Taller 3

Métodos Computacionales para Políticas Públicas - URosario

Entrega: viernes 24-feb-2017 11:59 PM

[David Valles]

[david.valles@gmail.com]

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 `2017_1_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 2017_1_mcpp_taller_3_listas_ejemplos.py

```
In [3]: run 2017_1_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 `2017_1_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 [4]: l0
Out[4]: []
```

```
In [16]: l1
Out[16]: [1, 'abc', 5.7, [1, 3, 5]]
```

```
In [17]:  l2
Out[17]: [10, 11, 12, 13, 14, 15, 16]
```

1. [1]

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

```
In [10]: list1 = [7, "xyz", 2.7]
         print (list1)
```

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

2. [1]

Halle la longitud de la lista `l1`.

```
In [12]: len (l1)
```

```
Out[12]: 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 [24]: l1[2]
```

```
Out[24]: 5.7
```

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

```
Out[5]: 5
```

4. [1]

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

Saldrá un error dado que la lista l1 tiene solo 4 elementos pero python esta en base 0 por lo cual el cuarto elemento sería l1[3]

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

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-6-2c2a81dbcaa5> 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.

Saldrá el últimp elemento de la lista l2

```
In [29]: l2[-1]
```

```
Out[29]: 16
```

6. [1]

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

```
In [12]: print(l1)
         l1[3][1]=15.0
         print (l1)
```

```
[1, 'abc', 5.7, [1, 3, 5]]
[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 [14]: l2[1:5]
```

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

8. [1]

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

```
In [16]: l2[:3]
```

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

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?

i = 0 N = 4 while i < N: l0.append("1") i = i + 1 print(l0)

```
In [118]: l0.append("a")
          l0.append("b")
          l0.append("c")
          l0.append("d")
          print (l0)
          l0.pop(2)
          print(l0)
```

```
['a', 'b', 'c', 'd']
['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 [121]: n1= l0 + l1
          n1
```

```
Out[121]: ['a', 'b', 'd', 1, 'abc', 5.7, [1, 15.0, 5]]
```

```
In [122]: n1[0]="z"
          n1
```

```
Out[122]: ['z', 'b', 'd', 1, 'abc', 5.7, [1, 15.0, 5]]
```

```
In [126]: print(l0)
          print(l1)
```

```
['a', 'b', 'd']
[1, 'abc', 5.7, [1, 15.0, 5]]
```

No cambian las listas l0 y l1 por que la edición se presenta en la nueva lista n1

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 [146]: all_pos = True
          l3=[1,2,3,4,-5,6,-7]
          for i in l3:
              if i < 0:
                  all_pos = False
          print ("all_pos=", all_pos)
```

```
all_pos= False
```

13. [2]

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

In [150]:
```
l3_positivo=[]
print(l3_positivo)
for i in l3:
    if i>0:
        l3_positivo.append(i)
print(l3_positivo)
```

```
[]
[1, 2, 3, 4, 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 [184]:
```
nl=[]
l3=[1,2,3,4,-5,6,-7]
for i in l3:
    if i>0:
        nl.append(True)
    else:
        nl.append(False)
print(nl)
print(l3)
```

```
[True, True, True, True, False, True, False]
[1, 2, 3, 4, -5, 6, -7]
```

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 [248]:
```
nl=list(range(1,8))
for i in l3:
    if i>=0:
        nl.remove(i)
        nl.append(True)
    else:
        nl.pop(i)
        nl.append(False)
```

In [252]:
```
print(l3)
print(nl)
```

```
[1, 2, 3, 4, -5, 6, -7]
[True, True, True, True, False, 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.

In [337]:
```
import random

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

Out[337]: [0, 9, 8, 8, 2, 3, 0, 7, 7, 0]

In [338]:
```
count = []
for x in range(0,10):
    count.append(random_numbers.count(x))
count
```

Out[338]: [3, 0, 1, 1, 0, 0, 0, 2, 2, 1]

In [344]:
```
conteo=[]
for i in range(0,10):
    if i == i in random_numbers:
        conteo.append(i+1)
conteo
```

Out[344]: [1, 3, 4, 8, 9, 10]