Fundamentos Python 4 — Funciones

Las herramientas del científico de datos

Juan Manuel Moreno — jmmoreno@profesorescol.imf.com

IMF
Smart Education

ÍNDICE

IMF
Smart Education

01 Objetivos unidad 1

IMF
Smart Education

1.– Objetivos Unidad 1

- Conocer los principales fundamentos de Python.
- Saber instalar Jupyter Notebook, la herramienta que vamos a utilizar para trabajar con Python.
- Realizar desarrollos básicos en Python a través de Jupyter Notebook
- Saber cómo declarar, procesar y distinguir distintos tipos de variables.
- Conocer y manejar las sentencias condicionales If — Else.
- Conocer el funcionamiento de los bucles en Python, for y while.
- Trabajar con las principales estructuras de datos en Python: Tuplas, listas y diccionarios de datos.
- Comprender cómo modularizar los programas a través de funciones.
- Resolver problemas de diferente dificultad.

02

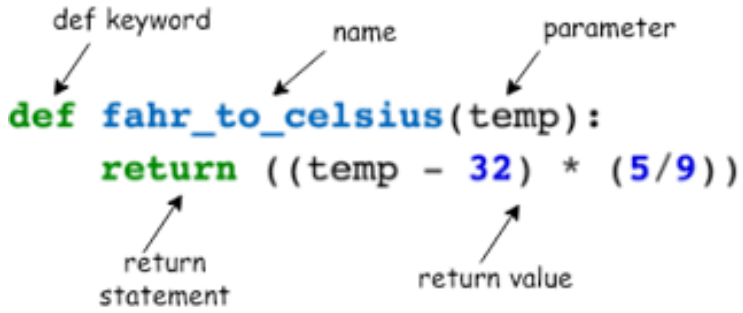
Funciones

IMF
Smart Education

2.– Funciones

2.1.– Definición de una función

- Muy importantes para estructurar el código fuente de un programa.
- Sirven para abstraer partes de un programa.
- Solamente se procesan cuando son llamadas.
- En general, las funciones casi siempre reciben y devuelven parámetros, aunque no siempre tiene que ser así.
- En Python una función se define con `def nombre_función()`



2.2.– Función simple

- Un ejemplo de función básica sería aquella que no reciba ningún parámetro y que tampoco devuelva ningún parámetro.
- Para invocar una función utilizaremos el nombre_funcion() cualquier parámetro que reciba irá entre paréntesis.

```
1  def saludar():
2      print('Hola !')
```

```
1  saludar()
```

```
Hola !
```

2.3.- Parámetros de una función

- Para pasar un parámetro a una función simplemente escribiremos el nombre de la variable que va a operar dentro de la función entre los paréntesis del nombre de la misma.

- Posteriormente, cuando llamemos a la función tendremos que pasarle una variable a los parámetros que hayamos definido.

```
1 def saluda_nombre(nombre):
2     print('Hola', nombre, '!')
```

```
1 saluda_nombre(nombre='Juan')
```

```
Hola Juan !
```

2.3.- Parámetros de una función

- Cuando utilicemos más de un parámetro, simplemente tendremos que utilizar más variables tanto en la definición de la función, como en la llamada a la misma.

```
1 def aplica_iva(nombre_artículo, precio):
2     print('El precio total de', nombre_artículo, 'es: ', round(precio + (precio*0.21),2))
```

```
1 aplica_iva(nombre_artículo='NVIDIA GTX 2080', precio=2045)
```

```
El precio total de NVIDIA GTX 2080 es:  2474.45
```

2.4.– Parámetros de salida

- Lo más normal es querer devolver uno o más resultados mediante la función, no solamente mostrar un mensaje por pantalla, para eso, se utiliza el comando **return**

- Todas las variables que vayan después del return serán devueltas por la función.

```
1  def obten_precio_total(precio):
2      precio_final = round(precio + (precio * 0.21), 2)
3
4      return precio_final
```

```
1  precio_total = obten_precio_total(precio=2000)
```

```
1  print('El precio final del artículo es: ', precio_total, 'euros')
```

```
El precio final del artículo es:  2420.0 euros
```

Seguimiento práctico del contenido

A partir de aquí, vamos a ver cómo declarar funciones y cómo funcionan los parámetros tanto de entrada como de salida.

1_5_Funciones.ipynb

Para conocer cómo procesar archivos csv, json y xml se seguirá el siguiente notebook.

1_6_Gestion_archivos.ipynb

IMF

Smart Education