mcpp_taller3_juan_salgado

August 22, 2019

1 Taller 3

Métodos Computacionales para Políticas Públicas - URosario

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

Juan Camilo Salgado Ramírez juanca.salgado@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:

[1]:
```
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]

1.2 1. [1]

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

[2]:
```
lista_creada = [7, "xyz", 2.7]
lista_creada
```

[2]:
```
[7, 'xyz', 2.7]
```

1.3 2. [1]

Halle la longitud de la lista l1.

[3]:
```
len(l1)
```

[3]:
```
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.

[4]:
```
print(l1[2])
print(l1[len(l1)-1][2])
```

```
5.7
5
```

1.5 4. [1]

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

[5]:
```
## Python 'indexa' desde cero. Es decir que para extraer la primera posición de␣
 ↪una lista 'x' se debe usar
## x[0] en vez de x[1]. Por lo tanto, todas las posiciones se deben reducir en␣
 ↪una unidad para extraerlas.
## En el caso de 'l1', que tiene 4 elementos, para extraer la cuarta posición␣
 ↪se debe usar l1[3] en vez de
## l1[4], por la lógica explicada anteriormente. Si se usa l1[4] debería salir␣
 ↪un error, porque la posición
## '4' no existe en 'l1'.

l1[4]
```

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

```
IndexError                                Traceback (most recent call last)

<ipython-input-5-bdf3b6c8f55e> in <module>
  5 ## '4' no existe en 'l1'.
  6
----> 7 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.

[6]:
```
## Cuando se usa el índice '-1' se extrae el último elemento de la lista. Por lo tanto, l2[-1] debería
## arrojar el valor 16.

l2[-1]
```

[6]:
```
16
```

1.7 6. [1]

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

[7]:
```
print(" Lista inicial: ", l1)

l1[len(l1)-1][1] = 15.0

print("\n Lista final: ", l1)
```

```
 Lista inicial:  [1, 'abc', 5.7, [1, 3, 5]]

 Lista final:  [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.

[8]:
```
print(l2)
l2[1:5]
```

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

[8]:
```
[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.

[9]:
```
print(l2)
l2[:3]
```

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

[9]:
```
[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.

[10]:
```
print(l2)
l2[1:]
```

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

[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?

[11]:
```
## Número de 'appends' a hacer: 4 (uno por cada número)

print(' l0:',l0)

v1 = 9
v2 = 8
v3 = 7
v4 = 6
l0.append(v1)
l0.append(v2)
l0.append(v3)
l0.append(v4)

print('\n l0:',l0)

l0.remove(l0[2])
print('\n l0:',l0)
```

```
 l0: []

 l0: [9, 8, 7, 6]

 l0: [9, 8, 6]
```

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?

[12]:
```
## En este caso, si bien 'n1' se construye usando 'l0' y 'l1', para el␣
 ↪computador serían variables distintas.
## Por lo tanto no es necesario usar el 'slicing' [:]. Si 'n1' fuese construida␣
 ↪como n1 = l0 o n1 = l1, si
## sería necesario utilizar [:].

n1 = l0 + l1
print(" Es n1 igual a l0 o l1:",n1 == l0 or n1 == l1)
print("\n n1:",n1)
print("\n l0:",l0)
print("\n l1:",l1)

l0[0] = [4,4,4]
print("\n Cambio en l0: no afecta a n1")
print("\n n1:",n1)
print("\n l0:",l0)

n1[0] = [1,2,3]
print("\n Cambio en n1: no afecta a l0")
print("\n n1:",n1)
print("\n l0:",l0)
```

```
Es n1 igual a l0 o l1: False

n1: [9, 8, 6, 1, 'abc', 5.7, [1, 15.0, 5]]

l0: [9, 8, 6]

l1: [1, 'abc', 5.7, [1, 15.0, 5]]

Cambio en l0: no afecta a n1

n1: [9, 8, 6, 1, 'abc', 5.7, [1, 15.0, 5]]

l0: [[4, 4, 4], 8, 6]

Cambio en n1: no afecta a l0

n1: [[1, 2, 3], 8, 6, 1, 'abc', 5.7, [1, 15.0, 5]]

l0: [[4, 4, 4], 8, 6]
```

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.

[13]:
```
l3 = [0,-6,7,8,-3,-5,6,-7]
all_pos = True

for val in l3:
    if val <= 0:
        all_pos = False
        break

print(all_pos)
```

```
False
```

1.14 13. [2]

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

[14]:
```
l4 = l3[:]
i = 0
count = 0
print("l4 inicial:", l4)

while (i < len(l4)):
    if l4[i] <= 0:
        l4.remove(l4[i])
        continue
    i+=1

print("\nl4 final:", l4)
```

```
l4 inicial: [0, -6, 7, 8, -3, -5, 6, -7]

l4 final: [7, 8, 6]
```

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.

[15]:
```
n1 = []

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

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

```
 l3: [0, -6, 7, 8, -3, -5, 6, -7]

 n1: [False, False, True, True, False, 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 False en otro caso.

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

[16]:
```
n1 = []

for i in range(len(l3)):
    n1.append(False)

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

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

```
 l3: [0, -6, 7, 8, -3, -5, 6, -7]

 n1: [False, False, True, True, False, 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:

[17]:
```
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:

[18]:
```
count = []
for x in range(0,10):
    count.append(random_numbers.count(x))
```

[19]:
```
count_tot = []
for i in range(0,10):
    count_tot.append(["Freq " + str(i), 0])

for val in random_numbers:
    count_tot[val][1] = count_tot[val][1] + 1

print(' Clase:',count)
print('\n Propio:',count_tot)
```

```
 Clase: [1023, 982, 958, 1049, 955, 1030, 1024, 970, 1021, 988]

 Propio: [['Freq 0', 1023], ['Freq 1', 982], ['Freq 2', 958], ['Freq 3', 1049],
['Freq 4', 955], ['Freq 5', 1030], ['Freq 6', 1024], ['Freq 7', 970], ['Freq 8',
1021], ['Freq 9', 988]]
```

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.
