Taller 3

Métodos Computacionales para Políticas Públicas - URosario

Entrega: viernes 22-feb-2019 11:59 PM

César Mateo Lancheros Cañón

cesar.lancheros@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 2019_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:

In [1]:

```
run 2019_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 2019_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 [2]:

```
print(l0)
print(l1)
print(l2)
```

```
[]
[1, 'abc', 5.7, [1, 3, 5]]
[10, 11, 12, 13, 14, 15, 16]
```

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

1. [1]

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

In [3]:

```
elementos = [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 [6]:

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

Out[6]:

5

4. [1]

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

Al ejecutar la expresión l1[4] aparecerá IndexError, porque el índice de la lista esta fuera del rango.

In [7]:

```
l1[4]
```

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

Prediga qué ocurrirá si se evalúa la expresión l2[-1] y luego pruébelo.

Al ejecutar la expresión l2[-1] aparecerá el último índice de la lista, el elemento 16, porque Python tiene indexación en base cero.

In [12]:

```
l2[-1]
```

Out[12]:

```
16
```

6. [1]

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

In [13]:

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

In [14]:

```
l1
```

Out[14]:

```
[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 [15]:

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

Out[15]:

```
[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:]
```

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?

Se deben hacer cuatro append, uno por cada elemento, porque esta opción unicamente acepta un elemento. Sin embargo, sí se desea que los elementos añadidos sean los mismos se puede ejecutar tantas veces como se desee el código y así, ese número de ejecuciones son el mismo número de veces que el elemento es añadido a la lista.

In [18]:

```
l0.append(15.4)
```

In [19]:

```
l0.append("metodos_computacionales")
```

In [20]:

```
l0.append("python")
```

In [21]:

```
l0.append(68)
```

In [22]:

```
l0
```

Out[22]:

```
[15.4, 'metodos_computacionales', 'python', 68]
```

In [23]:

```
l0.remove("python")
```

In [24]:

```
print(l0)
```

```
[15.4, 'metodos_computacionales', 68]
```

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?

Al realizar los cambios correspondientes, primero la concatenación y posterior la modificación, no se realiza ningun cambio en las listar originales.

In [25]:

```
n1 = l0 + l1
```

In [26]:

```
n1
```

Out[26]:

```
[15.4, 'metodos_computacionales', 68, 1, 'abc', 5.7, [1, 15.0, 5]]
```

In [27]:

```
n1[6] = "xyz"
```

In [28]:

```
print (n1)
print (l0)
print (l1)
```

```
[15.4, 'metodos_computacionales', 68, 1, 'abc', 5.7, 'xyz']
[15.4, 'metodos_computacionales', 68]
[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 [29]:

```
import random
l3 = []
N = 40
for i in range (N):
    l3.append(random.randint(-20, 20))
```

In [30]:

```
for i in l3:
    if i > 0:
        all_pos = True
    else:
        all_pos = False
        break
```

In [31]:

```
print (all_pos)
```

```
False
```

In [32]:

```
l3
```

Out[32]:

```
[-13,
 -18,
 -2,
 10,
 -20,
 15,
 -11,
 -19,
 7,
 16,
 -16,
 -16,
 -6,
 11,
 0,
 1,
 16,
 -10
```

```
-10,
-17,
-19,
-13,
8,
15,
-5,
-12,
-8,
-13,
-17,
-2,
0,
13,
10,
0,
12,
-7,
-18,
2,
-8,
4,
-8]
```

In [33]:

```
len(l3)
```

Out[33]:

```
40
```

13. [2]

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

In [34]:

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

In [35]:

```
new_list
```

Out[35]:

```
[10, 15, 7, 16, 11, 1, 16, 8, 15, 13, 10, 12, 2, 4]
```

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 [36]:

```
n2 = []
for i in l3:
    if i > 0:
        n2.append(True)
    elif i <= 0:
        n2.append(False)
```

In [37]:

```
n2
```

Out[37]:

```
[False,
 False,
 False,
 True,
 False,
 True,
 False,
 False,
 True,
 True,
 False,
 False,
 False,
 True,
 False,
 True,
 True,
 False,
 False,
 False,
 False,
 True,
 True,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 True,
 True,
 False,
 True,
 False,
 False,
 True,
 False,
 True,
 False]
```

In [38]:

```
len(n2)
```

Out[38]:

```
40
```

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 [39]:

```
n3 = []
for i in range(len(l3)):
    if l3[i] > 0:
        n3.append(True)
    elif l3[i] <= 0:
        n3.append(False)
```

In [44]:

```
len(n3)
n3
```

Out[44]:

```
[False,
 False,
 False,
 True,
 False,
 True,
 False,
 False,
 True,
 True,
 False,
 False,
 False,
 True,
 False,
 True,
 True,
 False,
 False,
 False,
 False,
 True,
 True,
 False,
 False,
 False,
 False,
 False,
 False,
 True,
 True,
 False,
 True,
 False,
 False,
 True,
 False,
 True,
 False]
```

In [48]:

```
n3 == n2
```

Out[48]:

```
True
```

In [46]:

```
n4 = [False]*40
for i in range(len(l3)):
    if l3[i] > 0:
        n4[i] = True
```

In [47]:

```
n4
```

Out[47]:

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

```
 True,
 True,
 False,
 False,
 False,
 True,
 False,
 True,
 True,
 False,
 False,
 False,
 False,
 True,
 True,
 False,
 False,
 False,
 False,
 False,
 False,
 False,
 True,
 True,
 False,
 True,
 False,
 False,
 True,
 False,
 True,
 False]
```

In [49]:

```
n4 == n2
```

Out[49]:

```
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 [50]:

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

In [51]:

```
random_numbers
```

Out[51]:

Out[51]:

```
[3, 2, 9, 7, 5, 4, 5, 9, 8, 3]
```

In [52]:

```
contador = []
for i in range (10):
    frecuencia = 0
    for x in random_numbers:
        if x == i:
            frecuencia = frecuencia + 1
    contador.append(frecuencia)
```

In [53]:

```
contador
```

Out[53]:

```
[0, 0, 1, 2, 1, 2, 0, 1, 1, 2]
```