Funciones de usuario I

Contenidos

Introducción

al indice

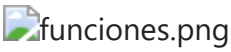
En este notebook tienes la primera parte de una guía para orientarte en el uso de funciones en Python... pero antes de seguir con todo el "rollo..."

Contenidos

Definición y sintaxis

Se trata de bloques de código que encapsulan una serie de operaciones. Se usan para modular nuestro código y evitar escribir código de más.

Mediante las **funciones** podemos encapsular código en formato entrada/salida. Por lo que si tienes un código repetitivo, que depende de ciertos inputs, las funciones pueden ser una buena solución.


funciones.png

Es una manera de agrupar conjuntos de operaciones en módulos. **¿Cuándo usarlas?** Cuando tengamos varias operaciones que ejecutamos repetidamente en distintas partes del código. En ese caso, encapsulamos las operaciones en

una función, y cada vez que haya que realizar tal operativa, llamamos a la función, y en una sola línea de código tenemos ejecutada esas operaciones.

Hasta ahora hemos estado utilizando funciones *built-in*, para operaciones sencillas como `len()`, `sum()` o `max()`. En este Notebook aprenderas a crear tus propias funciones.

La sintaxis es:

```
def nombre_funcion(entrada):
    operaciones varias
    return output
```

Fíjate que sigue la **sintaxis de línea** vista en Notebooks anteriores (tabulación detrás de los dos puntos, por ejemplo).

Además, todo lo que va después del `return` es ignorado, puesto que es la salida. En el `return` acaba la función. Ahora bien, eso no quiere decir que haya un único return. Si introducimos una sentencia `if/else`, podremos poner returns diferentes dependiendo de qué condición se cumpla.

Ejemplos

al indice

Vamos a crear nuestra primera función

In [2]:
```
# Vamos a crear un conversor de km a millas
def conversor_km_millas(distancia):
    millas = 0.62 * distancia
    return round(millas,1)
```

Al ejecutar el código anterior, no corre nada, simplemente almacenamos en memoria la función para usarla posteriormente. **Ahora podremos llamar a la función tantas veces como queramos, desde cualquier parte del codigo.**

In [6]:
```
conversor_km_millas(22215)
```

Out[6]: 13773.3

In [8]:
```
millas = conversor_km_millas(22215)
print(millas)
```

```
13773.3
```

Las funciones no tienen por qué llevar argumentos. Eso sí, **es obligatorio** poner los parentesis, tanto en la declaración, como luego al llamar la función.

In [11]:
```
from datetime import datetime

def que_hora_es():
    now = datetime.now().time()
    return now
```

In [12]:
```
print(que_hora_es())
```

```
10:27:52.561125
```

In [13]:
```
def que_hora_es():
    now = datetime.now().time()
    print(now)
que_hora_es()
```

```
10:28:49.506155
```

In [ ]: