Taller 5

Métodos Computacionales para Políticas Públicas - URosario

Entrega: viernes 18-sep-2020 11:59 PM

[Nicolás Garcés Rodríguez]

[nicolas.garces@urosario.edu.co]

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_taller5_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. Si tiene algún problema con la conversión, descárguelo en HTML.
  2. Suba los dos archivos (.pdf -o .html- y .ipynb) a su repositorio en GitHub antes de la fecha y hora límites.

(Todos los ejercicios tienen el mismo valor.)

1

Escríba una función que ordene (de forma ascedente y descendente) un diccionario según sus valores.

In [20]:

```
def sorter(x):
    a= dict(sorted(x.items()))
    b= dict(sorted(x.items(), reverse=True))
    print(f'la forma ascedente de {x} es {a}')
    print(f'la forma descendente de {x} es {b}')
```

In [21]:

```
aa = {'b':4,'c':6,'a':5}
sorter(aa)
```

```
la forma ascedente de {'b': 4, 'c': 6, 'a': 5} es {'a': 5, 'b': 4, 'c': 6}
la forma descendente de {'b': 4, 'c': 6, 'a': 5} es {'c': 6, 'b': 4, 'a': 5}
```

2

Escriba una función que agregue una llave a un diccionario.

In [195]:

```
def add(dicc):
    element = str(input("Que llave quiere añadir?:"))
    dicc[element]= dicc.get(element)
    return dicc.items()
```

In [196]:

```
a= {'Nicolas':0 ,'Laura':1}
add(a)
```

```
Que llave quiere añadir?:Sofia
```

Out[196]:

```
dict_items([('Nicolas', 0), ('Laura', 1), ('Sofia', None)])
```

3

Escriba un programa que concatene los siguientes tres diccionarios en uno nuevo:

dicc1 = {1:10, 2:20} dicc2 = {3:30, 4:40} dicc3 = {5:50,6:60} Resultado esperado: {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}

In [197]:

```
def unite(a,b,c):
    one = {**a,**b,**c}
    return one
```

In [198]:

```
dicc1 = {1:10, 2:20}
dicc2 = {3:30, 4:40}
dicc3 = {5:50, 6:60}
unite(dicc1,dicc2,dicc3)
```

Out[198]:

```
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
```

4

Escriba una función que verifique si una determinada llave existe o no en un diccionario.

In [199]:

```
def isin(dicc):
    element = str(input('Que elemento busca?:'))
    if (element in dicc.keys()) == True:
        print(f'{element} esta en {dicc}')
    else:
        print(f'{element} no esta en {dicc}')
```

In [200]:

```
a= {'Nicolas':0 ,'Laura':1}
isin(a)
```

```
Que elemento busca?:Nicolas
Nicolas esta en {'Nicolas': 0, 'Laura': 1}
```

In [52]:

```
a= {'Nicolas':0 ,'Laura':1}
isin(a)
```

```
Que elemento busca?:Sofia
Sofia no esta en {'Nicolas': 0, 'Laura': 1}
```

5

Escriba una función que imprima todos los pares (llave, valor) de un diccionario.

In [1]:

```
def printer(dicc):
     for k,v in dicc.items():
        print(f'{k} {v}')
```

In [2]:

```
a= {'Nicolas':0 ,'Laura':1}
printer(a)
```

```
Nicolas 0
Laura 1
```

6

Escriba una función que genere un diccionario con los números enteros entre 1 y n en la forma (x: x**2).

In [3]:

```
def generator(n):
    dic= dict()
    for i in range(1,n+1):
        dic[i]= i**2
    return dic
```

In [4]:

```
generator(5)
```

Out[4]:

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

7

Escriba una función que sume todas las llaves de un diccionario. (Asuma que son números.)

In [207]:

```
def sum_keys(dic):
    numbers= list(dic.keys())
    a=0
    for i in numbers:
        a= a+i
    print(a)
```

In [208]:

```
a={1:3,2:4,3:5}
sum_keys(a)
```

```
6
```

8

Escriba una función que sume todos los valores de un diccionario. (Asuma que son números.)

In [209]:

```
def sum_values(dic):
    numbers= list(dic.values())
    a=0
    for i in numbers:
        a= a+i
    print(a)
```

In [210]:

```
a={1:3,2:4,3:5}
sum_values(a)
```

```
12
```

9

Escriba una función que sume todos los ítems de un diccionario. (Asuma que son números.)

In [211]:

```
def sum_all(dic):
    numbers= list(dic.keys())
    numbers_two=list(dic.values())
    lista = numbers + numbers_two
    a=0
    for i in lista:
        a= a+i
    print(a)
```

In [212]:

```
d ={2:1, 3:2, 4:5}
sum_all(d)
```

```
17
```

10

Escriba una función que tome dos listas y las mapee a un diccionario por pares. (El primer elemento de la primera lista es la primera llave del diccionario, el primer elemento de la segunda lista es el valor de la primera llave del diccionario, etc.)

In [214]:

```
def mapper(a,b):
    dic= dict(zip(a,b))
    return dic
```

In [215]:

```
lis = [2,4,6,7,9,0]
li =  [1,3,5,7,9,0]
mapper(lis,li)
```

Out[215]:

```
{2: 1, 4: 3, 6: 5, 7: 7, 9: 9, 0: 0}
```

11

Escriba una función que elimine una llave de un diccionario.

In [124]:

```
def deleter(a):
```

```
    delet = str(input('Que elemento quiere eliminar?:'))
    del a[delet]
    return a
```

In [125]:

```
a= {'Nicolas':0 ,'Laura':1}
deleter(a)
```

```
Que elemento quiere eliminar?:Nicolas
```

Out[125]:

```
{'Laura': 1}
```

12

Escriba una función que arroje los valores mínimo y máximo de un diccionario.

In [138]:

```
def ext(x):
    vals = x.values()
    print(f"el valor maximo del diccionario es {max(vals)}")
    print(f"el valor minimo del diccionario es {min(vals)}")
```

In [139]:

```
d ={2:1, 3:2, 4:5}
ext(d)
```

```
el valor maximo del diccionario es 5
el valor minimo del diccionario es 1
```

13

sentence = "the quick brown fox jumps over the lazy dog" words = sentence.split() word_lengths = [] for word in words: if word != "the": word_lengths.append(len(word))

Simplifique el código anterior combinando las líneas 3 a 6 usando list comprehension. Su código final deberá entonces tener tres líneas.

In [219]:

```
sentence = "the quick brown fox jumps over the lazy dog"
words = sentence.split()
word_lengths= [len(x) for x in words if x!='the']
word_lengths
```

Out[219]:

```
[5, 5, 3, 5, 4, 4, 3]
```

In [141]:

```
sentence = "the quick brown fox jumps over the lazy dog"
words = sentence.split()
word_lengths = []
for word in words:
    if word != "the":
        word_lengths.append(len(word))
word_lengths
```

Out[141]:

```
[5, 5, 3, 5, 4, 4, 3]
```

14

Escriba UNA línea de código que tome la lista `a` y arroje una nueva lista con solo los elementos pares de `a`.

In [152]:

```
a=[1,2,3,4,5,6,7,8,9,0]
[x for x in a if x%2 ==0]
```

Out[152]:

```
[2, 4, 6, 8, 0]
```

15

Escriba UNA línea de código que tome la lista `a` del ejercicio 14 y multiplique todos sus valores.

In [159]:

```
from functools import reduce
a=[1,2,3,4,5,6,7,8,9]
reduce(lambda x,y: x*y, a)
```

Out[159]:

```
362880
```

16

Usando "list comprehension", cree una lista con las 36 combinaciones de un par de dados, como tuplas: [(1,1),(1,2),...,(6,6)].

In [177]:

```
import itertools
list(itertools.permutations(range(1,7), 2))+ list((x,x) for x in range(1,7))
```

Out[177]:

```
[(1, 2),
 (1, 3),
 (1, 4),
 (1, 5),
 (1, 6),
 (2, 1),
 (2, 3),
 (2, 4),
 (2, 5),
 (2, 6),
 (3, 1),
 (3, 2),
 (3, 4),
 (3, 5),
 (3, 6),
 (4, 1),
 (4, 2),
 (4, 3),
 (4, 5),
 (4, 6),
 (5, 1),
 (5, 2),
 (5, 3),
 (5, 4),
 (5, 6),
 (6, 1),
 (6, 2),
 (6, 3),
 (6, 4),
```

```
(6, 5),
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(6, 6)]
```