mcpp_taller3_john_caro

August 26, 2016

1 Taller 3

Métodos Computacionales para Políticas Públicas - URosario

Entrega: viernes 26-ago-2016 11:59 PM

[John Alexander caro Becerra] [Jhonalexbc@gmail.com]

1.1 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 mcpp_taller3_listas_ejemplos.py del repositorio, guárdelo en la misma carpeta en la que está trabajando este taller y ejecútelo con el siguiente comando:

```
In [12]: run mcpp_taller3_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 mcpp_taller3_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 [13]: l0

Out[13]: []

In [14]: l1

Out[14]: [1, 'abc', 5.7, [1, 3, 5]]

In [15]: l2

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

1.2 1. [1]

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

```
In [16]: lista1 = [7, "xyz", 2.7]
         lista1

Out[16]: [7, 'xyz', 2.7]
```

1.3 2. [1]

Halle la longitud de la lista l1.

```
In [17]: len(lista1)

Out[17]: 3
```

1.4 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 [18]: l1

Out[18]: [1, 'abc', 5.7, [1, 3, 5]]

In [19]: l1[2]

Out[19]: 5.7

In [20]: #int(l1[2] -0.7) solucion sin corregir el punto

In [21]: l1[3][2]

Out[21]: 5
```

1.5 4. [1]

Prediga qué ocurrirá si se evalúa la expresión l1[4] y luego pruébelo.
Sale Error ya que l1[4] se refiere que muestre el elemento 5, pero l1 solo tiene 4 elementos

```
In [22]: l1[4]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-22-2c2a81dbcaa5> in <module>()
----> 1 l1[4]

IndexError: list index out of range
```

1.6 5. [1]

Prediga qué ocurrirá si se evalúa la expresión l2[-1] y luego pruébelo.
Lo que va salir es el ultimo elemento de la lista l2

```
In [23]: l2[-1]
```

```
Out[23]: 16
```

1.7 6. [1]

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

```
In [24]: l1
```

```
Out[24]: [1, 'abc', 5.7, [1, 3, 5]]
```

```
In [25]: l1[3][1]=15.0
```

```
In [26]: l1
```

```
Out[26]: [1, 'abc', 5.7, [1, 15.0, 5]]
```

1.8 7. [1]

Escriba una expresión para crear un "slice" que contenga del segundo al quinto elemento (inclusive) de la lista l2.

```
In [27]: l2
```

```
Out[27]: [10, 11, 12, 13, 14, 15, 16]
```

```
In [28]: slices = l2[1:5]

In [29]: slices

Out[29]: [11, 12, 13, 14]

In [30]: l2

Out[30]: [10, 11, 12, 13, 14, 15, 16]
```

1.9 8. [1]

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

```
In [31]: l2

Out[31]: [10, 11, 12, 13, 14, 15, 16]

In [32]: slices2 = l2[0:3]

In [33]: slices2

Out[33]: [10, 11, 12]

In [34]: l2

Out[34]: [10, 11, 12, 13, 14, 15, 16]
```

1.10 9. [1]

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

```
In [35]: l2

Out[35]: [10, 11, 12, 13, 14, 15, 16]

In [36]: slices3 = l2[2:len(l2)]

In [37]: slices3

Out[37]: [12, 13, 14, 15, 16]

In [38]: l2

Out[38]: [10, 11, 12, 13, 14, 15, 16]
```

1.11 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 [39]: l0

Out[39]: []

In [40]: for x in range(4):
             l0.append(x)
             print(x)

0
1
2
3


In [41]: l0

Out[41]: [0, 1, 2, 3]

In [42]: l0.pop(2)

Out[42]: 2

In [43]: l0

Out[43]: [0, 1, 3]
```

Debe hacerce 4 appends

1.12 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 [44]: l0

Out[44]: [0, 1, 3]

In [45]: l1

Out[45]: [1, 'abc', 5.7, [1, 15.0, 5]]

In [46]: nl= l0 + l1

In [47]: nl

Out[47]: [0, 1, 3, 1, 'abc', 5.7, [1, 15.0, 5]]
```

```
In [48]: nl[1]=5

In [49]: nl

Out[49]: [0, 5, 3, 1, 'abc', 5.7, [1, 15.0, 5]]

In [50]: l0

Out[50]: [0, 1, 3]

In [51]: l1

Out[51]: [1, 'abc', 5.7, [1, 15.0, 5]]
```

No cambia ninguno de los listas

1.13 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 [52]: l3=[1,-2,3,-4,5,-6,7,-8,9,-10]

In [53]: l3

Out[53]: [1, -2, 3, -4, 5, -6, 7, -8, 9, -10]

In [54]: for y in range(len(l3)):
             if l3[y] < 0:
                 print("False")
                 break
             elif y == (len(l3)-1) and l3[y] >0:
                 print("True")

False
```

1.14 13. [2]

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

```
In [55]: import random
         l4=[]
         for t in range(len(l3)):
             if l3[t] > 0:
                 l4.append(l3[t])

In [56]: l4

Out[56]: [1, 3, 5, 7, 9]
```

1.15 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 [57]: l3

Out[57]: [1, -2, 3, -4, 5, -6, 7, -8, 9, -10]

In [58]: nl2=[]
         for w in range(len(l3)):
             if l3[w] > 0:
                 nl2.append(True)
             else:
                 nl2.append(False)

In [59]: nl2

Out[59]: [True, False, True, False, True, False, True, False, True, False]
```

1.16 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 Falso en otro caso.

Pista: Comience por crear una lista de longitud adecuada, con False en cada índice.

```
In [60]: nl3=[]
         for w in range(len(l3)):
             if l3[w] > 0:
                 nl3.append(True)
             else:
                 nl3.append(False)

In [61]: nl3

Out[61]: [True, False, True, False, True, False, True, False, True, False]
```

1.17 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 [62]:
```
import random

N = 100
random_numbers = []
cero = 0
uno = 0
dos = 0
tres = 0
cuatro = 0
cinco = 0
seis = 0
siete = 0
ocho = 0
nueve = 0

for i in range(N):
    random_numbers.append(random.randint(0,9))
    if random_numbers[i] == 0:
        cero = cero + 1
    elif random_numbers[i] == 1:
        uno = uno + 1
    elif random_numbers[i] == 2:
        dos = dos + 1
    elif random_numbers[i] == 3:
        tres = tres + 1
    elif random_numbers[i] == 4:
        cuatro = cuatro + 1
    elif random_numbers[i] == 5:
        cinco = cinco + 1
    elif random_numbers[i] == 6:
        seis = seis + 1
    elif random_numbers[i] == 7:
        siete = siete + 1
    elif random_numbers[i] == 8:
        ocho = ocho + 1
    elif random_numbers[i] == 9:
        nueve = nueve + 1

frecuencia=[cero, uno, dos, tres, cuatro, cinco, seis, siete, ocho, nueve]
```

In [63]:
```
frecuencia
```

Out[63]:
```
[9, 14, 13, 9, 11, 10, 6, 5, 13, 10]
```