Previa_clase3

March 26, 2021

0.0.1 Seminario de Lenguajes - Python

0.1 Cursada 2021

0.1.1 Previa clase 3

1 Hagamos un repaso de la clase del martes

- En la clase estuvimos trabajando con este desafío: **necesitamos procesar las notas de los estudiantes de este curso**. Queremos saber:
  - cuál es el promedio de las notas,
  - qué estudiantes están por debajo del promedio.
- Y plantemos este pseudocódigo para la solución:

```
Ingresar las notas
Calcular el promedio
Mostar quiénes tienen notas menores al promedio
```

- Implementamos el primer proceso:

[27]:
```
def ingreso_notas():
    """ Esta función retorna un diccionario con los nombres y notas de␣
 ↪estudiantes """

    nombre = input("Ingresa un nombre (<FIN> para finalizar)")
    dicci = {}
    while nombre != "FIN":
        nota = int(input(f"Ingresa la nota de {nombre}"))
        dicci[nombre] = nota
        nombre = input("Ingresa un nombre (<FIN> para finalizar)")
    return dicci

notas_de_estudiantes = ingreso_notas()
notas_de_estudiantes
```

```
Ingresa un nombre (<FIN> para finalizar)Clau
Ingresa la nota de Clau5
Ingresa un nombre (<FIN> para finalizar)Vivi
Ingresa la nota de Vivi10
Ingresa un nombre (<FIN> para finalizar)Sofía
Ingresa la nota de Sofía9
```

```
Ingresa un nombre (<FIN> para finalizar)Fede
Ingresa la nota de Fede9
Ingresa un nombre (<FIN> para finalizar)FIN
```

[27]:
```
{'Clau': 5, 'Vivi': 10, 'Sofía': 9, 'Fede': 9}
```

2 Ahora observemos este código que implementa el segundo proceso:

[28]:
```
def calculo_promedio(notas):
    """ Esta función calcula el promedio de las notas recibida por parámetro.

    notas: es un diccionario de forma nombre_estudiante: nota
    """
    suma = 0
    for estu in notas:
        suma += notas[estu]
    promedio = 0 if len(notas)==0 else suma/len(notas)
    return promedio

calculo_promedio(notas_de_estudiantes)
```

[28]:
```
8.25
```

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

3 Parámetros en Python

- Veamos un ejemplo más sencillo:

[29]:
```
def modifico_parametro(x):
    x = 10

a = 2
modifico_parametro(a)
print(a)
```

```
2
```

3.0.1 Y ahora analicemos este otro ejemplo:

[30]:
```
def modifico_parametro1(x):
    x[0] = "cero"

lista = [1, 20]
modifico_parametro1(lista)
```

```
print(lista)
```

```
['cero', 20]
```

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

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

5 Analicemos de nuevo los ejemplos anteriores

[31]:
```
def modifico_parametro(x):
    x = 10

a = 2
modifico_parametro(a)
print(a)

def modifico_parametro1(x):
    x[0] = "cero"

lista = [1, 20]
modifico_parametro1(lista)
print(lista)
```

```
2
['cero', 20]
```

6 Ahora miremos este otro ejemplo

[61]:
```
def modifico_lista(x):
    y = x[:]
    y[0] = "cero"

lista = [1,20]
modifico_lista(lista)

print(lista)
```

```
[1, 20]
```

- ¿Qué pasa en este caso?
- Y si invoco con **modifico_lista(lista[:])** sería necesario la primer asignación? ¿Les parece buena esta práctica?

7 ¿Podemos retornar más de un valor?

7.0.1 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.

- ¿Qué tipo de dato retorna la función?

[34]:
```
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)
```

[34]:
```
tuple
```

8 ¿Cómo accedemos a los valores retornados?

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

[35]:
```
vocales_abiertas = retorno_varios("Seminario de Python")[0]
vocales_abiertas
```

[35]:
```
5
```

[37]:
```
abiertas, cerradas, longitud = retorno_varios("Este video va a ser corto")
cerradas
```

[37]:
```
1
```

9 Los parámetros en Python pueden tener valores por defecto

[44]:
```
def mi_musica(dicci_musica, nombre, tipo_musica="nacional"):
    """  .... """
    if nombre in dicci_musica:
        interpretes = dicci_musica[nombre]
        for elem in interpretes[tipo_musica]:
            print(elem)
    else:
        print(f"¡Hola {nombre}! No tenés registrada música en esta colección")
```

```
dicci_musica = {"clau": {"internacional": ["AC/DC", "Led Zeppelin", "Bruce␣
↪Springsteen"],
                         "nacional": ["Pappo", "Miguel Mateos", "Los Piojos",␣
↪"Nonpalidece"]
                         },
                "vivi": {"internacional": ["Ricky Martin", "Maluma", "Madona"],
                          "nacional": ["Lali"]}
                }
mi_musica(dicci_musica, "vivi", "internacional")
```

```
Ricky Martin
Maluma
Madona
```

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**.

[45]:
```
def mi_musica(dicci_musica, nombre, tipo_musica="nacional"):
    if nombre in dicci_musica:
        interpretes = dicci_musica[nombre]
        for elem in interpretes[tipo_musica]:
            print(elem)
    else:
        print(f"¡Hola {nombre}! No tenés registrada música en esta colección")


mi_musica(nombre="vivi", tipo_musica="internacional", dicci_musica=dicci_musica␣
↪)
```

```
Ricky Martin
Maluma
Madona
```

10 Obervemos con atención este código

[47]:
```
def agrego(a, L=[]):
    L.append(a)
    return L


print(agrego(1))
print(agrego(2))
print(agrego(3))
```

```
[1]
```

```
[1, 2]
[1, 2, 3]
```

10.0.1 IMPORTANTE: los valores por defecto se evalúan UNA ÚNICA VEZ en la definición de la función.

11 Veamos otro tipo predefinido de Python

12 Conjuntos en Python

- Un conjunto es una colección de datos heterogéna, **desordenada**, **NO indexada** y **sin elementos duplicados**.

[50]:
```
bandas = {"AC/DC", "Metallica", "Greta Van Fleet", "Soda Stéreo", "Los Piojos"}
print(bandas)
```

```
{'AC/DC', 'Greta Van Fleet', 'Los Piojos', 'Soda Stéreo', 'Metallica'}
```

[52]:
```
bandas_nacionales = set(("Soda Stéreo", "La Renga"))
bandas_nacionales
bandas_internacionales = set(["Greta Van Fleet", "Led Zeppelin"])
bandas_internacionales
```

[52]:
```
{'Greta Van Fleet', 'Led Zeppelin'}
```

[53]:
```
letras = set("alabanza")
letras
```

[53]:
```
{'a', 'b', 'l', 'n', 'z'}
```

13 No confundir

[54]:
```
dicci = {}
conjunto = set()

type(conjunto)
```

[54]:
```
set
```

[56]:
```
c1 = {'alabanza'}
c2 = set('alabanza')
c2
```

[56]:
```
{'a', 'b', 'l', 'n', 'z'}
```

14 ¿Cómo recorremos un conjunto?

[57]:
```
bandas = {"AC/DC", "Metallica", "Greta Van Fleet", "Soda Stéreo", "Los Piojos"}

for elem in bandas:
    print(elem)
```

```
AC/DC
Greta Van Fleet
Los Piojos
Soda Stéreo
Metallica
```

15 Y, ¿cómo sabemos la cantidad de elementos de un conjunto?

[59]:
```
len(bandas)
c2 = set('alabanzas')
len(c2)
```

[59]: 6

16 Operaciones con conjuntos

- Pensemos en las operaciones matemáticas sobre conjuntos:
  - **in**: retonar si un elemento pertenece o no a un conjunto.
  - |: unión entre dos conjuntos.
  - &: intersección entre dos conjuntos.
  - -: diferencia de conjuntos.

[60]:
```
bandas = {"AC/DC", "Metallica", "Greta Van Fleet", "Soda Stéreo", "Los Piojos"}
bandas_nacionales = set(("Soda Stéreo", "La Renga", "Los Piojos"))

print("Foo Fighters" in bandas)

todos = bandas | bandas_nacionales
print(todos)

en_ambos = bandas & bandas_nacionales
print(en_ambos)

solo_en_1 = bandas - bandas_nacionales
print(solo_en_1)
```

```
False
{'Metallica', 'La Renga', 'Los Piojos', 'Greta Van Fleet', 'Soda Stéreo',
'AC/DC'}
```

```
{'Los Piojos', 'Soda Stéreo'}
{'Metallica', 'AC/DC', 'Greta Van Fleet'}
```

17 También hay métodos que se pueden utilizar

- Para copiar conjuntos podemos usar la sentencia de asignación o usando **copy()**.
- Se puede adicionar un elemento con **add()**.

[ ]:
```python
bandas = {"AC/DC", "Metallica", "Greta Van Fleet", "Soda Stéreo", "Los Piojos"}
bandas_nacionales = set(("Soda Stéreo", "La Renga", "Los Piojos"))

#mis_bandas = bandas
mis_bandas = bandas.copy()
print(id(mis_bandas))
print(id(bandas))

bandas.add("Foo Fighters")
print(bandas)
```

18 Probar en casa otros métodos

- issubset(), isdisjoint(), issuperset(), update(), discard(), remove()
- + Info en Real Python
- + Info en w3schools
- + Info en el sitio oficial