Funciones

October 17, 2020

1 Funciones

Podemos crear funciones, usando la palabra reservada `def` y podemos setear parámetros dentro del paréntesis. Una función puede no tener parámetros.

[4]:
```
def print_mensaje():
    print("Hola a todos!!!")

def print_hello(nombre: str) -> str:
    print(f"hola {nombre}")


print_mensaje()
print_hello("Emmanuel")
print_hello(2)
```

```
Hola a todos!!!
hola Emmanuel
hola 2
```

Cuando se envían argumentos a una función, y estos se reciben los parámetros en el orden que fueron defindos, estos son `argumentos por posición`

[5]:
```
def foo(a, b) -> float:
    return a * b

res = foo(2, 3)
print(res)
```

```
6
```

También se puede enviar los parámetros con sus nombres, ya no estamos obligados a llamar a la función respetando el orden con el que fueron definidos

[14]:
```
res = foo(b=3, a=2)
print(res)
```

```
6
```

Podemos usar parámetros por defecto, dónde le asignamos el valor del parámetro en la defincón de la función, de esta forma, ese argumento ya va a tener un valor, por lo que podemos ignorarlo durante la llamada

[15]:
```
def bar(a, b=3):
    return a * b

print(bar(2))
print(bar(3, 4))
print(bar(b=3, a=4))
```

```
6
12
12
```

[13]:
```
def bar(a, b=3):
    return a * b, a, b


res1, res2, res3 = bar(2)

print(res1)
print(res2)
print(res3)
```

```
6
2
3
```

1.1 Ejemplo

[6]:
```
def iva():
    iva_percent = 0.21
    precio = int(input("Cual es el precio del producto"))
    print(f"El impuesto es de ${precio * iva_percent}")

iva()
```

```
Cual es el precio del producto100
El impuesto es de $21.0
```

[8]:
```
def iva(precio, iva_percent = 0.21):
    return precio * iva_percent

impuesto_a_pagar = iva(100, 0.35)

print(f"El impuesto es de ${impuesto_a_pagar}")
```

```
El impuesto es de $35.0
```

[ ]:
```
# magick methods
```