mcpp_taller3_julian_ramirez

August 23, 2019

1 Taller 3

Métodos Computacionales para Políticas Públicas - URosario
Entrega: viernes 23-ago-2019 11:59 PM
Julián Santiago Ramírez julians.ramirez@urosario.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 2019_2_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:

2 run 2019_2_mcpp_taller_3_listas_ejemplos.py

```
In [13]: run 2019_2_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_2_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]: print(l0)
        print(l1)
        print(l2)

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

2.1 1. [1]

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

```
In [5]: primera_lista=[7,"xyz",2.7]
```

2.2 2. [1]

Halle la longitud de la lista l1.

```
In [8]: ## Longitud de L1
        longitud=len(l1)
        print("la longitud de l1 es:",longitud)

la longitud de l1 es: 4
```

2.3 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 [17]: ### Obtener el valor 5.7
         posicion_1=0
         for i in range(0,len(l1)):
             if l1[i]==5.7:
                 print ("El valor 5.7 esta en la posicion:", i)
                 posicion_1=i

         ### obtener el valor de 5
         posicion_2=0
         for j in range(0,len(l1)):
```

```
            if j==(len(l1)-1):
                for h in range(0,len(l1[j])):
                    if l1[j][h]==5:
                        print ("El valor 5 esta en la posicion:", h, ", de la lista de l1 qu
                        posicion_2=h
```

```
El valor 5.7 esta en la posicion: 2
El valor 5 esta en la posicion: 2 , de la lista de l1 que esta en la posicion: 3
```

2.4 4. [1]

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

2.5 Rta:

Yo predigo que el programa mostrará error dado que las posiciones de las listas en python empiezan en cero, por ende el último elemento es len(l1)-1 en este caso 3. Con 4 salimos del rango de posiciones de la lista l1.

```
In [18]: l1[4]
```

```
---------------------------------------------------------------------------

IndexError                                Traceback (most recent call last)

<ipython-input-18-7ffdcb2c9f2e> in <module>()
----> 1 l1[4]


IndexError: list index out of range
```

2.6 5. [1]

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

2.7 Rta:

Yo predigo que el programa mostrará el último elemento de la lista, en este caso 16, dado que el -1 indica eso.

```
In [21]: l2[-1]
```

```
Out[21]: 16
```

2.8 6. [1]

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

```
In [39]: print("lista l1:", l1)
         for j in range(0,len(l1)):
             if j==(len(l1)-1):
                 for h in range(0,len(l1[j])):
                     if l1[j][h]==3:
                         #cambiamos
                         l1[j][h]=15.0

         print("lista l1, con cambio en el ultimo elemento:", l1)
```

```
lista l1: [1, 'abc', 5.7, [1, 3, 5]]
lista l1, con cambio en el ultimo elemento: [1, 'abc', 5.7, [1, 15.0, 5]]
```

2.9 7. [1]

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

```
In [45]: # Segundo elemento seria 11 y el quinto seria 14
         slice_1=l2[1:5]
         print(slice_1)
```

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

```
In [51]: ### Ahora si es por posiciones (en caso tal de que no haya entendido bien la pregunta,
         slice_2=l2[2:5]
         print(slice_2)
```

```
[12, 13, 14]
```

2.10 8. [1]

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

```
In [54]: slice_3=l2[:3]
         print("lista l2:",l2)
         print("lista con slice:", slice_3)
```

```
lista l2: [10, 11, 12, 13, 14, 15, 16]
lista con slice: [10, 11, 12]
```

2.11 9. [1]

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

```
In [56]: slice_4=l2[1:]
         print("lista l2:",l2)
         print("lista con slice:", slice_4)
```

```
lista l2: [10, 11, 12, 13, 14, 15, 16]
lista con slice: [11, 12, 13, 14, 15, 16]
```

2.12 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 [66]: import random as r
         ### Agregare numeros aleatorios

         l0=[]

         for i in range(0,4):
             b=r.randint(0,100)
             l0.append(b)
             print(i)
         print("lista l0 agregando cuatro elementos: ", l0)

         ## Eliminar elemento
         l0.pop(2)
         print("lista l0 eliminando el tercer elemento: ", l0)

         print( " cantidad de appends fueron: 4")
```

```
0
1
2
3
lista l0 agregando cuatro elementos:  [40, 21, 27, 77]
lista l0 eliminando el tercer elemento:  [40, 21, 77]
 cantidad de appends fueron: 4
```

2.13 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 [7]: n1=l1+l0
        print("n1 concatenando l1 y l0:", n1)

        # cambio el primero elemento de n1
        n1[0]="hola"
        print("n1 cambiando el primer elemento de '1' a 'hola': ", n1)

        print("l0:",l0)

        print("l1:",l1)
```

```
n1 concatenando l1 y l0: [1, 'abc', 5.7, [1, 3, 5]]
n1 cambiando el primer elemento de '1' a 'hola':  ['hola', 'abc', 5.7, [1, 3, 5]]
l0: []
l1: [1, 'abc', 5.7, [1, 3, 5]]
```

2.14 Rta:

Ni l0, ni l1 cambian dado n1 es una especie de copia de l1+l0, esta "copia" no cambia los valores originales de las listas.

2.15 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 [1]: ### Construyo la lista l3 ##

        l3=[1,2,3,4,5,6,8,7,9,10,0,-1,-2,-3,-4,-5,-7]

        all_pos=True
        for i in l3:
            if i<=0:
                all_pos=False
                break
        if all_pos ==True:
            print("La lista l3 solo tiene valores positivos")
        else:
            print("la lista l3 puede tener valores negativos o igual a cero")
```

```
la lista l3 puede tener valores negativos o igual a cero
```

2.16 13. [2]

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

```
In [3]: #### lista solo valores positivos

        lista_positivos=[]
        for j in l3:
            if j>0:
                lista_positivos.append(j)

        print("la lista de número positvos de l3 es: ", lista_positivos)
```

```
la lista de número positvos de l3 es:  [1, 2, 3, 4, 5, 6, 8, 7, 9, 10]
```

2.17 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 [10]: n1=[]
         for j in l3:
             if j >0:
                 n1.append(True)
             else:
                 n1.append(False)

         print("l3:", l3)
         print("n1:", n1)
```

```
l3: [1, 2, 3, 4, 5, 6, 8, 7, 9, 10, 0, -1, -2, -3, -4, -5, -7]
n1: [True, True, True, True, True, True, True, True, True, True, False, False, False, False, Fa
```

2.18 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 [12]: n1=[False]*len(l3)

         for i in range(0,len(l3)):
             if l3[i]>0:
                 n1[i]=True

         print("l3: ", l3)
         print("n1: ", n1)
```

```
l3:  [1, 2, 3, 4, 5, 6, 8, 7, 9, 10, 0, -1, -2, -3, -4, -5, -7]
n1:  [True, True, True, True, True, True, True, True, True, True, False, False, False, False, F
```

2.19 16. [4]

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

```
In [ ]: #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:

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

```
In [91]: import random

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

         lista_contador=[0]*10
         for i in random_numbers:
             if i==0:
                 lista_contador[i]=lista_contador[i]+1
             elif i==1:
                 lista_contador[i]=lista_contador[i]+1
             elif i==2:
                 lista_contador[i]=lista_contador[i]+1
             elif i==3:
                 lista_contador[i]=lista_contador[i]+1
             elif i==4:
                 lista_contador[i]=lista_contador[i]+1
             elif i==5:
                 lista_contador[i]=lista_contador[i]+1
             elif i==6:
                 lista_contador[i]=lista_contador[i]+1
             elif i==7:
                 lista_contador[i]=lista_contador[i]+1
             elif i==8:
                 lista_contador[i]=lista_contador[i]+1
             else:
                 lista_contador[i]=lista_contador[i]+1

         completa=[]
```

```
        for j in range(0,10):
            pareja=[str(j)+":",lista_contador[j]]
            completa.append(pareja)
        print("La cantidad de cada uno de los numeros de 0 a 9 en la lista random numbers es:
```

```
La cantidad de cada uno de los numeros de 0 a 9 en la lista random numbers es:  [['0:', 959],
```

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.
