Empezando a Programar en Python

October 7, 2017

1 Comenzando a programar en Python

El presente contenido pretende mostrar los aspectos básicos del lenguaje de programación Python. Se parte de que la persona que hace uso de este contenido parte con conocimientos previos de programación y por lo tanto conoce y domina nociones básicas de programación.

2 Python como Calculadora

En su forma más básica Python puede ser usado como una simple calculadora.

```
In [1]: #Sumamos dos números
        5 + 5
```

```
Out[1]: 10
```

```
In [2]: #Restamos dos números
        11 - 5
```

```
Out[2]: 6
```

```
In [3]: #Multiplicamos dos números
        13 * 9
```

```
Out[3]: 117
```

```
In [4]: #Dividimos dos números
        9 / 8
```

```
Out[4]: 1.125
```

```
In [6]: #Obtenemos el módulo, es decir, el resto de dividir dos números
        9 % 8
```

```
Out[6]: 1
```

```
In [7]: #Hacemos uso del operador exponente
        2 ** 5
```

```
Out[7]: 32
```

3 Variables en Python

Una variables en Python es un espacio en la memoria del ordenador donde se almacenará un valor que podrá cambiar durante la ejecución del programa. Para asignar un valor a una variables se hace uso del operador (=).

```
In [8]: #Le asignamos el valor 5 a la variable x
        x = 5
        #Mostramos el valor de la variable x
        print(x)
```

```
5
```

En una variable podemos almacenar valores y operar con las variables.

```
In [9]: #Almacenamos el valor 5 en la variable x y el valor 6 en la variable 6
        x = 5
        y = 6
        #Guardamos en la variable producto la multiplicación de estas dos variables
        producto = x * y
        #Mostramos el resultado
        print(producto)
```

```
30
```

Además de tomar valores numéricos una variable puede tomar valores de tipo string o de tipo booleanos.

```
In [10]: #Le asignamos el string Hola a la variable saludo
         saludo = 'Hola'
         #Mostramos por pantalla el contenido de saludo
         print(saludo)
```

```
Hola
```

```
In [11]: #Le asignamos el valor True a la variable boolean
         boolean = True
         #Mostramos por pantalla el valor de la variable boolean
         print(boolean)
```

```
True
```

Dependiendo del tipo de valor que tome una variable, los operadores suma, producto etc le afectarán de una forma u otra. Por ejemplo si una variable es de tipo numérico afecta de una forma esperada. Sin embargo si esta variable es de tipo string el operador producto y el operador suma afectan de distinta forma.

```
In [12]: #Guardamos en las variables saludo y nombre dos strings
         saludo = 'Hola'
         nombre = 'Pepe'
         #Hacemos uso del operador suma para ver su efecto
         print(saludo + nombre)
```

```
HolaPepe
```

```
In [13]: #Guardamos en saludo un string
         saludo = 'Hola'
         #Hacemos uso del operador producto para ver su efecto
         print(saludo * 3)
```

```
HolaHolaHola
```

Podemos ver como en el caso de strings el operador suma concatena, mientras que en el caso del producto se repite el string tantas veces como le indiquemos al operador.

4 Listas

Ya hemos visto como una variable solo puede almacenar un único valor ya sea decimal (float), entero (int), string(str), booleano etc. En determinadas ocasiones podemos querer guardar en una misma variable distintos valores, por ejemplo el tamaño de las distintas habitaciones en nuestra casa, para ello Python ofrece las listas. Las listas nos permiten almacenar distintas variables. Para crear una lista en Python hacemos uso de los corchetes ([ ]) y dentro separamos por comas los distintos valores que queremos almacenar.

```
In [38]: #Nos creamos distintas variables que almacena en millones el número de habitantes
         #en ciudades
         Madrid = 3.16
         Barcelona = 1.6
         Valencia = 0.790
         Sevilla = 0.690
         #Nos creamos una lista de ciudades que contiene la población de distintas ciudades
         lista_ciudades = [Madrid, Barcelona, Valencia, Sevilla]
         #Mostramos el contenido de la lista
         print(lista_ciudades)
```

```
[3.16, 1.6, 0.79, 0.69]
```

Una lista puede contener distintos tipos de variables.

```
In [39]: #Nos creamos una lista de ciudades
         lista_ciudades = ['Madrid', Madrid, 'Barcelona', Barcelona, 'Valencia', Valencia, 'Sevi
         #Mostramos el contenido de la lista
         print(lista_ciudades)
```

```
['Madrid', 3.16, 'Barcelona', 1.6, 'Valencia', 0.79, 'Sevilla', 0.69]
```

A la hora de seleccionar elementos de una lista debemos de tener encuenta que en Python el primer índice es el 0 y no el 1, esto quiere decir que el primer elemento de una lista se encuentra en el índice 0. Python también nos permite seleccionar elementos de derecha a izquierda, para ello bastará con indicar los índices con valores negativos, en este caso el primer elemento (el último de nuestra lista) toma el valor de -1.

```
In [19]: #Accedemos al número de habitantes de la ciudad de Madrid
         print(lista_ciudades[1])
```

```
3.16
```

```
In [20]: #Accedemos al número de habitantes de Sevilla
         print(lista_ciudades[-1])
```

```
690500
```

Python nos permite seleccionar diferentes elementos consecutivos de una lista haciendo uso del operador ':'. El formato para seleccionar es el siguiente lista[start:end], donde start es el índice inicial y end el índice final, debemos tener encuenta que el índice final no esta incluido a la hora de seleccionar. También podemos querer seleccionar todos los elementos a partir de un determinado índice para ello debemos hacer uso el formato lista[indice:] que nos seleccionará todos los elementos a partir del índice (este incluido) o el formato lista[:indice] que nos seleccionará todos los elementos de la lista hasta el índice (este no incluido).

```
In [22]: #Accedemos a los tres primeros elementos de nuestro lista
         print(lista_ciudades[0:3])
         print(lista_ciudades[:3])
```

```
['Madrid', 3.16, 'Barcelona']
['Madrid', 3.16, 'Barcelona']
```

```
In [23]: #Accedemos desde el tercer elemento hasta el final
         print(lista_ciudades[2:])
```

```
['Barcelona', 1.6, 'Valencia', 790000, 'Sevilla', 690500]
```

Python permite renombrar o cambiar elementos de una lista, para ello basta con seleccionar el índice y indicarlo su nuevo valor.

```
In [33]: #Cambiamos el valor del nombre de la ciudad de Barcelona
         lista_ciudades[2] = 'Bcn'
         #Mostramos nuestra lista
         print(lista_ciudades)
```

```
['Madrid', 3.16, 'Bcn', 1.6, 'Valencia', 790000, 'Sevilla', 690500]
```

Para eliminar elementos de una lista disponemos de la función del() mediante la cual le indicamos los elementos que queremos eliminar mediante el índice.

```
In [34]: #Eliminamos el nombre de Sevilla de nuestra lista
         del(lista_ciudades[-2])
         print(lista_ciudades)
```

```
['Madrid', 3.16, 'Bcn', 1.6, 'Valencia', 790000, 690500]
```

EL método insert() de Python nos permite agregar un elemento a una lista en la posición que nosotros estimemos adecuada.

```
In [35]: #Insertamos la ciudad de Sevilla en la posición que se encontraba antes
         lista_ciudades.insert(5, 'Sevilla')
         print(lista_ciudades)
```

```
['Madrid', 3.16, 'Bcn', 1.6, 'Valencia', 'Sevilla', 790000, 690500]
```

La método append() nos permite agregar elementos a una lista, pero esta los agrega al final de la lista.

```
In [40]: #Agregamos la ciudad Granada y su población
         lista_ciudades.append('Granada')
         lista_ciudades.append(0.234)
         #Mostramos el contenido
         print(lista_ciudades)
```

```
['Madrid', 3.16, 'Barcelona', 1.6, 'Valencia', 0.79, 'Sevilla', 0.69, 'Granada', 0.234]
```

Lo relizado anteriormente se puede hacer en un único paso haciendo uso del método extend que nos permite concatenar elementos a una lista, es decir, en el fondo es una concatenacion de listas.

```
In [41]: #Concatenamos elementos a una lista.
         lista_ciudades.extend(['Oviedo', 0.22])
         print(lista_ciudades)
```

```
['Madrid', 3.16, 'Barcelona', 1.6, 'Valencia', 0.79, 'Sevilla', 0.69, 'Granada', 0.234, 'Oviedo'
```

El método reverse() nos permite invertir el orden de una lsita.

```
In [45]: #Invertimos el orden de la lista.
         lista_ciudades.reverse()
         print(lista_ciudades)
```

```
[0.22, 'Oviedo', 0.234, 'Granada', 0.69, 'Sevilla', 0.79, 'Valencia', 1.6, 'Barcelona', 3.16, 'M
```

También podemos hacer uso del método count() que nos permite ver el número de veces que aparece un string o valor numérico dentro de una lista.

```
In [46]: print(lista_ciudades.count('Sevilla'))
```

```
1
```

El método index() nos permite conocer el índice en el cuál se encuentra un determinado elemento de una lista. En caso de que tengamos un elemento repetido en una lista, nos devolverá el índice del primer elemento.

```
In [47]: print(lista_ciudades.index('Granada'))
```

```
3
```

La función in nos permite saber si un determinado elemento se encuentra o no en una lista. Esta función devuelve True en caso de que el elemento se encuentre en la lista y False en caso de que no se encuentre.

```
In [48]: print('Roma' in lista_ciudades)
```

```
False
```

```
In [49]: print('Madrid' in lista_ciudades)
```

```
True
```

El método pop lo que hace es eliminar el último elemento de una lista.

```
In [50]: #Hacemos uso de pop para eleminar el último elemento de la lista
         lista_ciudades.pop()
         print(lista_ciudades)
```

```
[0.22, 'Oviedo', 0.234, 'Granada', 0.69, 'Sevilla', 0.79, 'Valencia', 1.6, 'Barcelona', 3.16]
```

El método remove() también nos permite eliminar elementos de una lista.

```
In [51]: #Eliminamos la ciudad oviedo
         lista_ciudades.remove('Oviedo')
         print(lista_ciudades)
```

```
[0.22, 0.234, 'Granada', 0.69, 'Sevilla', 0.79, 'Valencia', 1.6, 'Barcelona', 3.16]
```

Para el caso de listas en Python el operador suma lo que hace es concatenar listas mientras que el operador producto repite una lista tantas veces como se le indique.

```
In [52]: #Nos creamos dos listas
         lista1 = [1,2,3]
         lista2 = [4,5]
         #Concatenamos con el operador suma
         lista3 = lista1 + lista2
         print(lista3)
```

```
[1, 2, 3, 4, 5]
```

```
In [53]: #Replicamos lista3, 5 veces
         print(lista3 * 5)
```

```
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
```