Taller 3

Métodos Computacionales para Políticas Públicas - URosario

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

[Mónica Alejandra Robayo González]

[monica.robayo@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.)

In [ ]:

```
```

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:

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 [3]:

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

In [199]:

```
l0
```

Out[199]:

```
[]
```

In [207]:

```
l1
```

Out[207]:

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

In [4]:

```
l2
```

Out[4]:

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

1. [1]

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

In [46]:

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

In [47]:

```
lista
```

Out[47]:

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

2. [1]

Halle la longitud de la lista `l1`.

In [16]:

```
len(l1)
```

Out[16]:

```
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]:

```
l1 [2]
```

Out[34]:

```
5.7
```

In [41]:

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

Out[41]:

```
5
```

4. [1]

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

In [ ]:

```
#Respuesta: da error, ya que el índice de lista se encontraría fuera de rango.
```

In [42]:

```
l1 [4]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last
t)
<ipython-input-42-ad16827f93d4> 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 [ ]:

```
#Respuesta: toma la último posicion de la lista, es decir 16.
```

In [43]:

```
l2 [-1]
```

Out[43]:

```
16
```

6. [1]

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

In [208]:

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

In [209]:

```
l1
```

Out[209]:

```
[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 [50]:

```
l2
```

Out[50]:

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

In [52]:

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

Out[52]:

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

8. [1]

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

In [55]:

```
l2
```

Out[55]:

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

In [53]:

```
l2 [:3]
```

Out[53]:

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

9. [1]

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

In [56]:

```
l2
```

Out[56]:

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

In [54]:

```
l2 [1:7]
```

Out[54]:

```
[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 [7]:

```
l0
```

Out[7]:

```
[]
```

In [10]:

```
v = [1, 2, 3, 4]
for i in (v):
    v = i
    l0.append(v)
```

In [202]:

```
l0
```

Out[202]:

```
[1, 2, 3, 4]
```

In [203]:

```
del l0 [2]
```

In [204]:

```
l0
```

Out[204]:

```
[1, 2, 4]
```

In [ ]:

```
#Respuesta:El codigo debe hacer 4 appends, agrega 4 elementos.
```

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 [205]:

```
l0
```

Out[205]:

```
[1, 2, 4]
```

In [210]:

```
l1
```

Out[210]:

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

In [211]:

```
nl = l0 + l1
```

In [212]:

```
nl
```

Out[212]:

```
[1, 2, 4, 1, 'abc', 5.7, [1, 15.0, 5]]
```

In [ ]:

```
# No hay ningun cambio en las listas.
```

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 [297]:

```
l3 = [-10,-5,5,10,15]
```

In [298]:

```
l3
```

Out[298]:

```
[-10, -5, 5, 10, 15]
```

In [299]:

```
l3 = list(range(5,16,5))
```

In [300]:

```
l3
```

Out[300]:

```
[5, 10, 15]
```

In [301]:

```
type(True)
```

Out[301]:

```
bool
```

In [302]:

```
all_pos = True

for i in (l3):
    if i < 0:
        all_pos = False
print(all_pos)
```

```
True
```

In [303]:

```
all_pos = True

for i in [-10,-5,5,10,15]:
    if i < 0:
        all_pos = False
print(all_pos)
```

```
False
```

In [328]:

```
#otra forma de hacerlo:
for i in range(-10,16,5):
    if i < 0:
        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 [305]:

```
l3 = [-10,-5,5,10,15]
```

In [306]:

```
l3
```

Out[306]:

```
[-10, -5, 5, 10, 15]
```

In [307]:

```
l4 = []
for i in l3:
    if i >= 0:
        l4.append(i)
```

In [308]:

```
l4
```

Out[308]:

```
[5, 10, 15]
```

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 [310]:

```
nl = []
```

In [334]:

```
l3
```

Out[334]:

```
[-10, -5, 5, 10, 15]
```

In [377]:

```
for i in l3:
    if i >=0:
        nl.append("true")
    else:
        nl.append("false")
```

In [378]:

```
nl
```

Out[378]:

```
['false', 'false', 'true', 'true', 'true']
```

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 [427]:

```
nl =[]
```

In [428]:

```
nl
```

Out[428]:

```
[]
```

In [416]:

```
l3 = [-10,-5,5,10,15]
```

In [417]:

```
l3
```

Out[417]:

```
[-10, -5, 5, 10, 15]
```

In [440]:

```
for i in range(len(l3)):
    if l3[i] >= 0:
        nl[i]= "true"
    else:
        nl[i]= "false"
```

In [441]:

```
nl
```

Out[441]:

```
['false', 'false', 'true', 'true', 'true']
```

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 [1]:

```
import random

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

In [7]:

```
frecuencia = [0]*10

for i in list(range(len(random_numbers))):
    for j in list(range(10)):
        if random_numbers[i]==j:
            frecuencia[j]= frecuencia[j]+ 1
```

In [8]:

```
print(list(range(10)))
print (frecuencia)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[982, 1020, 996, 1022, 1012, 998, 989, 1026, 999, 956]
```