Taller 5

Métodos Computacionales para Políticas Públicas - URosario

Entrega: viernes 8-mar-2019 11:59 PM

Juan Sebastián Valbuena

sebsatian.valbuena256@gmail.com

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 aquí]" 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

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

In [1]:
```
a= {"a":1,"c":4,"b":2}
def main():
    sorted_e = sorted((key, value) for (key,value) in e.items())
    print (sorted_e)
    print(sorted(sorted_e, reverse=True))
main()
```

```
[('a', 1), ('b', 2), ('c', 4)]
[('c', 4), ('b', 2), ('a', 1)]
```

2

Escriba una función que agregue una llave a un diccionario.

In [2]:
```
d = {1: "one", 3: "three"}
def main1():
    print (d)
    d1 = {2: "two"}
    d.update(d1)
    print(d)
main1()
```

```
{1: 'one', 3: 'three'}
{1: 'one', 3: 'three', 2: 'two'}
```

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

In [4]:
```
Resultado_esperado = dicc1.copy()
Resultado_esperado.update(dicc2)
Resultado_esperado.update(dicc3)
print (Resultado_esperado)
```

```
{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 [1]:
```
Colombia = {"nombre":"RepublicadeColombia", "poblacion":49000000}
def main3():
    busca=input("Introduce la búsqueda: ")
    print("Resultado: ", Colombia.get(busca,"No Se Encuentra"))
main3()
```

```
Introduce la búsqueda: nombre
Resultado:  RepublicadeColombia
```

5

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

In [2]:
```
Resultado={1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
def main4():
    for key in Resultado:
        print("{}: {}".format(key, Resultado[key]))
main4()
```

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

6

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

In [5]:
```
n=int(input("Input a number "))
d = dict()
def main5():
    for x in range(1,n+1):
        d[x]=x**x
        print(d)
main5()
```

```
Input a number 3
{1: 1}
{1: 1, 2: 4}
{1: 1, 2: 4, 3: 27}
```

7

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

In [13]:
```
dic = { '1.2' : [1,2,3] , '2.2' : [4,5,6] , '3.3' :[7,8,9]}
new_dic = {}
def main8():
    for key,lis in dic.items():
        new_dic[key] = sum(lis)
        print (new_dic)
main8()
```

```
{'1.2': 6}
{'1.2': 6, '2.2': 15}
{'1.2': 6, '2.2': 15, '3.3': 24}
```

8

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

In [16]:
```
def suma_valores_dic(main_dic):
        diccionario_result = {}
        for key, value in main_dic.items():
            suma = 0
            for v in value:
                suma += v
            diccionario_result[key] = suma
        return diccionario_result
diccionario = {'1': [50, 23, 45, 99], '2': [23, 49,1]}
resultado = suma_valores_dic(diccionario)
print(resultado)
```

```
{'1': 219, '2': 73}
```

9

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

In [22]:
```
Resultado={1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
def main9():
    sum(Resultado.keys())+sum(Resultado.values())
    print (sum(Resultado.keys()))
    print (sum(Resultado.values()))
    print (sum(Resultado.keys())+sum(Resultado.values()))
main9()
```

```
21
210
231
```

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 [25]:
```
keys=[1,2,3,4,5]
values=["a","b","c","d","e"]
def main15():
    dict = {keys[i]: values[i] for i in range(len(keys))}
    print (dict)
main15()
```

```
{1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e'}
```

11

Escriba una función que elimine una llave de un diccionario.

In [33]:
```
Resultado={"uno": 10, "dos": 20, "tres": 30, "cuatro": 40, "cinco": 50, "seis": 60}
def main99():
    result = Resultado.pop("uno", None)
    print("Deleted item's value = ", result)
    print("Updated Dictionary :" , Resultado)
main99()
```

```
Deleted item's value =  10
Updated Dictionary : {'dos': 20, 'tres': 30, 'cuatro': 40, 'cinco': 50, 'seis': 60}
```

12

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

In [27]:
```
def maxmin88():
    dict = {5: 60, 6: 71, 7: 54, 8: 77, 20: 36, 21: 57, 22: 24, 23: 84}
    key_max = max(dict.keys(), key=lambda k: dict[k])
    key_min = min(dict.keys(), key=lambda k: dict[k])
    print (key_min)
    print (key_max)
maxmin88()
```

```
22
23
```

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 [78]:
```
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))
```

In [143]:
```
sentence = "the quick brown fox jumps over the lazy dog"
words = sentence.split()
word_lengths = [len(word) for word in words if word != "the"]
```

14

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

In [92]:
```
listal=[1,2,3,4,5]
pareslistal=[listal for listal in listal if listal % 2 == 0]
print (pareslistal)
```

```
[2, 4]
```

15

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

In [100]:
```
>>> from functools import reduce
reduce(lambda x, y: x*y, [1,2,3,4,5])
```

Out[100]:
```
120
```

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 [119]:
```
combs = reduce(lambda x, y: list(itertools.combinations([1, 2, 3,4,6], y)) + x, range(len([1, 2, 3,4,6])+1), [])
print (combs)
```

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

In [47]:
```
def zipp (liste1,liste2):
    return [(liste1[i], liste2[i]) for i in range(min(len(liste1),len(liste2)))]
dicc= {tuple(liste1[i]): liste2[i] for i in range(length)}
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-47-108908bf11cc> in <module>
      1 def zipp (liste1,liste2):
      2     return [(liste1[i], liste2[i]) for i in range(min(len(liste1),len(liste2)))]
----> 3 dicc= {tuple(liste1[i]): liste2[i] for i in range(length)}

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

In [ ]:
```
roll= list(range(0,13)
for i in range (1,13):
        roll[i]=set()
for i in range (1,7):
```

In [48]:
```
>>> d = {2: ["a", "b"], 3: ["b", "c", "d"], 4: ["a"]}
>>> pairs = [(k, v) for v in d[k]] for k in d]
>>> list(itertools.product(*pairs))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-48-87f63a60e8d4> in <module>
      1 d = {2: ["a", "b"], 3: ["b", "c", "d"], 4: ["a"]}
      2 pairs = [(k, v) for v in d[k]] for k in d]
----> 3 list(itertools.product(*pairs))

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