Taller 3

Métodos Computacionales para Políticas Públicas - URosario

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

[Andrés Ramírez Vela]

[andrese.ramirez@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:

```
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 [ ]: run 2020-I_mcpp_taller_3_listas_ejemplos.py
```

```
In [2]: l0
Out[2]: []
```

```
In [3]: l1
Out[3]: [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 [5]: lista = [7, "sxz", 2.7]
```

2. [1]

Halle la longitud de la lista l1.

```
In [6]: len(lista)
Out[6]: 3
```

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 [9]:
```
[l1 [2] , l1 [3][2]]
```

Out[9]:
```
[5.7, 5]
```

4. [1]

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

El código presentará un error por estar fuera de rango puesto que la posición 4 no existe en esta lista.

In [10]:
```
l1 [4]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-10-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.

Muestra el último elemento de esta lista que sería 16

In [11]:
```
l2[-1]
```

Out[11]:
```
16
```

6. [1]

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

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

```
[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 [13]:
```
l2[1:5]
```

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

8. [1]

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

In [14]:
```
l2[:3]
```

Out[14]:
```
[10, 11, 12]
```

9. [1]

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

In [15]:
```
l2[1:]
```

Out[15]:
```
[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 [16]:
```
#Se deben realizar 4 appends
l0.append (l2[0])
l0.append (l2[1])
l0.append (l2[2])
l0.append (l2[3])
```

In [17]:
```
l0
```

Out[17]:
```
[10, 11, 12, 13]
```

In [18]:
```
del l0[2]
```

In [19]:
```
l0
```

Out[19]:
```
[10, 11, 13]
```

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 [20]:
```
nl = l0 + l1
nl [4] = "def"
```

In [21]:
```
nl
```

Out[21]:
```
[10, 11, 13, 1, 'def', 5.7, [1, 15.0, 5]]
```

In [22]:
```
#Al volver a mostrar las listas l0 y l1, ninguna de las dos ha cambiado
l0
```

Out[22]:
```
[10, 11, 13]
```

In [23]:
```
l1
```

Out[23]:
```
[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 [30]:
```
l3 = [-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5]
pos_all = True
for i in l3:
    if (i < 0):
        pos_all = False
print(pos_all)
```

```
False
```

In [58]:
```
#Sin embago, no estaba seguro sobre la interpretación del punto, razón p
or la cual decido correr el siguiente código
l3 = [-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5]
for i in l3:
    if (i < 0):
        print (pos_all == False)
    else:
        print (pos_all == True)
```

```
False
False
False
False
False
False
True
True
True
True
True
```

13. [2]

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

In [50]:
```
for i in l3:
    if (i >= 0):
        l4 = l3[i:]
print(l4)
```

```
[0, 1, 2, 3, 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 [49]:
```
for i in l3:
    if i >= 0:
        nl2.append(True)
    else:
        nl2.append (False)

print(nl2)
```

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

In [48]:
```
nl2 = []
```

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 [51]:
```
n15 = []
for i in l3:
    n15.append(False)

# Crea una variable con la longitud de la lista
n15_length = len(n15)

# llena la lista dependiendo de la condicion
for n in range(0, n15_length):
    if (l3[n])>=0: n15[n]=True
    else: n15[n]=False
print(n15)
```

```
[False, False, False, False, False, True, True, True, 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 [52]:
```
import random

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

counter = [0,0,0,0,0,0,0,0,0,0]

for i in random_numbers:
    for n in range(0,10):
        if i==n:
            counter[i]=counter[i]+1
print(counter)
```

```
[1000, 954, 983, 981, 1017, 985, 1061, 1027, 1014, 978]
```