

# Taller 3

Métodos Computacionales para Políticas Públicas - UROSARIO

**Entrega: viernes 21-feb-2020 11:59 PM**

**\*\* Juan Sebastián Gómez \*\***

juansebastian.gomez@urosario.edu.co

## Instrucciones:

- Guarde una copia de este *Jupyter Notebook* en su computador, idealmente en una carpeta destinada al material del curso.
- Modifique el nombre del archivo del *notebook*, agregando al final un guión inferior y su nombre y apellido, separados estos últimos por otro guión inferior. Por ejemplo, mi *notebook* se llamaría:  
mcpp\_taller3\_santiago\_matallana
- Marque el *notebook* con su nombre y e-mail en el bloque verde arriba. Reemplace el texto "[Su nombre acá]" con su nombre y apellido. Similar para su e-mail.
- Desarrolle la totalidad del taller sobre este *notebook*, insertando las celdas que sea necesario debajo de cada pregunta. Haga buen uso de las celdas para código y de las celdas tipo *markdown* según el caso.
- Recuerde salvar periódicamente sus avances.
- Cuando termine el taller:
  1. Descárguelo en PDF.
  2. Suba los dos archivos (.pdf y .ipynb) a su repositorio en GitHub antes de la fecha y hora límites.

(El valor de cada ejercicio está en corchetes [ ] después del número de ejercicio.)

Antes de iniciar, por favor descargue el archivo [2020-I\\_mcpp\\_taller\\_3\\_listas\\_ejemplos.py](#) del repositorio, guárdelo en la misma carpeta en la que está trabajando este taller y ejecútelo con el siguiente comando:

In [4]:

```
run 2020-I_mcpp_taller_3_listas_ejemplos.py
```

Este archivo contiene tres listas (10, 11 y 12) que usará para las tareas de esta sección. Puede ver los valores de las listas simplemente escribiendo sus nombres y ejecutándolos en el Notebook. Inténtelo para verificar que [2020-I\\_mcpp\\_taller\\_3\\_listas\\_ejemplos.py](#) quedó bien cargado. Debería ver:

In [1]: 10

Out[1]: []

In [2]: 11

Out[2]: [1, 'abc', 5.7, [1, 3, 5]]

In [3]: 12

Out[3]: [10, 11, 12, 13, 14, 15, 16]

In [5]:

10

Out[5]:

[]

In [6]:

11

Out[6]:

[1, 'abc', 5.7, [1, 3, 5]]

In [7]:

12

Out[7]:

[10, 11, 12, 13, 14, 15, 16]

## 1. [1]

Cree una lista que contenga los elementos 7, "xyz" y 2.7.

In [8]:

lista\_ex\_1 = [7,"xyz",2.7]

In [16]:

lista\_ex\_1

Out[16]:

[7, 'xyz', 2.7]

## 2. [1]

Halle la longitud de la lista l1.

In [9]:

```
len(l1)
```

Out[9]:

4

## 3. [1]

Escriba expresiones para obtener el valor 5.7 de la lista l1 y para obtener el valor 5 a partir del cuarto elemento de l1.

In [10]:

```
for i in range(0, len(l1)):
    if l1[i] == 5.7 :
        print(f"El valor se encuentra en la posición {i}")
```

El valor se encuentra en la posición 2

In [11]:

```
l1[2]
```

Out[11]:

5.7

In [12]:

```
l1[3][2]
```

Out[12]:

5

## 4. [1]

Prediga qué ocurrirá si se evalúa la expresión l1[4] y luego pruébelo.

Python va a buscar el elemento en la posición 4 dentro de la lista "l1", lo que va a causar que arroje error debido a que la lista está compuesta por 4 elementos (el último de ellos siendo otra lista), y al iniciar desde la posición 0, el último elemento se encontrará en la posición 3, por lo que no va a haber ningún elemento en la posición 4 y arrojará error.

In [13]:

```
l1[4]
```

```
-----  
-----  
IndexError                                Traceback (most recent call l  
ast)  
<ipython-input-13-7ffdc2c9f2e> in <module>  
----> 1 l1[4]  
  
IndexError: list index out of range
```

## 5. [1]

Prediga qué ocurrirá si se evalúa la expresión `l2[-1]` y luego pruébelo.

Python va a buscar el elemento que se encuentra en la última posición de la lista

In [14]:

```
l2[-1]
```

Out[14]:

```
16
```

## 6. [1]

Escriba una expresión para cambiar el valor 3 en el cuarto elemento de `l1` a 15.0.

In [15]:

```
l1[3][1] = 15.0
```

In [16]:

```
l1
```

Out[16]:

```
[1, 'abc', 5.7, [1, 15.0, 5]]
```

## 7. [1]

Escriba una expresión para crear un "slice" que contenga del segundo al quinto elemento (inclusive) de la lista `l2`.

In [17]:

```
slice1 = l2 [1:5]  
slice1
```

Out[17]:

```
[11, 12, 13, 14]
```

## 8. [1]

Escriba una expresión para crear un "slice" que contenga los primeros tres elementos de la lista l2.

In [18]:

```
slice2 = l2 [:3]  
slice2
```

Out[18]:

```
[10, 11, 12]
```

## 9. [1]

Escriba una expresión para crear un "slice" que contenga del segundo al último elemento de la lista l2.

In [19]:

```
slice3 = l2[1:]  
slice3
```

Out[19]:

```
[11, 12, 13, 14, 15, 16]
```

## 10. [1]

Escriba un código para añadir cuatro elementos a la lista l0 usando la operación append y luego extraiga el tercer elemento (quítelo de la lista). ¿Cuántos "appends" debe hacer?

In [20]:

```
import random
l0 = []
for i in range(0,4):
    a = random.randint(1,10)
    l0.append(a)
    print(i)
l0
```

0  
1  
2  
3

Out[20]:

[6, 2, 7, 3]

Para el punto anterior se usa un solo append, pero puede hacerse usando 4 append, en cada uno agregando un elemento nuevo a la lista

In [21]:

```
del l0[2] #<- elimino el tercer elemento que se encuentra ubicado en la posición 2
print(l0)
```

[6, 2, 3]

## 11. [1]

Cree una nueva lista n1 concatenando la nueva versión de l0 con l1, y luego actualice un elemento cualquiera de n1. ¿Cambia alguna de las listas l0 o l1 al ejecutar los anteriores comandos?

In [22]:

```
n1 = l0 + l1
print(n1)
```

[6, 2, 3, 1, 'abc', 5.7, [1, 15.0, 5]]

In [23]:

```
n1[2] = 85
print(n1) #<- Al imprimir la lista n1 se ve que el elemento en la posición 2 se modificó, de 6 a 85
```

[6, 2, 85, 1, 'abc', 5.7, [1, 15.0, 5]]

In [24]:

```
print(l0) #<- se ve que la lista l0 no cambia al modificar nl
```

[6, 2, 3]

In [25]:

```
print(l1) #<- se ve que la lista l1 no cambia al modificar nl
```

[1, 'abc', 5.7, [1, 15.0, 5]]

## 12. [2]

Escriba un loop que compute una variable all\_pos cuyo valor sea True si todos los elementos de la lista l3 son positivos y False en otro caso.

In [26]:

```
l3 = []
a = 20
for i in range (a):
    l3.append(random.randint(-10,10))
l3

for i in (l3):
    if i > 0:
        all_pos = True
    else:
        all_pos = False
        break
```

In [27]:

```
print(l3)
```

[-8, 7, 3, 8, 6, -9, -8, -7, -5, -9, -5, 8, 0, -2, -8, -2, -8, 6, 7, 0]

In [28]:

```
print(all_pos)
```

False

## 13. [2]

Escriba un código para crear una nueva lista que contenga solo los valores positivos de la lista l3.

In [29]:

```
l4=[] # <- Lista con solo valores positivos de la lista l3
for i in l3:
    if i > 0:
        l4.append(i)
```

In [30]:

```
print(l4)
```

```
[7, 3, 8, 6, 8, 6, 7]
```

## 14. [2]

Escriba un código que use `append` para crear una nueva lista `nl` en la que el *i*-ésimo elemento de `nl` tiene el valor `True` si el *i*-ésimo elemento de `l3` tiene un valor positivo y `Falso` en otro caso.

In [31]:

```
nl2 = []
for i in l3:
    if i > 0:
        nl2.append(True)
    else:
        nl2.append(False)
```

In [32]:

```
print(nl2)
```

```
[False, True, True, True, True, False, False, False, False, False, False, False, True, True, False]
```

## 15. [3]

Escriba un código que use `range`, para crear una nueva lista `nl` en la que el *i*-ésimo elemento de `nl` es `True` si el *i*-ésimo elemento de `l3` es positivo y `False` en otro caso.

**Pista:** Comience por crear una lista de longitud adecuada, con `False` en cada elemento.

In [33]:

```
nl3 = [False] * len(l3)
for i in range(len(l3)):
    if l3[i] > 0:
        nl3[i] = True
```



In [34]:

```
print (nl3)
print(l3)
```

```
[False, True, True, True, True, False, False, False, False, False, False, False, True, False, False, False, False, False, True, True, False]
[-8, 7, 3, 8, 6, -9, -8, -7, -5, -9, -5, 8, 0, -2, -8, -2, -8, 6, 7, 0]
```

In [35]:

```
print(l3)
```

```
[-8, 7, 3, 8, 6, -9, -8, -7, -5, -9, -5, 8, 0, -2, -8, -2, -8, 6, 7, 0]
```

## 16. [4]

En clase construimos una lista con 10000 números aleatorios entre 0 y 9, a partir del siguiente código:

```
import random
N = 10000
random_numbers = []
for i in range(N):
    random_numbers.append(random.randint(0,9))
```

Y creamos un "contador" que calcula la frecuencia de ocurrencia de cada número del 0 al 9, así:

```
count = []
for x in range(0,10):
    count.append(random_numbers.count(x))
```

Cree un "contador" que haga lo mismo, pero sin hacer uso del método "count". (De hecho, sin usar método alguno.)

### Pistas:

- Esto puede lograrse con un loop muy sencillo. Si su código es complejo, piense el problema de nuevo.
- Es muy útil iniciar con una lista "vacía" de 10 elementos. Es decir, una lista con 10 ceros.

In [36]:

```
import random
N = 10
random_numbers = []
for i in range(N):
    random_numbers.append(random.randint(0,9))
```

In [37]:

```
random_numbers
```

Out[37]:

```
[8, 8, 5, 7, 4, 7, 1, 8, 8, 6]
```

In [40]:

```
my_count = []
for i in range (N):
    frec = 0
    for a in random_numbers:
        if a == i:
            frec = frec + 1
    my_count.append((i,frec)) #<- hago que mi lista muestre el número y la frecuencia...
                                #<-...de ocurrencia de cada número. ej: el 0 aparece
                                #<-... 0 veces, por lo que (0,0)
```

In [41]:

```
print(my_count)
```

```
[(0, 0), (1, 1), (2, 0), (3, 0), (4, 1), (5, 1), (6, 1), (7, 2), (8, 4), (9, 0)]
```