mcpp_taller5_Daniela_Gaitan

September 8, 2016

1 Taller 5

Métodos Computacionales para Políticas Públicas - URosario

Entrega: viernes 9-sep-2016 11:59 PM

[Daniela Gaitán Cotrino] [daniela.gaitanc@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

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

```
In [1]: def order(dictionary, falling):
            ordenando = sorted(dictionary.values(), reverse = falling)
            return ordenando
```

```
In [2]: clase = {"Daniela": 5, "Juan":2, "Paula":3, "Camila":1, "juan":4}
        #Cuando los valores son enteros

In [3]: clase

Out[3]: {'Camila': 1, 'Daniela': 5, 'Juan': 2, 'Paula': 3, 'juan': 4}

In [4]: order(clase ,False)

Out[4]: [1, 2, 3, 4, 5]

In [5]: order(clase,True)

Out[5]: [5, 4, 3, 2, 1]

In [6]: deportes = {1: "equitación", 2:"natación", 3:"tennis", 4:"golf"}
        # Cuando los valores son strings

In [7]: deportes

Out[7]: {1: 'equitación', 2: 'natación', 3: 'tennis', 4: 'golf'}

In [8]: order(deportes ,False)

Out[8]: ['equitación', 'golf', 'natación', 'tennis']

In [9]: order(deportes,True)

Out[9]: ['tennis', 'natación', 'golf', 'equitación']
```

1.1.2 2

Escriba una función que agregue una llave a un diccionario.

```
In [10]: def adicion(dictionary, llaven, valorllaven):
             dictionary2 = dictionary
             dictionary2[llaven] = valorllaven
             return dictionary2

In [11]: deportes = {1: "equitación", 2:"natación", 3:"tennis", 4:"golf"}

In [12]: adicion(deportes, 5, "baloncesto")

Out[12]: {1: 'equitación', 2: 'natación', 3: 'tennis', 4: 'golf', 5: 'baloncesto'}

In [13]: clase = {"Daniela": 5, "Juan":2, "Paula":3, "Camila":1, "juan":4}

In [14]: adicion(clase, "liliana", 3)

Out[14]: {'Camila': 1, 'Daniela': 5, 'Juan': 2, 'Paula': 3, 'juan': 4, 'liliana': 3
```

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

```
In [15]: dicc1 = {1:10, 2:20}
         dicc2 = {3:30, 4:40}
         dicc3 = {5:50,6:60}
```

```
In [16]: def agregar_dicc(diccionary1, diccionary2):
             diccionary = diccionary1.update(diccionary2)
             return diccionary
```

```
In [17]: agregar_dicc(dicc1, dicc2)
         agregar_dicc(dicc1, dicc3)
```

```
In [18]: dicc1
```

```
Out[18]: {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 [19]: def existencia_dicc(dictionary, llave):
             dictionary= dictionary
             if dictionary.get(llave) == None:
                 print("La llave no existe en el diccionario")
             else:
                 print("La llave existe en el diccionario y su valor es:")
                 return dictionary.get(llave)
```

```
In [20]: deportes = {1: "equitación", 2:"natación", 3:"tennis", 4:"golf"}
         deportes
```

```
Out[20]: {1: 'equitación', 2: 'natación', 3: 'tennis', 4: 'golf'}
```

```
In [21]: existencia_dicc(deportes, 7)
```

```
La llave no existe en el diccionario
```

```
In [22]: existencia_dicc(deportes, 1)
```

```
La llave existe en el diccionario y su valor es:
```

```
Out[22]: 'equitación'
```

```
In [23]: clase = {"Daniela": 5, "Juan":2, "Paula":3, "Camila":1, "juan":4}
```

```
In [24]: existencia_dicc(clase, "Paula")
```

```
La llave existe en el diccionario y su valor es:
```

```
Out[24]: 3
```

1.1.5 5

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

```
In [25]: def pares_dicc(dictionary):
             dictionary = dictionary
             for a, b in dictionary.items():
                 print(a , b)

In [26]: deportes = {1: "equitación", 2:"natación", 3:"tennis", 4:"golf"}

In [27]: pares_dicc(deportes)

1 equitación
2 natación
3 tennis
4 golf


In [28]: clase = {"Daniela": 5, "Juan":2, "Paula":3, "Camila":1, "juan":4}

In [29]: pares_dicc(clase)

Juan 2
Daniela 5
juan 4
Paula 3
Camila 1
```

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). Ejemplo: n = 5 Resultado esperado: 1: 1, 2: 4, 3: 9, 4: 16, 5: 25

```
In [30]: def num_cuadrado(num_final):
             dictionary = {}
             for j in range(1,num_final + 1):
                 dictionary[j] = j ** 2
             return dictionary

In [31]: n = 15
         num_cuadrado(n)

Out[31]: {1: 1,
          2: 4,
          3: 9,
          4: 16,
          5: 25,
          6: 36,
```

```
 7: 49,
 8: 64,
 9: 81,
 10: 100,
 11: 121,
 12: 144,
 13: 169,
 14: 196,
 15: 225}
```

1.1.7 7

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

```
In [32]: def sum_keys_dicc(dictionary):
             from functools import reduce
             dictionary = dictionary
             keys = list(dictionary.keys())
             sumatoria = reduce((lambda x, y:x + y), keys)
             return sumatoria

In [33]: deportes = {1: "equitación", 2:"natación", 3:"tennis", 4:"golf"}

In [34]: sum_keys_dicc(deportes)

Out[34]: 10
```

1.1.8 8

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

```
In [35]: def sum_values_dicc(dictionary):
             from functools import reduce
             dictionary = dictionary
             values = list(dictionary.values())
             sumatoria = reduce((lambda x, y:x + y), values)
             return sumatoria

In [36]: clase = {"Daniela": 5, "Juan":2, "Paula":3, "Camila":1, "juan":4}

In [37]: sum_values_dicc(clase)

Out[37]: 15
```

1.1.9 9

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

```
In [38]: def sum_items_dicc(dictionary):
             from functools import reduce
             dictionary = dictionary

             values = list(dictionary.values())
             sumatoria_values = reduce((lambda x, y:x + y), values)

             keys = list(dictionary.keys())
             sumatoria_keys = reduce((lambda x, y:x + y), keys)

             sumatoria_total = sumatoria_keys + sumatoria_values

             print("La sumatoria de las llaves es: ",sumatoria_keys)
             print("\nLa suma de los valores es: ",sumatoria_values)
             print("\nLa sumatoria total de los items es: ")

             return sumatoria_total

In [39]: numeros = {1: 1,2: 4,3: 9,4: 16,5: 25,6: 36,7: 49,8: 64,9: 81,10: 100,11:

In [40]: sum_items_dicc(numeros)

La sumatoria de las llaves es:  120

La suma de los valores es:  1240

La sumatoria total de los items es:


Out[40]: 1360
```

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 [41]: def mapear(clase, nota):
             return(dict(zip(clase, nota)))


In [42]: clase =["juana", "maria", "viviana"]
         nota = [3, 4, 2]
         mapear(clase, nota )

Out[42]: {'juana': 3, 'maria': 4, 'viviana': 2}
```

1.1.11 11

Escriba una función que elimine una llave de un diccionario.

```
In [43]: def delete(dictionary, key):
             dictionary2 = dictionary
             del dictionary2[key]
             return dictionary2

In [44]: deportes = {1: 'equitación', 2: 'natación', 3: 'tennis', 4: 'golf', 5: 'ba

In [45]: delete(deportes, 4)

Out[45]: {1: 'equitación', 2: 'natación', 3: 'tennis', 5: 'baloncesto'}
```

1.1.12 12

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

```
In [46]: def min_max(dictionary):
             dictionary2 = dictionary
             minimum = min(list(dictionary2.values()))
             maximum = max(list(dictionary2.values()))
             print("El valor mínimo y el valor máximo del diccionario son: ")
             return minimum, maximum

In [47]: numeros = {1: 1,2: 4,3: 9,4: 16,5: 25,6: 36,7: 49,8: 64,9: 81,10: 100,11:

In [48]: min_max(numeros)

El valor mínimo y el valor máximo del diccionario son:


Out[48]: (1, 225)
```

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.

```
In [49]: sentence = "the quick brown fox jumps over the lazy dog"
         words, word_lengths = sentence.split(),[]
         [word_lengths.append(len(word)) for word in words if word != "the"]

Out[49]: [None, None, None, None, None, None, None]

In [50]: sentence

Out[50]: 'the quick brown fox jumps over the lazy dog'
```

```
In [51]: words

Out[51]: ['the', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']

In [52]: word_lengths

Out[52]: [5, 5, 3, 5, 4, 4, 3]
```

1.1.14 14

```
In [53]: a = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

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

```
In [54]: [x for x in a if x%2 == 0]

Out[54]: [4, 16, 36, 64, 100]
```

1.1.15 15

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

```
In [55]: from functools import reduce
         reduce((lambda x,y: x*y), a )

Out[55]: 13168189440000
```

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)].

```
In [56]: caras = [1, 2, 3, 4, 5, 6]

In [57]: combin =[(dado2,dado1) for dado1 in caras for dado2 in caras]

In [58]: len (combin)

Out[58]: 36

In [59]: combin

Out[59]: [(1, 1),
          (2, 1),
          (3, 1),
          (4, 1),
          (5, 1),
          (6, 1),
          (1, 2),
          (2, 2),
```

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