Taller 3

Métodos Computacionales para Políticas Públicas - URosario

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

[Nicolás Garcés R]

[nicolas.garces@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:

In [2]:

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

In [3]:

```
l1
```

Out[3]:

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

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 [16]:

```
lista = [7, "xyz", 2.7]
lista
```

Out[16]:

```
[7, 'xyz', 2.7]
```

2. [1]

Halle la longitud de la lista l1.

In [4]:

```
len(l1)
```

Out[4]:

```
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 [5]:

```
l1[2]
```

Out[5]:

```
5.7
```

In [29]:

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

Out[29]:

```
5
```

4. [1]

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

saldrá un error, pues la lista no tiene ningun valor definido para esa posicion

In [6]:

```
l1[4]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-6-7ffdcb2c9f2e> 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.

tendrá como resultado el numero 16, pues es el ultimo de la lista

In [7]:

```
l2[-1]
```

Out[7]:

```
16
```

6. [1]

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

In [4]:

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

Out[4]:

```
[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 [9]:

```
l2[1:5]
```

Out[9]:

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

8. [1]

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

In [10]:

```
l2[:3]
```

Out[10]:

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

9. [1]

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

In [11]:

```
l2[1:]
```

Out[11]:

```
[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?

se debe hacer un append por elemento añadido, por ende, se necesitan 4 append para añadir los 4 elementos

In [5]:

```
l0.append("a")
l0.append("b")
l0.append("c")
l0.append("d")
l0
del l0[2]
l0
```

Out[5]:

```
['a', 'b', 'd']
```

11. [1]

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

In [7]:

```
nl= l0 + l1
print(l1)
nl[3] = 3
print(l1)
```

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

No, cambiar no cambia la lista inicial.

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 [9]:

```
l3 = [8,27,-9,4,5,-10]
for i in l3:
    if i>0 and i+1 >0 and i+2 >0 and i+3 >0 and i+4 >0 and i+5 >0:
        all_pos= True
    else:
        all_pos= False

all_pos
```

Out[9]:

```
False
```

13. [2]

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

In [10]:

```
n_list = []

for i in l3:
    if i> 0:
        n_list.append(i)

n_list
```

Out[10]:

```
[8, 27, 4, 5]
```

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 [11]:

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

Out[11]:

```
[True, True, 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 [15]:

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

nl
```

Out[15]:

```
[True, True, False, True, True, 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.

- Es muy útil iniciar con una lista "vacía" de 10 elementos. Es decir, una lista con 10 ceros.

In [20]:

```
import random

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


count = [0] *10
for i in random_numbers:
    if i == 0:
        count[0]= count[0] + 1
    elif i == 1:
        count[1]= count[1] + 1
    elif i == 2:
        count[2]= count[2] + 1
    elif i == 3:
        count[3]= count[3] + 1
    elif i == 4:
        count[4]= count[4] + 1
    elif i == 5:
        count[5]= count[5] + 1
    elif i == 6:
        count[6]= count[6] + 1
    elif i == 7:
        count[7]= count[7] + 1
    elif i == 8:
        count[8]= count[8] + 1
    elif i == 9:
        count[9]= count[9] + 1
count
```

Out[20]:

```
[986, 991, 947, 984, 1014, 998, 1077, 972, 1014, 1017]
```