Taller 3

Métodos Computacionales para Políticas Públicas - URosario

Entrega: viernes 26-feb-2021 11:59 PM

"[Juan Diego Castro Rodríguez]"

[juand.castro@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. Descarguelo 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 descargar el archivo 2021-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 2021-I_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 2021-I_mcpp_taller_3_listas_ejemplos.py quedó bien cargado. Debería ver:

```
In [18]: l1
Out[18]: [1, 'abc', 5.7, [1, 3, 5]]

In [19]: l2
Out[19]: [10, 11, 12, 13, 14, 15, 16]

In [20]: l0
Out[20]: []
```

```
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 [14]: lista1=[7,"xyz",2.7]

In [16]: lista1
Out[16]: [7, 'xyz', 2.7]
```

2. [1]

Halle la longitud de la lista `l1`.

```
In [17]: print(len(l1))
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 [22]: l1[2]
Out[22]: 5.7

In [32]: l1[3][2]
Out[32]: 5
```

4. [1]

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

el conteo de elementos al interior de una lista comienza desde el 0, luego en la l1 no existe de momento un [4], es decir un quinto elemento, razón por la cual ejecutar el código l1[4], nos arroja error

```
In [34]: l1[4]
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-34-7ffdcb2c0f2e> 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.

en el conteo de los elementos de una lista el -1 hace referencia al úlitmo elemento de la misma, luego al ejecutar el código l2[-1], se llamará al último elemento de la lista 2 el cual es el 16

```
In [35]: l2[-1]
Out[35]: 16
```

6. [1]

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

```
In [36]: l1
Out[36]: [1, 'abc', 5.7, [1, 3, 5]]

In [37]: l1[3][1]= 15.0

In [38]: l1
Out[38]: [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 [39]: l2
Out[39]: [10, 11, 12, 13, 14, 15, 16]

In [41]: l2[1:5]
Out[41]: [11, 12, 13, 14]
```

8. [1]

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

```
In [42]: l2[0:3]
Out[42]: [10, 11, 12]
```

9. [1]

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

```
In [44]: l2[1:]
Out[44]: [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 (quitelo de la lista). *¿Cuántos "appends" debe hacer?*

```
In [45]: l0
Out[45]: []

In [47]: l0.append("uno")

In [49]: l0.append("dos")

In [51]: l0.append("tres")

In [52]: l0.append("cuatro")

In [53]: l0
Out[53]: ['uno', 'dos', 'tres', 'cuatro']

In [54]: del l0[2]

In [55]: l0
Out[55]: ['uno', 'dos', 'cuatro']
```

Utilice 4 appends

11. [1]

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

```
In [56]: ln1=l0+l1
         ln1
Out[56]: ['uno', 'dos', 'cuatro', 1, 'abc', 5.7, [1, 15.0, 5]]

In [57]: ln1[2]="cinco"
         ln1
Out[57]: ['uno', 'dos', 'cinco', 1, 'abc', 5.7, [1, 15.0, 5]]

In [58]: l0
Out[58]: ['uno', 'dos', 'cuatro']

In [60]: l1
Out[60]: [1, 'abc', 5.7, [1, 15.0, 5]]
```

No cambian ninguna de las lista l0 o l1 porque cree una lista utilizando los elementos de l0 y l1, luego estoy creando una copia independiente de las mismas y no cambiandole el nombre a una misma lista

12. [2]

Escriba un loop que compute una variable `all_pos` cuyo valor será `True` si todos los elementos de la lista `l3` son positivos y `False` en otro caso.

```
In [91]: l3 = [1,2,3,-1,-2]
         for x in l3:
             if all(x > 0 for x in l3):
                 all_pos="True"
             else:
                 all_pos="False"
         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 [77]: l3n=[x for x in l3 if x>0]
         l3n
Out[77]: [1, 2, 3]
```

14. [2]

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

```
In [96]: ln1=[]
         for x in l3:
             if x>0:
                 ln1.append("True")
             else:
                 ln1.append("Falso")
         print(ln1)
['True', 'True', 'True', 'Falso', 'Falso']
```

15. [3]

Escriba un código que use `range`, para crear una nueva lista `sl` en la que el *i*-ésimo elemento de `sl` 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 [107]: ln1=[]
          l3=[-5,-4,-3,-2,-1]
          for x  in range (len(l3)):
              if l3[x]>0:
                  ln1.append("True")
              else:
                  ln1.append("False")
          print(ln1)
['False', 'False', 'False', 'False', 'False']
```

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 [3]: N=10000
        import random
        random_numbers = []
        for i in range(N):
            random_numbers.append(random.randint(0,9))
        c = [0]*10
        for number in random_numbers:
            c[number] +=1
        print(c)
[979, 962, 1045, 922, 986, 1009, 1055, 1006, 1005, 1031]
```
