Clase2_1_Funciones

March 24, 2023

1 Seminario de Lenguajes - Python

1.0.1 Definición de funciones. Parámetros

2 Funciones en Python: una forma de definir procesos o subprogramas

Recordemos el **desafío 5** de la clase:

> Volvemos a procesar las películas de Harry Potter.

Queremos saber: - cuál fue la duración, en minutos, promedio; y - **qué** películas duran más que el promedio, en minutos.

Veamos un posible pseudocódigo de la solución:

```
Ingresar los nombres y duración  de las películas.
Calcular el promedio.
Mostrar qué  películas duran más que el promedio.
```

Podríamos pensar en dividir en tres procesos:

- En Python, usamos **funciones** para definir estos procesos.
- Las funciones pueden recibir **parámetros**.
- Y también retornan siempre un valor. Esto puede hacerse en forma implícita o explícita usando la sentencia **return**.

3 Ya usamos funciones

- float(), int()
- input(), print()
- sum()

En estos ejemplos, sólo **invocamos** a las funciones ya predefinidas de Python.

4 Podemos definir nuestras propias funciones

```
def nombre_funcion(parametros):
    sentencias
    return <expresion>
```

- **IMPORTANTE:** el cuerpo de la función debe estar **indentado**.

5 La función para el primer proceso del desafío

[29]:
```
def ingreso_pelis():
    """ Esta función retorna un diccionario con los nombres y duración de␣
 ↪películas """

    peli = input("Ingresa el nombre de una película de Harry Potter nombre␣
 ↪(<FIN> para finalizar)")
    dicci = {}
    while peli != "FIN":
        cant_minutos = int(input(f"Ingresa la duración de la película  {peli}"))
        dicci[peli] = cant_minutos
        peli = input("Ingresa el nombre de una película de Harry Potter nombre␣
 ↪(<FIN> para finalizar)")
    return dicci
```

[30]:
```
pelis = ingreso_pelis()
pelis
```

```
Ingresa el nombre de una película de Harry Potter nombre (<FIN> para
finalizar)aa
Ingresa la duración de la película  aa152
Ingresa el nombre de una película de Harry Potter nombre (<FIN> para
finalizar)bb
Ingresa la duración de la película  bb147
Ingresa el nombre de una película de Harry Potter nombre (<FIN> para
finalizar)FIN
```

[30]:
```
{'aa': 152, 'bb': 147}
```

- **¡IMPORTANTE!** Definición vs. invocación.

6 El docstring

- Es una secuencia de caracteres que describe la función.
- Se sugiere siempre utilizar triples ”.
- Son procesados por el intérprete.

[32]:
```
help(print)
```

```
Help on built-in function print in module builtins:

print(*args, sep=' ', end='\n', file=None, flush=False)
    Prints the values to a stream, or to sys.stdout by default.
```

```
    sep
      string inserted between values, default a space.
    end
      string appended after the last value, default a newline.
    file
      a file-like object (stream); defaults to the current sys.stdout.
    flush
      whether to forcibly flush the stream.
```

6.1 ¿Qué pasa si no incluyo el return?

[35]:
```
def demo_funcion_sin_return():
    var = 10
    print(var)

x = demo_funcion_sin_return()
print(x)
```

```
10
None
```

Ahora observemos este código que implementa una posible solución al segundo proceso:

[36]:
```
def calculo_promedio(dicci_duraciones):
    """ Esta función calcula el promedio de las duraciones de las películas␣
 ↪recibida por parámetro.
    dicci: es un diccionario de forma nombre_pelucula: duracion
    """
    suma = sum(dicci_duraciones.values())
    cant_pelis = len(dicci_duraciones)
    promedio = 0 if cant_pelis ==0 else suma/cant_pelis
    return promedio

calculo_promedio(pelis)
```

[36]:
```
149.5
```

- A diferencia de la función anterior, ésta tiene un parámetro.
- ¿Hay distintas formas de pasar parámetros en Python? ¿Cómo podemos probar esto?

7 Parámetros en Python

Veamos un ejemplo más sencillo: ¿qué podemos observar?

[37]:
```
def proceso_parametros_simples(param):
    "Esta función  modifica el parámetro recibido."

```

```
    print(f"El valor de param al ingresar a la función es {param}")
    param = 0
    print(f"El valor de param en la función es {param}")

num = 10
proceso_parametros_simples(num)
print(f"Luego de invocar a la función el valor de num es {num}")
```

```
El valor de param al ingresar a la función es 10
El valor de param en la función es 0
Luego de invocar a la función el valor de num es 10
```

7.0.1 Y ahora analicemos este otro ejemplo:

[38]:
```
def proceso_parametros_colecciones(param):
    """Esta función actualiza la primera posición de la colección recibida como␣
 ↪parámetro."""

    print(f"El valor de param al ingresar a la función es {param}")
    param[0] = "cero"
    print(f"El valor de param en la función es {param}")


lista = [100, 200, 300]
proceso_parametros_colecciones(lista)
print(f"Luego de invocar a la función el valor de lista es {lista}")
```

```
El valor de param al ingresar a la función es [100, 200, 300]
El valor de param en la función es ['cero', 200, 300]
Luego de invocar a la función el valor de lista es ['cero', 200, 300]
```

7.0.2 Entonces, ¿qué podemos decir sobre el pasaje de parámetros en Python?

#

Cuando pasamos un parámetro a una función, pasamos **una copia de la referencia** al objeto pasado

8 Miremos este otro ejemplo

[39]:
```
def proceso_parametros_colecciones_1(param):
    """Esta función actualiza la primera posición de la colección recibida como␣
 ↪parámetro."""

    mi_lista = param[:]
    print(f"El valor de param al ingresar a la función parametros_colecciones␣
 ↪es {mi_lista}")
```

```
    mi_lista[0] = "cero"
    print(f"El valor de param en la función parametros_colecciones es␣
 ↪{mi_lista}")

lista = [100, 200, 300]
proceso_parametros_colecciones_1(lista)
print(f"Luego de invocar a la función el valor de lista es {lista}")
```

```
El valor de param al ingresar a la función parametros_colecciones es [100, 200,
300]
El valor de param en la función parametros_colecciones es ['cero', 200, 300]
Luego de invocar a la función el valor de lista es [100, 200, 300]
```

- ¿Qué pasa en este caso?

9 ¿Podemos retornar más de un valor?

9.1 Desafio 1

Queremos definir una función que, dada una cadena de caracteres, retorne la **cantidad de vocales abiertas, vocales cerradas y la cantidad total de caracteres** de la misma.

10 Una posible solución

- ¿Qué tipo de dato retorna la función?

[41]:
```
def retorno_varios(cadena):
    """ ..... """
    cadena = cadena.lower()
    cant_aeo = cadena.count("a") + cadena.count("e") + cadena.count("o")
    cant_iu = cadena.count("i") + cadena.count("u")
    return cant_aeo, cant_iu, len(cadena)


algo = retorno_varios("Seminario de Python")
type(algo)
```

[41]:
```
tuple
```

11 ¿Cómo accedemos a los valores retornados?

- En el return se devuelve una tupla, por lo tanto, accedemos como en cualquier tupla:

[42]:
```
vocales_abiertas = retorno_varios("Seminario de Python")[0]
vocales_abiertas
```

[42]: 5

```
[43]: abiertas, cerradas, longitud = retorno_varios("Espero que este video sea corto")
      cerradas
```

[43]: 2

12 Los parámetros pueden tener valores por defecto

Observemos esta estructura. ¿De qué tipo es?

```
[44]: dicci_musica = {"bart": {"internacional": ["AC/DC", "Led Zeppelin", "Bruce␣
       ↪Springsteen"],
                              "nacional": ["Pappo", "Miguel Mateos", "Los Piojos",␣
       ↪"Nonpalidece"]
                              },
                     "lisa": {"internacional": ["Ricky Martin", "Maluma"],
                              "nacional": ["Lali", "Tini", "Wos"]}
                    }
```

```
[53]: def mi_musica(dicci_musica, nombre, tipo_musica="nacional"):
          """ .... """
          if nombre in dicci_musica:
              usuario = dicci_musica[nombre]
              for elem in usuario[tipo_musica]:
                  print(elem)
          else:
              print(f"¡Hola {nombre}! No tenés registrada música en esta colección")
```

```
[51]: mi_musica(dicci_musica, "bart")
```

```
Pappo
Miguel Mateos
Los Piojos
Nonpalidece
```

Si hay más de un argumento, los que tienen **valores por defecto siempre van al final de la lista de parámetros**.

Los parámetros formales y reales se asocian de acuerdo al **orden posicional**, pero invocar a la función con los parámetros en **otro orden** pero **nombrando al parámetro**.

```
[55]: mi_musica(dicci_musica, tipo_musica="internacional", nombre="lisa")
```

```
Ricky Martin
Maluma
```

13 Un último ejemplo

En realidad podemos utilizar el slicing como **secuencia[i:j:k]**

donde: - **i:** representa el límite inferior para comenzar, - **j:** la posición hasta donde queremos incluir elementos - **k:** cada cuántos valores queremos que nos muestre, o sea, el salto de un elemento al siguiente.

14 Analicemos este código

[56]:
```
como_recorro = "invertido"
def muestro_cadena(cadena, orden=como_recorro):
    """ Esta función retorna la cadena de acuerdo al parámetro orden """

    return cadena[::-1] if orden == "invertido" else cadena[:]
```

[57]:
```
muestro_cadena("Hola")
```

[57]:
```
'aloH'
```

[58]:
```
como_recorro = "normal"
```

¿Qué pasa ahora si vuelvo a invocar a la función? **Investigar qué es lo que sucede acá.**

[61]:
```
muestro_cadena("Hola")
```

[61]:
```
'aloH'
```