Taller 3

Métodos Computacionales para Políticas Públicas - URosario

Entrega: viernes 4-sep-2020 11:59 PM

[Ivonne Paola Ubaque Galán]

[ivonne.ubaque@urosario.edu.co (mailto:ivonne.ubaque@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 descarge el archivo `2020-II_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:

```
run 2020-II_mcpp_taller_3_listas_ejemplos.py
```

Este archivo contiene tres listas (`l0`, `l1` y `l2`) 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-II_mcpp_taller_3_listas_ejemplos.py` quedó bien cargado. Debería ver:

```
In [1]: l0
Out[1]: []


In [2]: l1
Out[2]: [1, 'abc', 5.7, [1, 3, 5]]


In [3]: l2
Out[3]: [10, 11, 12, 13, 14, 15, 16]
```

1. [1]

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

```
In [1]: run 2020-II_mcpp_taller_3_listas_ejemplos.py

In [2]: l20 = [7, "xyz", 2.7]

In [3]: l20
Out[3]: [7, 'xyz', 2.7]
```

2. [1]

Halle la longitud de la lista `l1`.

```
In [4]: len(l20)
Out[4]: 3
```

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 [5]: l1[2]
Out[5]: 5.7

In [6]: l1[-2]
Out[6]: 5.7

In [8]: l1[4:6]
Out[8]: []
```

4. [1]

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

In [9]:
```
#Va a aparecer un error en el entendido que la lista está vacía, no contiene un e

l1[4]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-9-601b40e16395> in <module>
      1 #Va a aprecer un error en el entendido que la lista está vacía, no cont
iene un elemento ni en la posición 4 u algún otro
      2
----> 3 l1[4]

IndexError: list index out of range
```

5. [1]

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

In [10]:
```
#Va a aparecer el componente ubicado en el primer puesto de izquierda a derecha d
l2[-1]
```

Out[10]:
```
16
```

6. [1]

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

In [11]:
```
l1
```

Out[11]:
```
[1, 'abc', 5.7, [1, 3, 5]]
```

In [12]:
```
#si somos puristas del lenguaje de python, la lista l1 no tiene cuarto elemento,
l1[3] = [1 , 3, 15.0]
```

In [13]:
```
l1
```

Out[13]:
```
[1, 'abc', 5.7, [1, 3, 15.0]]
```

7. [1]

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

In [14]:
```
l2
```

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

In [15]:
```
l2[1:6]
```

Out[15]:
```
[11, 12, 13, 14, 15]
```

8. [1]

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

In [16]:
```
l2[0:4]
```

Out[16]:
```
[10, 11, 12, 13]
```

9. [1]

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

In [17]:
```
l2[1:7]
```

Out[17]:
```
[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 [18]:
```
l0.append("aa")
l0.append("17.6")
l0.append("146")
l0.append("asdf")
l0
```

Out[18]:
```
['aa', '17.6', '146', 'asdf']
```

In [19]:
```
del l0[2]
l0
```

Out[19]:
```
['aa', '17.6', 'asdf']
```

Respuesta: Fue necesario efectuar 4 "append".

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 [20]:
```
n1 = l0 + l1
n1
```

Out[20]:
```
['aa', '17.6', 'asdf', 1, 'abc', 5.7, [1, 3, 15.0]]
```

In [21]:
```
n1.insert(2, "Sofía")
n1
```

Out[21]:
```
['aa', '17.6', 'Sofía', 'asdf', 1, 'abc', 5.7, [1, 3, 15.0]]
```

In [22]:
```
l0
```

Out[22]:
```
['aa', '17.6', 'asdf']
```

In [23]:
```
l1
```

Out[23]:
```
[1, 'abc', 5.7, [1, 3, 15.0]]
```

Respuesta: no cambian las listas originales l0 y l1, en el entendido que hay una tercera nueva lista n1 que es a la cual se le está insertando el nuevo componente string "Sofía"

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 [33]:
```
l3 = range(-2,6)
```

In [34]:
```
list(l3)
```

Out[34]:
```
[-2, -1, 0, 1, 2, 3, 4, 5]
```

In [35]:
```
for x in l3:
    if x>0:
        print(True)
    else:
        print(False)
```

```
False
False
False
True
True
True
True
True
```

13. [2]

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

In [37]:
```
```

Out[37]:
```
[]
```

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.

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.

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.