mcpp_taller_3_juan_munoz

August 22, 2019

1 Taller 3

Métodos Computacionales para Políticas Públicas - URosario

Entrega: viernes 23-ago-2019 11:59 PM

Juan Sebastián Muñoz Vargas jsebastianmvargas@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 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: run $2019_2mcpp_taller_3listas_ejemplos.pyEstearchivocontienetreslistas(l0, l1yl2)queusarparalastareasdeestaseccin.Puedeve$

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]

[1]:
```
run 2019_2_mcpp_taller_3_listas_ejemplos.py
```

[3]:
```
l0
```

[3]:
```
[]
```

[4]:
```
l1
```

[4]:
```
[1, 'abc', 5.7, [1, 3, 5]]
```

[5]:
```
l2
```

[5]:
```
[10, 11, 12, 13, 14, 15, 16]
```

1.2 1. [1]

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

[6]:
```
Lista1 = [7, "xyz", 2.7]
Lista1
```

[6]:
```
[7, 'xyz', 2.7]
```

1.3 2. [1]

Halle la longitud de la lista l1.

[7]:
```
len (l1)
```

[7]:
```
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 cuarto elemento de l1.

[8]:
```
l1
```

[8]:
```
[1, 'abc', 5.7, [1, 3, 5]]
```

[11]:
```
l1[2]
```

[11]:
```
5.7
```

[12]:
```
l1[3][2]
```

[12]:
```
5
```

1.5 4. [1]

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

Va a aparecer un error porque a pesar de que hay cuatro elementos, en python la base es 0. Por lo que solo va a haber de 0 - 3 y 4 no encontraría nada en la lista

[13]:
```
l1 [4]
```

```
␣
↪---------------------------------------------------------------------------

       IndexError                                Traceback (most recent call␣
↪last)

       <ipython-input-13-ad16827f93d4> 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.

Va a apareer el número 16, ya que es el último elemento de la lista y, esto es precisamente, lo que hace el -1.

[14]:
```
l2 [-1]
```

[14]:
```
16
```

1.7 6. [1]

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

[15]:
```
l1 [3][2] = 15.0
l1
```

[15]:
```
[1, 'abc', 5.7, [1, 3, 15.0]]
```

1.8 7. [1]

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

[16]:
```
l2 [1:5]
```

[16]:
```
[11, 12, 13, 14]
```

1.9 8. [1]

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

[18]:
```
l2 [:3]
```

[18]:
```
[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.

[20]:
```
l2 [1: len (l2)]
```

[20]:
```
[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?

[23]:
```
l0.append (1)
l0.append (2)
l0.append (3)
l0.append (4)
```

[24]:
```
l0.pop (2)
l0
```

[24]:
```
[1, 2, 4]
```

Tuve que hacer uso de tres append's

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?

[25]:
```
n1 = l0 + l1
n1
```

[25]:
```
[1, 2, 4, 1, 'abc', 5.7, [1, 3, 15.0]]
```

[26]:
```
n1 [2] = 44
n1
```

[26]:
```
[1, 2, 44, 1, 'abc', 5.7, [1, 3, 15.0]]
```

No cambia niguno de los valores de l0 o de l1, ya que las modificaciones se están haciendo en la nueva lista

[27]:
```
l0
```

[27]:
```
[1, 2, 4]
```

[28]:
```
l1
```

[28]:
```
[1, 'abc', 5.7, [1, 3, 15.0]]
```

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.

[39]:
```
l3 = [-4, -3, -2, -1, 1, 2, 3, 4]
all_pos = True
for i in l3:
    if i <0:
        all_pos = False
    else:
        continue

print (all_pos)
```

```
True
```

1.14 13. [2]

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

[41]:
```
l3 = [-4, -3, -2, -1, 1, 2, 3, 4]
newlist = []
for i in l3:
    if i > 0:
        newlist.append (i)
    else:
        continue
newlist
```

[41]:
```
[1, 2, 3, 4]
```

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.

[45]:
```
n1 = []
l3 = [-4, -3, -2, -1, 1, 2, 3, 4]
for i in l3:
    if i > 0:
        n1.append (True)
    else:
        n1.append (False)
n1
```

[45]:
```
[False, False, False, False, True, True, True, True]
```

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 False en otro caso.

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

[9]:
```
n1 = []
l3 = [-4, -3, -2, -1, 1, 2, 3, 4]

for i in range(len(l3)):
    if l3 [i] > 0:
        n1.append (True)
    else:
        n1.append (False)
n1
```

[9]: `[False, False, False, False, True, True, True, True]`

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$_n$umbers = $[]foriinrange(N)$: $random_numbers.append(random.randint(0,9))Ycreamosun"co$
$count = []forxinrange(0,10)$: $count.append(random_numbers.count(x))Creeun"contador"quehagalomismo, perosin$

[19]:
```
import random
N =10000
not_a_counter = [0,0,0,0,0,0,0,0,0,0]
for i in range (N):
    i = random.randint(0,9)
    not_a_counter [i] = fiction_counter [i] +1
not_a_counter
```

[19]: `[1025, 1007, 963, 945, 994, 978, 1044, 1004, 1008, 1042]`

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.
