2021-I_mcpp_taller_5_Valentina_Cuenca

March 6, 2021

1 Taller 5

Métodos Computacionales para Políticas Públicas - URosario

Entrega: viernes 12-mar-2021 11:59 PM

Valentina Cuenca norma.cuenca@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.

[163]:
```
import operator
d={"valentina":20,"Nicolas":21,"Camilo": 22, "Miguel":19,"Gloria":40, "Tania":
 ↪15, "Natalia":22}
def ordenar(d,descendente=True):
    """
    d= diccionario a organizar
```

```
    descendente= Su valor es True si queremos que sea de mayor a menor, si es␣
↪False, es de menor a mayor
    """
    if descendente==True:
        do=sorted(d.items(),key=operator.itemgetter(1), reverse=True)
        d={}
        for i in range(len(do)):
            d[do[i][0]]=do[i][1]
    else:
        do=sorted(d.items(),key=operator.itemgetter(1), reverse=False)
        d={}
        for i in range(len(do)):
            d[do[i][0]]=do[i][1]
    return d

d=ordenar(d,False)
d
```

[163]:
```
{'Tania': 15,
 'Miguel': 19,
 'valentina': 20,
 'Nicolas': 21,
 'Camilo': 22,
 'Natalia': 22,
 'Gloria': 40}
```

1.1.2 2

Escriba una función que agregue una llave a un diccionario.

[164]:
```
def agregar(dic,agregar, valor= " "):
    """"
    dic= El diccionario al que le vamos a agregar la llave.
    agregar= La llave que le vamos a agregar al diccionario.
    valor= El valor que se le va a dar, por defecto es vacio.
    """
    dic[agregar]= valor

agregar(d,"Camila")
d
```

[164]:
```
{'Tania': 15,
 'Miguel': 19,
 'valentina': 20,
 'Nicolas': 21,
```

```
 'Camilo': 22,
 'Natalia': 22,
 'Gloria': 40,
 'Camila': ' '}
```

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}

[96]:
```
dicc1 = {1:10, 2:20}
dicc2 = {3:30, 4:40}
dicc3 = {5:50,6:60}
dicc1.update(dicc2)
dicc1.update(dicc3)
dicc1
```

[96]:
```
{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.

[115]:
```
def buscar(dic, llave,valor=False):
    """
    dic= Diccionarios al que vamos a buscar.
    llave=La llave que queremos buscar en el diccionario
    valor= Si queremos ver el valor de esa llave
    """
    if valor== False :
        if llave in dic:
            return True
        else:
            return False
    else:
        if llave in dic:
            return True, dic[llave]
        else:
                return False

buscar(d,"valentina",True)
```

[115]:
```
(True, 20)
```

1.1.5 5

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

[116]:
```
d={"valentina":20,"Nicolas":21,"Camilo": 22, "Miguel":19,"Gloria":40, "Tania":
↪15, "Natalia":17}
def PrintPar(dic):
    y=list(dic.items())
    for i in range(1,len(y),2):
        print(y[i])
PrintPar(d)
```

```
('Nicolas', 21)
('Miguel', 19)
('Tania', 15)
```

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).

[166]:
```
def generarCuadrado(n):
    dic={}
    for i in range(1,n+1):
        dic[i]=i**2
    return dic
diccionario=generarCuadrado(8)
diccionario
```

[166]:
```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64}
```

1.1.7 7

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

[126]:
```
c = {1 : 2, 2: 3}
def sum_llaves(dic):
    l=dic.keys()
    l=list(l)
    s=sum(l)
    return s




sum_llaves(c)
```

[126]:
```
3
```

1.1.8 8

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

[127]:
```
c = {1 : 2, 2: 3}
def sum_valores(dic):
    v=dic.values()
    v=list(v)
    s=sum(v)
    return s




sum_valores(c)
```

[127]: 5

1.1.9 9

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

[129]:
```
d ={2:1, 3:2, 4:5}
def sum_llyv(dic):
    v=sum_valores(dic)
    l=sum_llaves(dic)
    s=l+v
    return s
sum_llyv(d)
```

[129]: 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.)

[167]:
```
Mascota=["Gato","Perro","Camello","Conejo","Tortuga","Pez"]
Cantidad=[2,1,0,2,1,5]
def crear(l1,l2):
    """
    l1,l2: son listas, las cuales son la base para crear el diccionario, la␣
 ↪primera lista son las llaves y l segunda los valores.
    """
    dic={}
    if len(l1)==len(l2):
        for i in range(len(l1)):
            dic[l1[i]]=l2[i]
        return dic
    else:
        print("Las listas deben contener la misma longitud")
```

```
Animales=crear(Mascota,Cantidad)
Animales
```

[167]:
```
{'Gato': 2, 'Perro': 1, 'Camello': 0, 'Conejo': 2, 'Tortuga': 1, 'Pez': 5}
```

1.1.11 11

Escriba una función que elimine una llave de un diccionario.

[168]:
```
d={"valentina":20,"Nicolas":21,"Camilo": 22, "Miguel":19,"Gloria":40, "Tania":
 ↪15, "Natalia":17}
def eliminar(dic,llave):
    """
    dic= Al que se le va a aplicar
    llave= La llave que deseamos eliminar
    """
    nuevo={}
    dic=list(dic.items())
    for l,v in dic:
        if l==llave:
            dic.remove((l,v))
    for i in range(len(dic)):
        nuevo[dic[i][0]]=dic[i][1]
    dic=nuevo
    return(dic)
d=eliminar(d,"valentina")
d
```

[168]:
```
{'Nicolas': 21,
 'Camilo': 22,
 'Miguel': 19,
 'Gloria': 40,
 'Tania': 15,
 'Natalia': 17}
```

1.1.12 12

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

[172]:
```
def Min_Max(dic):
    v=list(dic.values())
    mx=max(v)
    mn=min(v)
    return mn, mx
Min_Max(d)
```

[172]:
```
(15, 40)
```

1.1.13 13

[175]:
```
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))
```

[175]:
```
[5, 5, 3, 5, 4, 4, 3]
```

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

[177]:
```
sentence = "the quick brown fox jumps over the lazy dog"
words = sentence.split()
word_lengths=[len(x) for x in words[1:]]
```

[177]:
```
[5, 5, 3, 5, 4, 3, 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.

[181]:
```
a=[1,3,4,5,6,7,7,6,4]
list(filter(lambda x: x%2==0, a))
```

[181]:
```
[4, 6, 6, 4]
```

1.1.15 15

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

[188]:
```
from functools import reduce
reduce(lambda x,y: x*y,a )
```

[188]:
```
423360
```

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)].

[194]:
```
[ (x,y) for x in range(1,7) for y in range(1,7)]
```

[194]:
```
[(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)]
```
