Taller 3

Métodos Computacionales para Políticas Públicas - URosario

Entrega: viernes 22-feb-2019 11:59 PM

[Su nombre acá]

[Su e-mail acá]

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_1_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_1_mcpp_taller_3_listas_ejemplos.py

In [2]:

```
run 2019_1_mcpp_taller_3_listas_ejemplos.py
```

In [3]:

```
l0
```

Out[3]:

```
[]
```

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_1_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 [40]:

```
run 2019_1_mcpp_taller_3_listas_ejemplos.py
```

In [ ]:

```

```

1. [1]

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

In [4]:

```
lista = [7, "xyz", 2.7]
```

2. [1]

Halle la longitud de la lista l1.

In [4]:

```
len (l1)
```

Out[4]:

```
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 [11]:

```
l1[2]
```

Out[11]:

```
5.7
```

In [23]:

```
l1[3][2]
```

Out[23]:

```
5
```

4. [1]

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

No habrá ningun elemento, por tanto exisitíra un error de indexación

In [24]:

```

```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-24-7ffdcb2c9f2e> in <module>
----> 1 l1[4]

IndexError: list index out of range
```

5. [1]

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

El output será el último elemento del conjunto a evaluar

In [25]:

```
l2[-1]
```

Out[25]:

```
16
```

6. [1]

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

In [26]:

```
l1[2]
```

Out[26]:

```
5.7
```

In [29]:

```
l1[2] = 15.0
```

In [30]:

```
l1
```

Out[30]:

```
[1, 'abc', 15.0, [1, 3, 5]]
```

7. [1]

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

In [33]:

```
l2[2:7]
```

Out[33]:

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

In [31]:

```
l2
```

Out[31]:

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

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

8. [1]

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

In [35]:

```
l2[0:3]
```

Out[35]:

```
[10, 11, 12]
```

9. [1]

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

In [36]:

```
l2[1:7]
```

Out[36]:

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

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?

Utilizaré la versión 4 veces

In [7]:

```
l0.append(1)
l0.append(2)
l0.append(3)
l0.append(4)
```

In [8]:

```
l0
```

Out[8]:

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

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 [69]:

```
n1 = (l0 + l1)
print(n1)
```

```
['lunes', 'martes', 'miercoles', 'jueves', 1, 'abc', 5.7, [1, 3, 5]]
```

In [70]:

```
n1[2] = 15.0
```

No cambia nada de las listas l0 y l1

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 [18]:

```
l3 = [12, 13, 14, 15, 16, -2]
```

In [19]:

```
for n in l3:
    if n >= 0:
        all_pos = True
        continue

    elif n < 0:
        all_pos = False
print(all_pos)
```

```
False
```

13. [2]

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

In [20]:

```
l3_pos = []

for n in l3:
    if n >= 0:
        l3_pos.append(n)
    elif n < 0:
        continue
print(l3_pos)
```

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

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 [21]:

```
nl2 = []

for n in l3:
    if n >= 0:
        n = True
        nl2.append(n)
    elif n < 0:
        n = False
        nl2.append(n)
print(nl2)
```

```
[True, True, True, True, True, False]
```

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 [22]:

```
len(l3)
```

Out[22]:

```
6
```

In [32]:

```
lf = []

for n in range(23):
    n = False
    lf.append(n)
print(lf)

for n in range(lf):
    if n >= 0:
        n = True
        nl3.append(n)
    elif n < 0:
        n = False
        nl3.append(n)
print(nl3)
```

```
[False, False, False, False, False, False, False, False, False, False, False, False, False, False,
False, False, False, False, False, False, False, False, False]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-32-2c4961b871f6> in <module>
      6 print(lf)
      7
----> 8 for n in range(lf):
      9     if n >= 0:
     10         n = True

TypeError: 'list' object cannot be interpreted as an integer
```

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 [60]:

```
N = 10000
random_numbers = []
```

```
Lista = [0,0,0,0,0,0,0,0,0,0]
for index, item in enumerate(Lista):
    random_numbers.append(random.randint(0,9))
print(random_numbers)
for n in random_numbers:
    if n == 0:
        n = 1
        Lista[0] = n + 1
    elif n == 1:
        n = 1
        Lista[1] = n + 1
    elif n == 2:
        n = 1
        Lista[2] = n + 1
    elif n == 3:
        n = 1
        Lista[3] = n + 1
print("contador")
print(Lista)
```

```
[5, 1, 0, 9, 6, 0, 9, 1, 3, 4]
contador
[2, 2, 0, 2, 0, 0, 0, 0, 0, 0]
```