mcpp_taller5_juan_salgado

September 6, 2019

1 Taller 5

Métodos Computacionales para Políticas Públicas - URosario

Entrega: viernes 6-sep-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_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.

```
In [1]: dic = {2:'b',  4:'d', 0: 'aa',1:'a', 3:'c'}
        def order(dic):
            ## Ascendente
            as_dic = list(dic.items())
```

```
            for i in range(len(as_dic)-1):
                for j in range(i, len(as_dic)):
                    if as_dic[i][1] > as_dic[j][1]:
                        aux = as_dic[j]
                        as_dic[j] = as_dic[i]
                        as_dic[i] = aux
            ## Descendente
            des_dic = list(dic.items())
            for i in range(len(des_dic)-1):
                for j in range(i, len(des_dic)):
                    if des_dic[i][1] < des_dic[j][1]:
                        aux = des_dic[j]
                        des_dic[j] = des_dic[i]
                        des_dic[i] = aux
            print(as_dic)
            print(des_dic)

        print(dic)
        order(dic)
```

```
{2: 'b', 4: 'd', 0: 'aa', 1: 'a', 3: 'c'}
[(1, 'a'), (0, 'aa'), (2, 'b'), (3, 'c'), (4, 'd')]
[(4, 'd'), (3, 'c'), (2, 'b'), (0, 'aa'), (1, 'a')]
```

1.1.2 2

Escriba una función que agregue una llave a un diccionario.

```
In [2]: def agregar(dic, llave):
            dic[llave] = ""

        print(dic)
        agregar(dic, 'nueva llave')
        print(dic)
```

```
{2: 'b', 4: 'd', 0: 'aa', 1: 'a', 3: 'c'}
{2: 'b', 4: 'd', 0: 'aa', 1: 'a', 3: 'c', 'nueva llave': ''}
```

1.1.3 3

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

```
In [3]: 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 [4]: def agregar_dic(dic_ini, dic_agr):
            for k, v in dic_agr.items():
                dic_ini[k] = v

        dic_nuevo = {}
        agregar_dic(dic_nuevo, dicc1)
        agregar_dic(dic_nuevo, dicc2)
        agregar_dic(dic_nuevo, dicc3)

        print(dic_nuevo)
```

```
{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.

```
In [5]: def existe_llave(dic, llave):
            return llave in dic

        dic = {'a': 1, 'b': 2, 'c': 3}
        print(dic.keys())
        print('Llave 2 en diccionario:', existe_llave(dic, 2))
```

```
dict_keys(['a', 'b', 'c'])
Llave 2 en diccionario: False
```

1.1.5 5

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

```
In [6]: def imprimir_vals(dic):
            for k, v in dic.items():
                print('Llave:', k, 'Valor:', v)

        dic = {1:2, 3:4, 5:6}
        print(dic)
        imprimir_vals(dic)
```

```
{1: 2, 3: 4, 5: 6}
Llave: 1 Valor: 2
Llave: 3 Valor: 4
Llave: 5 Valor: 6
```

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).

```
In [7]: def gen_num(n):
            dic = {}
            for i in range(1, n+1):
                dic[i] = i**2
            return dic

        n = 5
        print('n es:', n)
        gen_num(n)
```

```
n es: 5
```

```
Out[7]: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

1.1.7 7

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

```
In [8]: def sum_llaves(dic):
            suma = 0
            for k in dic:
                suma = suma + k
            return suma

        c = {1 : 2, 2: 3, 7: 4}
        print(c.keys())
        print('Suma llaves:',sum_llaves(c))
```

```
dict_keys([1, 2, 7])
Suma llaves: 10
```

1.1.8 8

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

```
In [9]: def sum_vals(dic):
            suma = 0
            for k in dic:
                suma = suma + dic[k]
            return suma

        c = {1 : 2, 2: 3, 7: 4}
        print(c.values())
        print('Suma valores:',sum_vals(c))
```

```
dict_values([2, 3, 4])
Suma valores: 9
```

1.1.9 9

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

```
In [10]: def sum_item(dic):
             suma = 0
             for k in dic:
                 suma = suma + k + dic[k]
             return suma

         c = {1 : 2, 2: 3, 7: 4}
         print(c.items())
         print('Suma llaves y valores:',sum_item(c))
```

```
dict_items([(1, 2), (2, 3), (7, 4)])
Suma llaves y valores: 19
```

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.)

```
In [11]: def dic_creator(list1, list2):
             dic = {}
             for i in range(0, len(list1)):
                 dic[list1[i]] = list2[i]
             return dic

         print('Lista 1:', [x for x in range(1,6)])
         print('Lista2:', [x for x in range(5,0,-1)])
         dic_creator([x for x in range(1,6)], [x for x in range(5,0,-1)])
```

```
Lista 1: [1, 2, 3, 4, 5]
Lista2: [5, 4, 3, 2, 1]


Out[11]: {1: 5, 2: 4, 3: 3, 4: 2, 5: 1}
```

1.1.11 11

Escriba una función que elimine una llave de un diccionario.