Taller 3

Métodos Computacionales para Políticas Públicas - URosario

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

Juan Sebastián Gómez

juansebastian.gomezm@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-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 [6]:

```
run 2020-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 2020-I_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]

In [10]:
```
l0
```
Out[10]:
```
[]
```

In [11]:
```
l1
```
Out[11]:
```
[1, 'abc', 5.7, [1, 3, 5]]
```

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

1. [1]

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

In [15]:
```
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 [18]:

```
len (l1)
```

Out[18]:

```
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 [34]:

```
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 [35]:

```
l1[2]
```

Out[35]:

```
5.7
```

In [36]:

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

Out[36]:

```
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 [37]:

```
l1[4]
```

```
---------------------------------------------------------------------
----
IndexError                                Traceback (most recent call l
ast)
<ipython-input-37-7ffdcb2c9f2e> 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.

In [ ]:

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

In [38]:

```
l2[-1]
```

Out[38]:

```
16
```

6. [1]

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

In [39]:

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

In [40]:

```
l1
```

Out[40]:

```
[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 [44]:

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

Out[44]:

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

8. [1]

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

In [47]:

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

Out[47]:

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

9. [1]

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

In [49]:

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

Out[49]:

```
[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 [61]:

```
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[61]:

```
[7, 10, 8, 6]
```

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 [64]:

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

Out[64]:

```
[7, 10, 6]
```

11. [1]

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

In [66]:

```
nl = l0 + l1
nl
```

Out[66]:

```
[7, 10, 6, 1, 'abc', 5.7, [1, 15.0, 5]]
```

In [69]:

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

Out[69]:

```
[7, 10, 85, 1, 'abc', 5.7, [1, 15.0, 5]]
```

In [70]:

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

Out[70]:

```
[7, 10, 6]
```

In [71]:

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

Out[71]:

```
[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 [97]:

```
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 [98]:

```
l3
```

Out[98]:

```
[1, -4, -1, -3, -3, -3, 6, 5, 1, 7, -10, -10, 0, -4, -10, -3, 6, -7, -
4, -2]
```

In [99]:

```
all_pos
```

Out[99]:

```
False
```

13. [2]

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

In [100]:

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

In [101]:

```
l4
```

Out[101]:

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

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 [102]:

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

In [103]:

```
nl2
```

Out[103]:

```
[True,
 False,
 False,
 False,
 False,
 False,
 True,
 True,
 True,
 True,
 False,
 False,
 False,
 False,
 False,
 False,
 True,
 False,
 False,
 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 [115]:

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

In [116]:

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

```
[True, False, False, False, False, False, True, True, True, True, Fals
e, False, False, False, False, False, True, False, False, False]
[1, -4, -1, -3, -3, -3, 6, 5, 1, 7, -10, -10, 0, -4, -10, -3, 6, -7, -
4, -2]
```

In [ ]:

```
l3
```

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))

In [117]:

```
import random

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

In [118]:

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

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 [137]:

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

In [138]:

```
random_numbers
```

Out[138]:

```
[5, 2, 0, 9, 1, 2, 4, 2, 2, 7]
```

In [139]:

```
my_count = []
for i in range (N):
    frec = 0
    for a in random_numbers:
        if a == i:
            frec = frec + 1
    my_count.append(frec)
```

In [140]:

```
my_count
```

Out[140]:

```
[1, 1, 4, 0, 1, 1, 0, 1, 0, 1]
```