mcpp_taller_5_juan_munoz

September 3, 2019

1 Taller 5

Métodos Computacionales para Políticas Públicas - URosario

Entrega: viernes 6-sep-2019 11:59 PM

Juan Sebastián Muñoz 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_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.)

1.1.1 1

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

[11]:
```
def order (dicti):
    y = (sorted(dicti.items(), key = lambda x : x[1]))
    return y

def unorder (dicti):
```

```
    y = (sorted(dicti.items(), key = lambda x : x[1], reverse=True))
    return y
dict0 = {"a": 4, "b": 9, "c": 8}
c = order(dict0)
print (c)
d = unorder(dict0)
print (d)
```

```
[('a', 4), ('c', 8), ('b', 9)]
[('b', 9), ('c', 8), ('a', 4)]
```

1.1.2 2

Escriba una función que agregue una llave a un diccionario.

[43]:
```
def addtodict (dictionary, key, value = 0):
    '''
    This function adds a new key and value to an existing dictionary
    Inputs:
    dictionary = The dict where we want the new element
    key = the key we want to add
    value = the value to be added
    '''
    y = dictionary.update ({key: value})
    return y
dict0 = {"a": 4, "b": 9, "c": 8}
addtodict (dict0, "h")
dict0
```

```
None
```

[43]:
```
{'a': 4, 'b': 9, 'c': 8, 'h': 0}
```

1.1.3 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}

[58]:
```
dicc1 = {1:10, 2:20}
dicc2 = {3:30, 4:40}
dicc3 = {5:50,6:60}
new_dic = {}
new_dic.update (dicc1)
new_dic.update (dicc2)
new_dic.update (dicc3)
new_dic
```

[58]:
```
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
```

1.1.4 4

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

[65]:
```
def check(dictionary, k):
    existence = False
    if k in dictionary:
        existence = True
    return existence
dict0 = {"a": 4, "b": 9, "c": 8}
check (dict0, "d")
```

[65]:
```
False
```

1.1.5 5

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

[72]:
```
def print_dict (dic):
    for k, v in dic.items():
        print(k,v)
dict0 = {"a": 4, "b": 9, "c": 8}
print_dict (dict0)
```

```
a 4
b 9
c 8
```

1.1.6 6

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

[133]:
```
def dictn (n):
    a = {x+1:0 for x in range (n)}
    return a
b = dictn (20)
b
```

[133]:
```
{1: 0,
 2: 0,
 3: 0,
 4: 0,
 5: 0,
 6: 0,
 7: 0,
 8: 0,
 9: 0,
 10: 0,
 11: 0,
 12: 0,
```

```
 13: 0,
 14: 0,
 15: 0,
 16: 0,
 17: 0,
 18: 0,
 19: 0,
 20: 0}
```

1.1.7 7

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

[134]:
```
c = {1 : 2, 2: 3}
sum_llaves(c)
```

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

NameError                                 Traceback (most recent call␣
↪last)

<ipython-input-134-4de57a72bb8e> in <module>
  1 c = {1 : 2, 2: 3}
----> 2 sum_llaves(c)


NameError: name 'sum_llaves' is not defined
```

[145]:
```
def sum_llaves (dic):
    a = 0
    for k, v in dic.items():
        a = k + a
    return a
c = {1 : 2, 2: 3}
sum_llaves (c)
```

[145]: 3

1.1.8 8

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

[146]:
```
def sum_valores (dic):
    a = 0
    for k, v in dic.items():
        a = v + a
```

```
    return a
c = {1 : 2, 2: 3}
sum_valores (c)
```

[146]: 5

1.1.9 9

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

[189]:
```
d ={2:1, 3:2, 4:5}
```

[147]:
```
def sum_items (dic):
    a = 0
    for k, v in dic.items():
        a = k + v + a
    return a
d ={2:1, 3:2, 4:5}
sum_items (d)
```

[147]: 17

1.1.10 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.)

[161]:
```
def lists_to_dict (list1, list2):
    new_dict = dict (zip (list1, list2))
    return new_dict
a = (1, 2, 3, 4)
b = ("a", "b", "c", "d")
c = lists_to_dict (a, b)
print (c)
```

```
{1: 'a', 2: 'b', 3: 'c', 4: 'd'}
```

1.1.11 11

Escriba una función que elimine una llave de un diccionario.

[164]:
```
def del_key (my_dict, key):
    my_dict.pop(key, None)
    return my_dict
dict0 = {1: 'a', 2: 'b', 3: 'c', 4: 'd'}
llave = 3
del_key (dict0, llave)
```

[164]: {1: 'a', 2: 'b', 4: 'd'}

1.1.12 12

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

[199]:
```
def find_min_max (d):
    a = d [min (d, key = lambda x: d[x])]
    b = d [max (d, key = lambda x: d[x])]
    print ('El valor mínimo es', a)
    print ('El valor máximo es', b)
dict0 = {'a': 7, 'b': 5, 'c': 2, 'd': 9}
find_min_max (dict0)
```

```
El valor mínimo es 2
El valor máximo es 9
```

1.1.13 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.

[213]:
```
sentence = "the quick brown fox jumps over the lazy dog"
words = sentence.split()
[len(word) for word in words if word != "the"]
```

[213]:
```
[5, 5, 3, 5, 4, 4, 3]
```

1.1.14 14

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

[294]:
```
a = [1,2,3,4,5,6,7,8,9,]
[b for b in a if b%2!=1]
```

[294]:
```
[2, 4, 6, 8]
```

1.1.15 15

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

[269]:
```
a = [1,2,3,4,5,6,7,8,9,]
from functools import reduce
w = reduce((lambda x, y: x*y), a)
w
```

[269]:
```
362880
```

1.1.16 16

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

[281]:
```
m = [1,2,3,4,5,6]
[(a,i) for a in m for i in m]
```

[281]:
```
[(1, 1),
 (1, 2),
 (1, 3),
 (1, 4),
 (1, 5),
 (1, 6),
 (2, 1),
 (2, 2),
 (2, 3),
 (2, 4),
 (2, 5),
 (2, 6),
 (3, 1),
 (3, 2),
 (3, 3),
 (3, 4),
 (3, 5),
 (3, 6),
 (4, 1),
 (4, 2),
 (4, 3),
 (4, 4),
 (4, 5),
 (4, 6),
 (5, 1),
 (5, 2),
 (5, 3),
 (5, 4),
 (5, 5),
 (5, 6),
 (6, 1),
 (6, 2),
 (6, 3),
 (6, 4),
 (6, 5),
 (6, 6)]
```