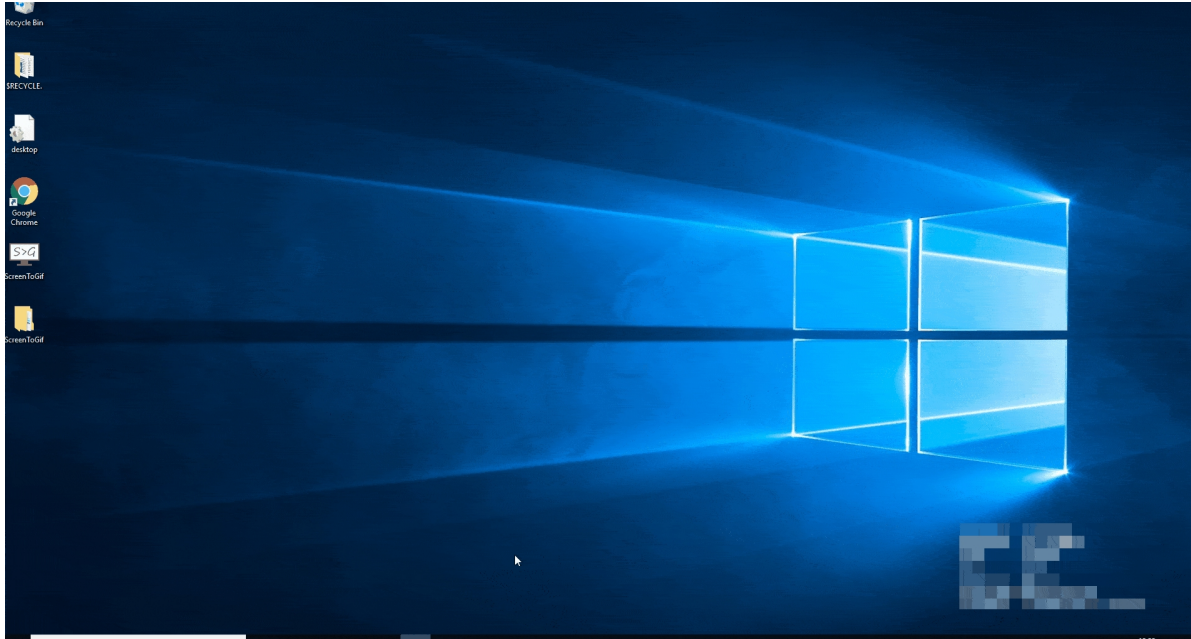
Aquí hay un gran video que te muestra cómo instalar Anaconda si aún no lo tienes.

¡Felicidades! ¡Acabas de instalar Python! Asegurémonos de que todo esté funcionando ahora.

Abre un símbolo del sistema (terminal) y escribe `conda --versión` Si todo ha sido instalado con éxito, luego de una pausa de momentos, el número de versión de tu conda debería imprimirse en tu pantalla.

3. Escribiendo Python en Spyder

Ahora que tenemos Python instalado, es hora de lanzar Spyder. Puedes hacer esto iniciando Anaconda. Navegador desde el menú de inicio. Cuando esto se inicie, haz clic en la aplicación Spyder para abrirla.



Una vez que hayas lanzado Spyder, verás tres paneles principales por defecto. Puedes editar el diseño de estos paneles si quieres, pero para este tutorial asumiré que no quieres meterte con eso aún. En el izquierda, el panel grande es donde puedes escribir guiones. En la parte superior de este panel puedes ver que hay se ha iniciado un guion para ti llamado temp.py. Si quisieras, podrías escribir código en este script (similar para escribir un documento de MS Word) y luego guardarlo para ejecutarlo más tarde.

Un guion en python es básicamente lo mismo que un guion en literatura. Un guion para una obra de teatro es escrito por el autor. para decir a los actores qué deben hacer y decir y en qué orden. Un guion para un programa de python está escrito por el autor para decirle a la computadora qué debe hacer y en qué orden.

Así que cuando escribes comandos en el script de Python a la izquierda, es como escribirlos en el regular guion. Tú almacenas tus líneas de diálogo (código) para ser ejecutadas (corridas) más tarde.

En la parte inferior derecha de la ventana verás la consola, y en eso es en lo que nos vamos a centrar. esta parte del tutorial. Esto es diferente al panel de scripts. En la consola, cada vez que escribes un **comando y luego presiona enter, ese comando se ejecutará de inmediato, no se almacenará para más tarde como en el script.** Puedes pensar en usar la consola de manera similar a usar una calculadora. Siempre que escribas algo en ello se ejecuta de inmediato. Puedes ver esto en la demostración a continuación. Primero le decimos a la consola qué el `variablex` es; luego le decimos que imprimax para que podamos verlo de nuevo. Después de esto le decimos que imprima otros productos `dex` Intenta escribir estas mismas líneas de código en tu propio editor y ve qué sucede.

```

Python 3.6.1 |Anaconda 4.4.0 (64-bit)| (default, May 11 2017, 13:25:24) [MSC v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license()" for more information.

IPython 5.3.0 -- An enhanced Interactive Python.
?      -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help    -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.

Restarting kernel...

In [1]:

```

Nota que mientras escribía líneas de código en la consola cometí un error y la consola me mostró un mensaje de error. En este caso, todo lo que necesitaba hacer era corregir el error y ejecutar la línea correcta.

4. Variables e Impresión

En Python, una de las cosas básicas que podemos hacer es establecer variables. Una variable es solo un almacén o un atajo para referirse a otra cosa. En el ejemplo anterior, asigné el número 5 a la variable `x`, `x=5`. Después de que yo asigné la variable `x` (una vez le dije a la consola que `x` era 5) entonces podría imprimir `x` y la consola sabía que debería imprimir el número 5. En Python no necesitamos ceñirnos a letras para los nombres de variables, al igual que en tus viejas clases de álgebra. Si queremos, podríamos decirle a la consola que `x=5`, `my_variable=0`, `yesterday=999`. Después de haber hecho esto y pedimos a la consola que imprima `(ayer + x)` ello debería decirnos que esto es simplemente 1004. ¿Por qué no intentas hacer esto?

Consejos rápidos: Python se preocupa mucho por la capitalización y los espacios.

- Cuando nombras una variable no puedes tener espacios en el nombre de la variable. Así que en lugar de usar espacios, las personas a menudo usan guiones bajos en su lugar. Esta es la razón por la que, en el ejemplo arriba, utilizamos `my_variable` en lugar de `mi variable`. Intentando asignar un nombre de variable con un espacio causará un error.
- Python no lee como lo hacemos nosotros, es 'sensible a mayúsculas y minúsculas'. Esto significa que por más que intente, nunca sabrá que `mi_variable` y `mi_variable` son el mismo. Puedes usar mayúsculas en tus variables, pero solo necesitas ser consistente.

También podemos asignar otras cosas a variables. En Python, los tipos base son enteros, punto flotante, números, cadenas, booleanos, listas y diccionarios. ¡No te desanimes! Pasaremos por lo que cada uno de estos

significa a su vez y ver por qué pueden ser útiles, todos son muy útiles. Antes de hacer eso, quiero te presento dos funciones en Python. A saber, `print` y `type`.

El `print` función

Ya hemos visto esto antes cuando imprimimos `x`. La forma de usar esta función es imprimir la cosa que debe ser `print`ed)Puedes imprimir cualquier información válida en Python. Intenta imprimir algunos números.

El `type` función

Usas esta función para averiguar el tipo de cualquier variable en Python. Por ejemplo, intenta ingresar lo siguiente en tu consola.

```
x=5
print(type(x))
```

[Copiar contenidos](#)

Según lo que está impreso aquí, ¿puedes decir qué tipo de variable es `x`? Inténtalo de nuevo con `x=5.0` y ver si obtienes una respuesta diferente.

Números enteros

Los enteros son solo números enteros, probablemente aprendiste a contarlos cuando eras niño. Podemos establecer variables iguales a enteros haciendo lo siguiente.

```
x=5
y=89
year=2018
```

[Copiar contenidos](#)

Inventa algunos nombres de variables, como `x`, y `oABBAy` dales valores enteros. Luego imprime el tipo de estas variables para asegurarse de que realmente son enteros.

Números de punto flotante

Los números de punto flotante son solo números que tienen un punto..., es decir, se expresan como un decimal.(Rápido ten en cuenta que python fue desarrollado en inglés americano y por lo tanto los decimales usan un punto en lugar de una coma por ejemplo

Intenta inventar algunos nombres de variables y darles números de punto flotante. `flotar`) valores.

For example you might say `glass_fullness = 0.5`, o los más pesimistas de ustedes podrían decir `glass_emptyness = 0.5`. Luego imprime el tipo de estas variables para asegurarte de que sean flotantes.

Cadenas

En Python, las cadenas se utilizan a menudo para almacenar variables de texto. Por ejemplo, podría querer saber `esocoding_club_rating = 'Great'` o `esobest_song = "Dancing Queen"`. Tenga en cuenta que en python it no importa si utilizamos comillas simples o comillas dobles al alrededor de algunos caracteres para hacer esto en una cadena. Cualquier cosa que esté entre comillas simples o dobles es una cadena. Otro más el uso práctico podría ser almacenar la fecha como una cadena `date = '17-10-2018'` o para almacenar el nombre de un especie que estamos examinando `species = "killer whale"`. Intenta inventar algunos nombres de variables y dar los valores de cadena. Luego imprime sus tipos.

Listas

En Python, cualquier cosa que esté rodeada por corchetes cuadrados `[]` y separado por comas es una lista. Puedes insertar cualquier cosa en una lista que incluya otras listas, y las entradas no tienen que ser del mismo tipo. Podríamos hacer un lista de números asignando `these_numbers = [2, 78, 1, 0, 12]`, o fechas por `these_dates = ['12-08-1987', '02-05-1852', '25-12-1999']`, o de la edad de los árboles por `tree_ages = [1, 4.5, 1000, 19.7, 'older than time itself']`

¿Es importante que tus listas tengan comas entre cada elemento y que comiencen y terminen con corchetes? paréntesis.

Booleanos

Una variable booleana es una variable que es o `True` o `False`. Tener variables como esta es útil en Python ya que nos brinda mucha flexibilidad. Por ejemplo, podríamos querer escribir algún código que verifique si es viernes y, si es así, nos dice que regresemos a casa temprano. En este código podríamos almacenar la variable `is_it_friday = Verdadero` y luego usa esto más tarde.

Diccionarios

(nota: los diccionarios son un poco más difíciles de entender que el resto del material aquí, así que siéntete libre de saltarte esta parte) y vuelve cuando estés listo) Los diccionarios son un tipo de dato más especializado en Python. Son un un poco como una lista, pero cada elemento en la lista (cada valor) tiene un nombre (una clave). En diccionarios normales you will find a list of definitions, where each definition has a key (ie the word you look up).

Por ejemplo, si buscas python (esta sería la clave) en el Diccionario Inglés de Oxford, podrías encontrar thevalue:

una gran serpiente de cuerpo robusto y no venenoso que se encuentra en toda la región tropical del Viejo Mundo, matando a sus presas por constricción y asfixia.

En los diccionarios de Python, tus claves deben ser cadenas y tus valores pueden ser de cualquier tipo. Los valores podrían ser enteros, flotantes, cadenas, listas, más diccionarios o cualquier otra cosa.

Sabrás que algo es un diccionario porque tendrá llaves alrededor. Aquí hay un example of a dictionary used to store movie quotes

```
my_dictionary={"jurassic park":"life finds a way",
               "terminator": ["i'll", "be", "back"] }
```

Copiar contenidos

El diccionario de arriba tiene dos claves,parque jurásicoyterminadorLos diccionarios son siempre de la formulario {key:value, key:value, key:value, ...} Entre cada clave y su valor ponemos un : y between each key:value pair we put a . Enmi_diccionariolos valores son de diferentes tipos. El valor for the key ‘jurassic park’ is a string; whereas the value for the key ‘terminator’ is a list of strings.

Podemos acceder a los valores en un diccionario buscándolos con su clave. En el ejemplo anterior podríamos imprimir la cita de Jurassic Park utilizandoimprimir(mi_diccionario['parque jurásico'])Podríamos imprimir el otra cita al poner la otra clave en los corchetes de la declaración de impresión. En general, accedes a los elementos en un diccionario colocando el nombre de la clave entre corchetes después de la variable de tu diccionario nombre. Así que si tu diccionario se llamaDrDiccionarioy quieres el valor asociado con la clave ‘coolKey’ que usaríasDrDictionary['coolKey'].

- Now make your own dictionary of your own favourite movie quotes. Try storing the movie citas como cadenas, listas, enteros o flotantes y luego imprímelas.

5. Matemáticas simples y operadores

Todos los operadores matemáticos básicos que conoces y amas, como+ y- están disponibles en Python. En esta sección vamos a mostrar y demostrar algunos de estos operadores.

- Adición y sustracción (números y cadenas)
- Otros operadores básicos
- Operadores de comparación
- Operadores booleanos (lógica Verdadero/Falso)

5a. Suma y resta (números y cadenas)

Python se puede utilizar esencialmente como una gran calculadora que puede hacer todos los cálculos que no desees hacer. Puede multiplicar, dividir, sumar, restar, etc., etc.

Lo primero que necesitas saber es que los operadores básicos tales como `+` y `-` actuar de manera ligeramente diferente dependiendo de los tipos de datos que estés utilizando. Por ejemplo, agregar dos enteros te dará qué tú esperas, escribiendo `56 + 19` en la consola daría `75`; pero agregar dos cadenas las 'concatenará' (pégalos juntos) `Hola Mundo` daría `HolaMundo`.

Try adding some integers together and printing the result; then try the same with some strings and some flotantes. Ej.:

- ¿qué es? `invierno` que se acerca?
- ¿qué es? `34 + 99`?
- ¿qué es? `5.14 + 17.87`?
- ¿qué es? `Verdadero + Verdadero`?

PREGUNTA DE DEPURACIÓN: ¿qué es? `4 + 2` y qué es? `'4' + '2'`? ¿Y por qué son diferentes las respuestas?

A continuación, deberías intentar restar algunos enteros y flotantes entre sí y mostrar los resultados. ¿Lo haces? ¿Crees que puedes restar cadenas entre sí?

5b. Otros operadores básicos

Ahora que tienes la idea de cómo funcionan los operadores, ¿por qué no te diviertes jugando con estos nuevos brillantes? unos

- `/` : dividir : ej `20/4` (answer : 5)
- `*` multiplicar : ej `14*3` (answer : 42)
- `%` módulo: ej `12 % 5` (respuesta : 2)
- `**` elevar a la potencia : ej `2**4` (answer : 16)

Intenta usar cada uno de estos operadores en enteros, flotantes y cadenas. No todos funcionarán en cada tipo de dato. escribe y recibirás algunos errores, ¡pero está bien! Para citar el famoso mantra de Silicon Valley las empresas 'mueven rápido, rompen cosas' (no es un buen lema si eres un comerciante de antigüedades, pero es genial para aprendiendo Python

5c. Operadores de comparación

En Python también podemos hacer comparaciones entre variables, entre números o entre cadenas. Estos operadores puedes verificar cuando un número es mayor que otro número (para verificar tu conjunto de datos para errores, como si la edad es mayor que cero `edad > 0`) o cuando una cadena es igual a otra cadena (para filtrar tu conjunto de datos, como a un nombre de especie determinado `species_name == 'lama glama'`). Esto se puede hacer usando los operadores de comparación:

- `>`: mayor que : ej `5 > 10` respuesta : Falso
- `<`: menos que : ej `1 < 0` (answer : True)
- `>=`: mayor que o igual a : ej `17 >= 17` (answer : True)
- `<=`: menor o igual que: ej `9 <= 20` (answer : False)
- `==`: equal to : eg `1 == 0` (answer : False), `'lama glama' == 'lama glama'` (answer : True)

Intenta hacer algunas comparaciones para asegurarte de que entiendes cómo funciona cada uno de estos. Aquí hay una lista de cosas para ayudarte a comenzar, pero no tienes que detenerte ahí

5d. Operadores booleanos (lógica Verdadero/Falso)

También hay un conjunto de operadores booleanos en Python. Nos permiten realizar operaciones en variables booleanas. Esto puede realizar ciertas tareas lógicas basadas en los resultados de otros cálculos.

Toma estas líneas de código como ejemplo, donde hacemos que el usuario escriba su nombre y luego comente sobre es:

- `&` y lógico Verdadero y Verdadero (answer : True), `True & False` (answer : False)
- `|` o lógico `True | False` (answer : True), `Falso | Falso` (answer : False)
- `!` no lógico `! Verdadero` (respuesta: Falso) `! Falso` (answer : True)

¿Puedes adivinar, o hacer que Python calcule, qué Verdadero | Verdadero ¿será? ¿Qué tal con Falso y Falso?

Nota: Una de las cosas realmente agradables de Python es cuánto puede leerse como inglés simple por usando palabras clave de Python. En los ejemplos anteriores, podrías reemplazar `&` con `and`, reemplazar `|` con `or`, y reemplazar `!` con `not` y aún obtener las mismas respuestas. ¿Por qué no intentar reescribir los ejemplos anteriores usando `and`, `or` y `not` palabras clave en lugar de los símbolos. Te empezaré.

```

Console 1/A
Python 3.7.0 (default, Jun 28 2018, 08:04:48) [MSC v.1912 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 6.5.0 -- An enhanced Interactive Python.
C:\Users\s1205782\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel\parentpoller.py:116: UserWarning: Parent poll
failed. If the frontend dies,
    the kernel may be left running. Please let us know
    about your system (bitness, Python, etc.) at
    ipython-dev@scipy.org
    ipython-dev@scipy.org"""
In [1]:

```

IPython console History log

Permissions: **RW** End-of-lines: **CRLF** Encoding: **UTF-8** Line: **8** Column: **1** Memory: **20 %**

Usando lo que sabe hasta ahora, ¿puede resolver el dilema de Hamlet a continuación? ¿Cuál será la respuesta sito_be = Verdadero o conto_be = False

```

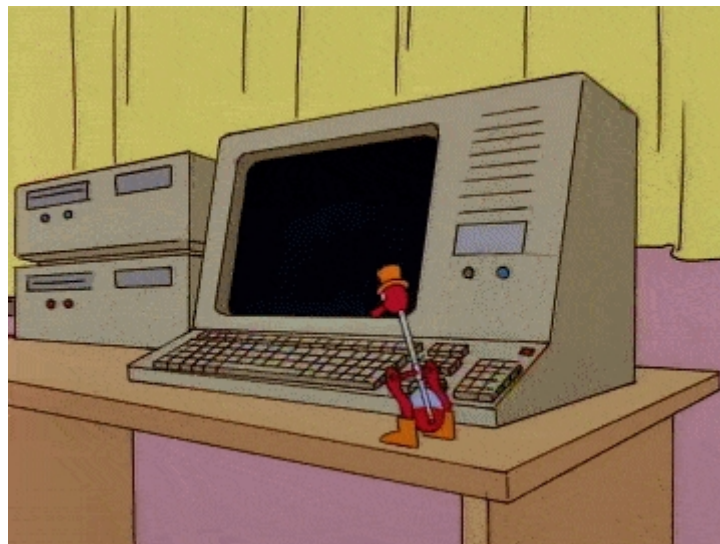
to_be=True
imprimir(ser o no ser)

```

Copiar contenidos

6. Bucles

En bucles escribimos instrucciones que se repetirán un número de veces seguidas. Podemos usar bucles para repetir acciones de manera más eficiente.



Digamos que tienes un trabajo más importante que hacer que presionar 'y', digamos por ejemplo que nos gustaría calcular la media de algunos datos.

```
precip=[2,7,1,9,0,2,4,5]
```

[Copiar contenido](#)

Si fueras a hacer esto manualmente con lápiz y papel, podrías decidir sumarlos en secuencia. Así que tú tomaría los dos primeros números y los sumaría ($2+7=9$) luego sumaría el siguiente número al total actual ($9+1=10$), luego agrega el siguiente número de la lista y así sucesivamente hasta que hayas agregado todos los números. Al final dividirías tu suma por cuántos números tenías (en este caso 8). Este es exactamente el tipo de situación en la que podríamos usar un bucle! A continuación, se muestra el código en python para sumar esta lista de números.

esta es la lista de números de los que nos gustaría encontrar la media

```
precip=[2,7,1,9,0,2,4,5]
```

[Copiar contenidos](#)

creamos una variable total a la que podemos añadir los elementos de la lista

```
total=0
```

en esta siguiente línea decimos que repetiremos este bucle de adición 8 veces
Durante este bucle, la variable i tomará los valores 0-7, aumentando en 1 en cada pase

for i in range(8):

 en cada paso sumamos el siguiente número al total que tenemos hasta ahora
 total=total+precipitaciones[i]

finalmente dividiremos por el número de elementos para obtener la media

```
mean_precip=total/8
```

Los bucles pueden ser difíciles cuando eres nuevo en la programación, así que puede llevarte un poco más de tiempo realmente entender tu cabeza alrededor de ellos.

- Intenta escribir un bucle para hacer el trabajo del pájaro bebedor de arriba. Tu bucle debería imprimir el `cadena` 10 veces.
- Intenta escribir un bucle para sumar una lista de 3 cadenas. Por ejemplo, si sumas la lista `caliente` ["line", "bling"] tu bucle debería crear la variable `total` = "hot line bling".

7. Funciones

Las funciones son cajas de computación pequeñas que escribimos en Python. Las funciones toman una entrada (podríamos llamarla x), haz algunos cálculos con ello y muestra el resultado.

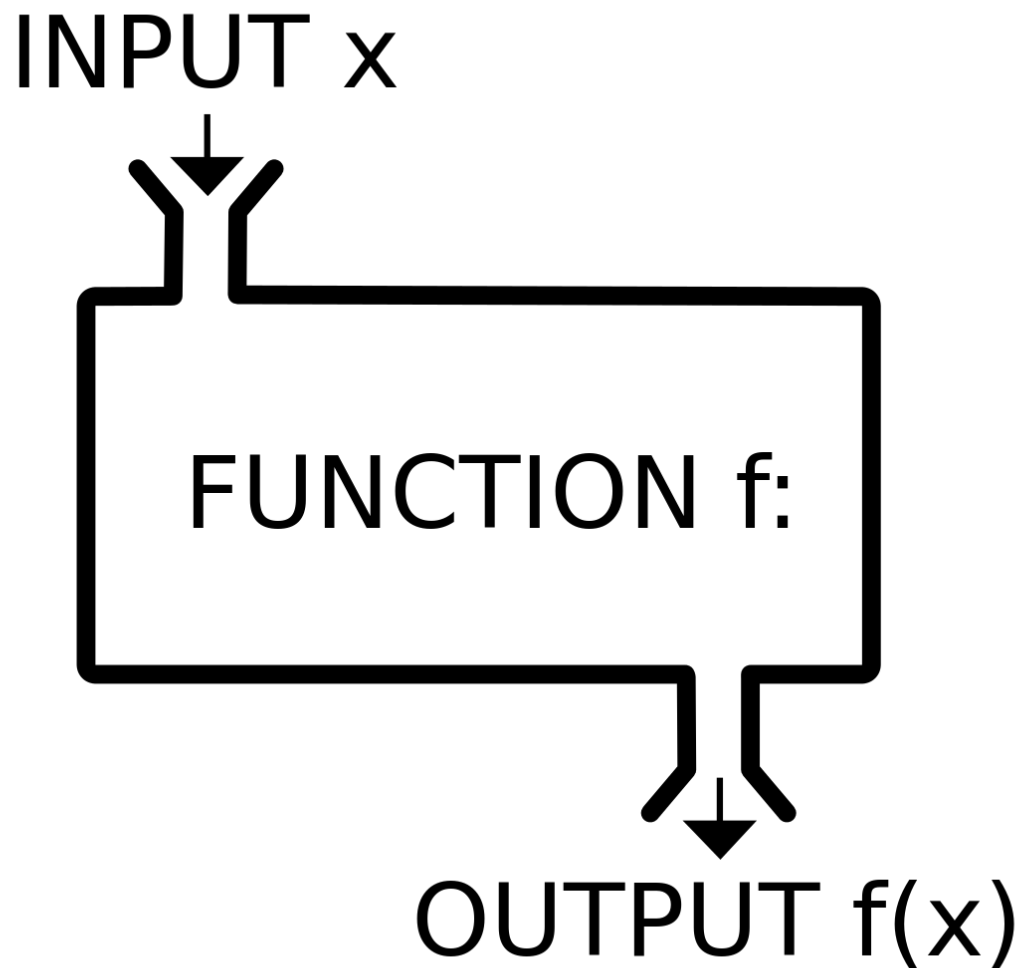


image credit: <https://hackernoon.com/a-simple-introduction-to-one-way-functions-a2429d427546>

Ya hemos estado usando dos funciones, `elimprimirfunción` y `eltipofunción`. `Elimprimirfunción` toma una entrada (hemos utilizado enteros, flotantes y cadenas) y copia su entrada a la consola.

Las funciones en Python son geniales porque nos permiten repetir cosas fácilmente. Digamos que tienes algunos datos que quieres procesar, como tasas de precipitación diarias para algunos sitios diferentes. Nos gustaría saber el media de cada una de estas muestras.

```
daily_precipitation_edinburgh=[2,7,1,9,0,2,4,5]
daily_precipitation_glasgow=[5,5,3,6,7,3,2,8]
daily_precipitation_dundee=[4,2,5,7,2,6,8,7]
```

Copiar contenido

Podríamos hacer esto como lo hicimos en el ejemplo anterior, copiando y pegando nuestro código 3 veces, (o nosotros podríamos incluso hacer un bucle dentro de otro bucle) pero esto es desordenado y como aspirantes a Pythonistas podemos hacerlo mejor.

En su lugar, podemos hacer una función que calculará la media de lo que pongamos.

Ten en cuenta que en python hacemos una función definiéndola (usamos `def` palabra clave). A continuación, escribimos un función que calcula la media de una lista de 8 números.

La palabra clave `def` le dice a Python que estás a punto de crear una función el nombre de la variable que viene después de '`def`' es el nombre de la función

en este caso nuestra función se llama `my_mean`

`x` es un sustituto para nuestra entrada. Lo llamamos `x` pero podría haber sido cualquier variable nombre en absoluto

`def mi_media(x):`

Creamos una variable total a la que podemos agregar los elementos de la lista.

`total=0`

en esta siguiente línea decimos que repetiremos este bucle de adición 8 veces

Durante este bucle, la variable `i` tomará los valores de 0 a 7, aumentando de 1 en 1.

en cada pase

`por en rango(8):`

En cada pase, añadimos el siguiente número al total que tenemos hasta ahora.

`total=total+x[i]`

Finalmente, dividiremos por el número de elementos para obtener la media.

`mean_value=total/8`

en esta línea final devolvemos la respuesta

el resto de las variables aquí, como `total`, serán desechadas

`return mean_value`

Copiar contenido

Ahora que hemos definido esta función, podemos usarla de la misma manera que usamos `elimprimir` función. Nosotros ingresa un valor y la función hará su trabajo y te devolverá la media. Para calcular la media de la precipitación en Edimburgo usaríamos:

`mean_precipitation_edinburgh=my_mean(daily_precipitation_edinburgh)`

Copiar contenidos

y obtendríamos la respuesta 3.75

- Intenta usar esta función para encontrar la precipitación media en Glasgow y Dundee

Ahora que tenemos esta función, podríamos usarla una y otra vez sin tener que reescribir el bucle para calcula la media, pero solo para listas de longitud 8.

- Esta función solo funcionará correctamente para listas que tengan 8 elementos de longitud. Intenta modificar esto. función para calcular la media de cualquier lista que se ingrese. Sugerencia: puedes encontrar la longitud de una lista `x` mediante `usandolen(x)`

8. declaraciones if-else

Las sentencias if else se utilizan en Python para que un script pueda hacer diferentes cosas basadas en algunas variable. Podemos querer un guion que nos motive a trabajar duro y jugar duro. Así que si hoy es un El viernes nos dice que vayamos a casa temprano, pero durante el resto de la semana nos anima a trabajar duro.

```
# establecer la variable hoy
today='Thursday'
```

Copiar contenido

```
comprobar si hoy es viernes
si hoy=='Viernes':
    si hoy es viernes, entonces esto se imprime
    ¿Por qué no ir a casa temprano?
sino:
    si hoy no es viernes, entonces esto se imprime
    ¡El trabajo duro es una virtud! ¡Puedes hacerlo! ¡Creo en ti!
```

9. Importaciones (paquetes)

Lo último que debes saber al comienzo de tu viaje con Python son los paquetes (complementos). oír a la gente hablar sobre 'paquetes', 'módulos' y 'bibliotecas' en Python, pero se refieren principalmente a la same thing. Packages are other pieces of code that we can use in Python once we import them. When instalaste anaconda también instalaste más de 100 paquetes que aún no hemos utilizado. Esto puede parecer daunting, but don't worry, you won't need to learn them all, and the idea of packages can seem confusing al principio.

Imagine packages to be your friends with a range of different skills. When you need something done you puedes invitar a tu amigo a hacer el trabajo por ti (son desinteresados así). Déjame presentarte rápidamente a 3 amigos que deberías conocer.

- **numpy** - numpy es excelente para hacer cosas con números. Como sumas, medias, varianzas, matrices álgebra, multiplicación, etc. Si necesitas hacer algunas sumas, numpy es el que debes llamar.
- **scipy**- scipy es realmente bueno en matemáticas y estadísticas. Invítalos a hacer regresión lineal, integración numérica y té, etc.
- **matplotlib**- matplotlib es tu amigo artístico y le gusta ayudarte a dibujar gráficos bonitos

Puedes invitar a un amigo a casa al importar invítalos. Cuando los invitas a tu casa, también puedes darles un apodo más corto (o incluso más largo) si no te gusta su nombre real. En el siguiente fragmento, invitaremos numpy sobre y hacer que calculen una media y una desviación estándar para nosotros. Le daremos a numpy el apodo 'np' porque así es como todos los demás los llaman, pero también podríamos darles el nombre voluntad, vivian, g o paquete_python_para_números.

```
# primero invitamos a numpy: import numpy
al mismo tiempo le damos a numpy el apodo de np : como np
import numpy como np
```

Copiar contenido

aquí están los datos que queremos investigar

```
x=[1,3,6,2,8,4,1]
```

Ahora que numpy (np) está aquí, podemos pedirles que usen su función media.

```
x_mean=np.mean(x)
```

entonces le pedimos a numpy que use su función de desviación estándar en los datos

```
x_desviación_estándar=np.std(x)
```

```
# finally we will print the result
```

```
imprimir(x_media)
```

```
imprimir(x_desviacion_estandar)
```

Los paquetes numpy, scipy y matplotlib son amigos muy populares y no tengo tiempo para mostrarte todo.

sus talentos en este tutorial, pero si los buscas en Google, podrías descubrir fácilmente más sobre sus muchas habilidades.

Nota: Otro amigo que podrías querer conocer es pandas. Si deseas cargar datos desde hojas de excel

y manipularlo, entonces pandas es la persona a invitar.

10. Encontrar ayuda en línea

Una de las mejores cosas sobre Python es la comunidad de Python. Hay millones de personas usando Python.

y así hay muchas y muchas personas tratando de hacer cosas similares. Esto significa que para casi cualquier

pregunta que puedas tener en python, como *¿cómo encuentro el valor máximo en una lista de números?* o *'cómo yo*

encuentra el enésimo elemento en una lista' hay buenas discusiones en línea por ahí. Para la mayoría de los usuarios de Python, esto es LO

modo de trabajar. ¿Por qué perder mucho tiempo luchando por escribir código cuando la respuesta a tu problema es solo un

¿a un google buscar?

Receta para buscar fragmentos de código

1. averigua qué estás tratando de hacer y reduce eso a una frase corta
2. ve a google y escribe tu frase + 'python'
3. Revisa algunos enlaces (stackoverflow es el sitio principal que querrás) y lee sobre el problemas que otras personas están tratando de resolver
4. Copia y pega cualquier buena solución que encuentres
5. Modifica el código si es necesario

Sé amable y vota positivamente las soluciones útiles

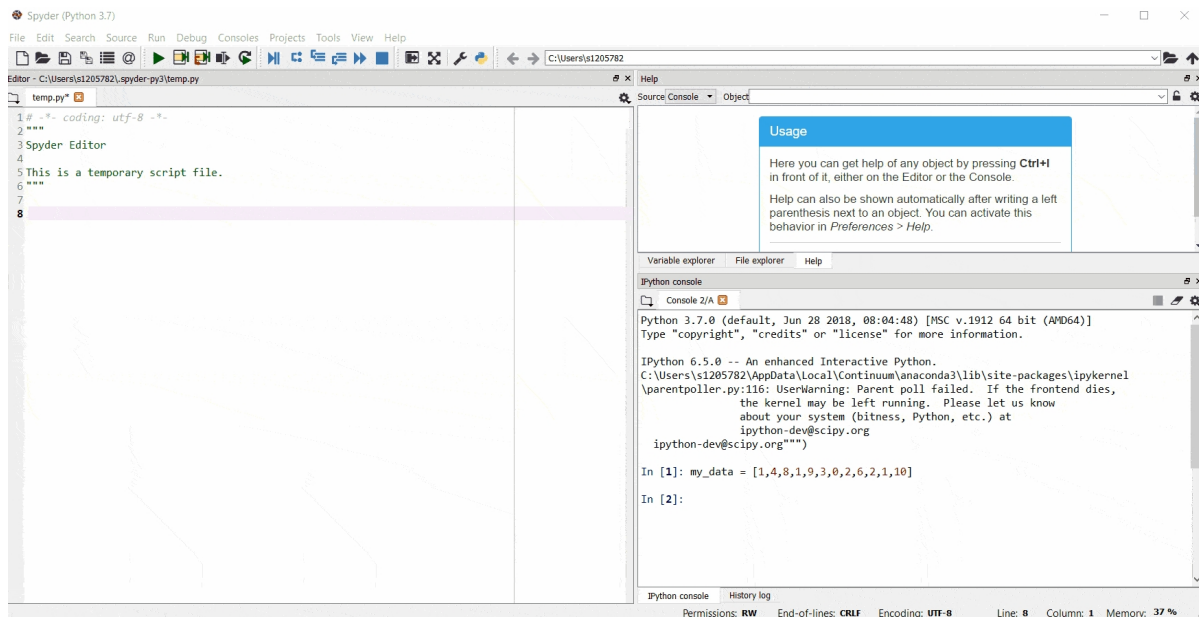
Vamos a través de un ejemplo. Tengo algunos datos y quiero ordenar los valores en la lista en orden descendente.

order. In the example below I would like the first item to be the 10, then the 9 then the 8 etc.

```
my_data=[1,4,8,1,9,3,0,2,6,2,1,10]
```

Copiar contenido

Podría escribir un bucle para hacer esto por mí y sería una buena práctica para mis habilidades en Python, pero esta vez estoy tengo prisa y preferiría no reinventar la rueda. Así que lo buscaré en Google en su lugar.



El enlace a la respuesta que encontré [aquí](#).

Como puedes ver, busqué la pregunta en Google y tuve que revisar dos enlaces para encontrar una pregunta que coincidiera. También tuve que redefinir mi pregunta en el camino para hacerla más específica sobre lo que quería. Cuando yo encontré una solución en stackoverflow, la copié y la edité antes de pegarla en mi consola.

Buscar respuestas a tus problemas de programación es una parte importante de escribir código de manera efectiva. Esto es algo a lo que deberías acostumbrarte. Al buscar en línea, ¿puedes encontrar formas de lograr el siguiente:

- agrega el número 50 al final de la lista `mis_datos` sin simplemente escribirlo manualmente
- convertir la cadena `15a` un entero

11. Aprendiendo más Python básico

Hay tantos recursos para aprender Python por ahí, más de lo que una persona podría esperar tener incluso. miré. Por lo tanto, ¿no puedo darte un método definitivo para seguir aprendiendo Python. Sin embargo, aquí son algunos recursos que he encontrado y que he disfrutado.

Cursos interactivos en línea

Creo que estas son una muy buena manera de empezar tanto con Python si eres nuevo en ello, pero también con Paquetes de Python que nunca has explorado antes. El beneficio de estos es que te brindan mucho de estructura a tu aprendizaje y se aseguran de que entiendas lo básico antes de avanzar.

Estos cursos tienen material para que leas y/o veas, así como ejercicios de computación interactivos. En los ejercicios interactivos donde completas los espacios en blanco de algunos scripts de codificación y recibes retroalimentación instantánea sobre lo que has escrito. Hay varios de estos, pero los dos con los que estoy personalmente familiarizado son

- [código academia](#)- artículos y ejercicios de codificación gratuitos. Pago por cuestionarios y otros extras.
- [datacamp](#)- vídeos, artículos y ejercicios de codificación. Las primeras secciones de cada módulo son gratuitas pero las cosas más avanzadas son de pago.

Blogs

Tal como [nuestro club de codificación](#) | Estos son grandes recursos para seguir, especialmente si encuentras blogs de personas que son trabajando en el mismo campo que tú. Pueden presentarte herramientas de Python específicas para tu dominio. Si tú Si ya conoces un poco de Python, estos son buenos recursos para llevarte de intermedio a avanzado.

Libros

Tengo opiniones mixtas sobre los libros cuando se trata de codificación. Es mi propia creencia que la única manera de aprender a codificar es realmente sentarse y hacerlo. Sin embargo, un libro puede ser una buena síntesis de conocimiento y ellos tienen su lugar. Un libro clásico es Numeric Recipes en Python, pero hay demasiados muchos para mencionar aquí y nuevos están saliendo todo el tiempo.

Desafíos

Hay algunos sitios web que están específicamente dedicados a albergar desafíos para que las personas completen o para competir. Si eres el tipo de persona que disfruta de un buen desafío para mejorar tu aprendizaje, entonces esto podría ser para ti.

- [kaggle](#)- Esta es una plataforma que alberga competiciones de aprendizaje automático para aprender, para felicitaciones e incluso por premios. Si estás interesado en profundizar en el aprendizaje automático y la ciencia de datos entonces este es un buen centro.
- [proyecto euler](#) Este es un sitio que alberga un catálogo de desafíos de programación para que tú completo. Es una manera realmente agradable de encontrar desafíos donde puedes aplicar lo que has aprendido. También hay diferentes dificultades, por lo que deberías poder encontrar un desafío sin importar donde te encuentras en tu ruta de Python.

Mantente al día y aprende sobre nuestros recursos más recientes siguiendo nos en [Twitter](#)!

¡Nos encantaría escuchar tus comentarios, por favor completa nuestra encuesta!

Contáctanos con cualquier pregunta sobre ourcodingclub@gmail.com

Tutoriales relacionados:

- [Introducción a Python](#)
- [Análisis de Datos en Python con Pandas y Matplotlib](#)
- [Analizando datos de ciencias de la Tierra y clima con Iris](#)
- [Números en Python con NumPy](#)
- [Tomodelado de PIC en Python](#)
- [Análisis de series temporales con pandas](#)