mcpp_taller3_Laura_Becerra

August 25, 2016

1 Taller 3

Métodos Computacionales para Políticas Públicas - URosario

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

[Laura Becerra] [l.becerra52@unaindes.edu.co]

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 [19]: 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 [20]: l1

Out[20]: [1, 'abc', 5.7, [1, 3, 5]]
```

1.2 1. [1]

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

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

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

1.3 2. [1]

Halle la longitud de la lista l1.

```
In [21]: len(l1)

Out[21]: 4
```

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 tercer elemento de l1.

```
In [22]: l1[2]

Out[22]: 5.7

In [23]: l1[3][2]

Out[23]: 5
```

1.5 4. [1]

Prediga qué ocurrirá si se evalúa la expresión l1[4] y luego pruébelo.
Arrojara error porque los indices van de 0 a 3

```
In [24]: l1[4]


---------------------------------------------------------------------------

IndexError                                Traceback (most recent call last)

<ipython-input-24-2c2a81dbcaa5> 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.
el valor que predice es 16

```
In [25]: l2[-1]

Out[25]: 16
```

1.7 6. [1]

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

```
In [26]: l1 [3] [1]= 15.0
         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[1:6]

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

1.9 8. [1]

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

```
In [28]: l2[0:3]

Out[28]: [10, 11, 12]
```

1.10 9. [1]

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

```
In [29]: l2[1:len(l2)]

Out[29]: [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 [30]: l0=[]
         l0.append(20)
         l0.append(21)
         l0.append(22)
         l0.append(23)
         l0
         l0.remove(22)
         l0
         #tuve que usar 4 appends

Out[30]: [20, 21, 23]
```

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 [31]: n1=l0+l1
         n1

Out[31]: [20, 21, 23, 1, 'abc', 5.7, [1, 15.0, 5]]

In [32]: n1[2]=50
         n1

Out[32]: [20, 21, 50, 1, 'abc', 5.7, [1, 15.0, 5]]

In [33]: l0

Out[33]: [20, 21, 23]

In [34]: l1

Out[34]: [1, 'abc', 5.7, [1, 15.0, 5]]
```

no cambian ni l0 ni l1 usando los comandos anteriores

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 [39]: l3= [50,51,52,-50,-51,-52,-53]
         for i in l3:
             if i >= 0:
                 all_pos = True
```

```
            print (i, all_pos)
        else:
            all_pos = False
            print (i, all_pos)
```

```
50 True
51 True
52 True
-50 False
-51 False
-52 False
-53 False
```

1.14 13. [2]

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

```
In [41]: nueva= []
         for i in l3:
             if i >= 0:
                 nueva.append(i)
         nueva
```

```
Out[41]: [50, 51, 52]
```

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 [52]: n1 = []
         for item in l3:
             if item >= 0:
                 all_pos = True
                 n1.append(all_pos)
             if item < 0:
                 all_pos = False
                 n1.append(all_pos)
         n1
```

```
Out[52]: [True, True, True, False, False, False, 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 [79]: n1= []
         for i in range(len(l3)):
             if l3[i] >= 0:
                 all_pos = True
                 n1.append(all_pos)

             if l3[i] < 0:
                 all_pos = False
                 n1.append(all_pos)
         n1
```

```
Out[79]: [True, True, True, False, False, False, False]
```

1.17 16. [4]

En clase construimos una lista con 10000 números aleatorios entre 0 y 9, a partir del siguiente código:

Y creamos un "contador" que calcula la frecuencia de ocurrencia de cada número del 0 al 9, así:

```
In [85]: import random
         N= 10000
         random_numbers=[]
         for i in range (N):
             random_numbers.append(random.randint(0,9))
```

```
In [86]: count = []
         for x in range(0,10):
             count.append(random_numbers.count(x))
         count
```

```
Out[86]: [1036, 987, 1013, 1045, 968, 1006, 1008, 958, 1038, 941]
```

```
In [88]: conteo=[0]*(10)
         for i in random_numbers:
             for n in range(10):
                 if i==n:
                     conteo[n]= conteo[n] + 1
         conteo
```

```
Out[88]: [1036, 987, 1013, 1045, 968, 1006, 1008, 958, 1038, 941]
```

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.
