Taller 5

Métodos Computacionales para Políticas Públicas - URosario

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

Juan Diego Castro Rodríguez]

[juand.castro@urosario.edu.co]

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

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

In [18]:
```
import operator
diccionario = {1: 2, 3: 4, 4: 3, 2: 1, 0: 0, 7: 5, 8: 7, 9: 6}
def orden_as_des(dicc):
    print('Diccionario Original : ',dicc)
    sorted_dicc =dict( sorted(dicc.items(), key=operator.itemgetter(1)))
    print('Diccionario en orden ascendente por valor : ',sorted_dicc)
    sorted_dicc =dict( sorted(dicc.items(), key=operator.itemgetter(1),reverse=True))
    print('Diccionario en orden descendente por valor : ',sorted_dicc)
```

In [19]:
```
orden_as_des(diccionario)
```

```
Diccionario Original :  {1: 2, 3: 4, 4: 3, 2: 1, 0: 0, 7: 5, 8: 7, 9: 6}
Diccionario en orden ascendente por valor :  {0: 0, 2: 1, 1: 2, 4: 3, 3: 4, 7: 5, 9: 6, 8: 7}
Diccionario en orden descendente por valor :  {8: 7, 9: 6, 7: 5, 3: 4, 4: 3, 1: 2, 2: 1, 0: 0}
```

2

Escriba una función que agregue una llave a un diccionario.

In [25]:
```
notas = {"Juan": 4.0, "Susana": 5.0, "Javier": 3.0}
def añadir_llave(dict, key, value):
    print("Diccionario original :", dict)
    dict[key] = value
    print("Diccionario modificado :", dict)
```

In [26]:
```
añadir_llave(notas, "Mateo", 3.1)
```

```
Diccionario original : {'Juan': 4.0, 'Susana': 5.0, 'Javier': 3.0}
Diccionario modificado : {'Juan': 4.0, 'Susana': 5.0, 'Javier': 3.0, 'Mateo': 3.1}
```

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 [33]:
```
dicc1 = {1:10, 2:20}
dicc2 = {3:30, 4:40}
dicc3 = {5:50,6:60}
def con_dic_by_3(tuplas_1, tuplas_2, tuplas_3):
    dict={}
    dict.update(tuplas_1)
    dict.update(tuplas_2)
    dict.update(tuplas_3)
    return dict
```

In [34]:
```
con_dic_by_3(dicc1, dicc2, dicc3)
```

Out[34]:
```
{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 [48]:
```
sexo={"Julian": "M", "Maria": "F", "Esteban": "M"}
def ver_key(dict, key):
    if key in dict.keys():
        print("La llave ", key, " si existe en el diccionario: ", dict)
        """Tener Cuidado con tildes, si la llave que verifica no es exacta a la que llama dara error"""
    else:
        print("La llave ", key, " no existe en el diccionario: ", dict)
```

In [49]:
```
ver_key(sexo, "Maria")
```

```
La llave  Maria  si existe en el diccionario:  {'Julian': 'M', 'Maria': 'F', 'Esteban': 'M'}
```

In [50]:
```
ver_key(sexo, "Maria")
```

```
La llave  Maria  no existe en el diccionario:  {'Julian': 'M', 'Maria': 'F', 'Esteban': 'M'}
```

5

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

In [53]:
```
def imprimir_key_val(dict):
    print(list(dict.items()))
```

In [54]:
```
imprimir_key_val(sexo)
```

```
[('Julian', 'M'), ('Maria', 'F'), ('Esteban', 'M')]
```

6

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

In [57]:
```
def añadir_llave_valor(dict, key, value):
    dict[key] = value
def new_dict_n_square (n):
    dict={}
    for x in range (1,n+1):
        añadir_llave_valor(dict, x, x**2)
    return (dict)
```

In [58]:
```
new_dict_n_square(5)
```

Out[58]:
```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

7

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

In [197]:
```
c = {1 : 2, 2: 3}
sum_llaves(c)
```

```
8
```

In [61]:
```
a={1: 2, 3: 4, 5: 6, 7: 8}
def sum_llaves(dict):
    sumatoria=sum(dict.keys())
    return(sumatoria)
```

In [62]:
```
sum_llaves(a)
```

Out[62]:
```
16
```

8

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

In [63]:
```
a={1: 2, 3: 4, 5: 6, 7: 8}
def sum_valores(dict):
    sumatoria1=sum(dict.values())
    return(sumatoria1)
```

In [64]:
```
sum_valores(a)
```

Out[64]:
```
20
```

9

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

In [189…
```
d ={2:1, 3:2, 4:5}
```

In [69]:
```
def sum_items(dict):
    sumatoria2=sum_valores(dict)+sum_llaves(dict)
    return(sumatoria2)
```

In [70]:
```
sum_items(a)
```

Out[70]:
```
36
```

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 [7]:
```
estudiantes=["Hugo", "Paco", "Luis"]
estudios=["pregrado", "especialización", "maestria"]
def crear_pares():
    nuevodic=dict(zip(estudiantes, estudios))
    print(nuevodic)
```

In [8]:
```
crear_pares()
```

```
{'Hugo': 'pregrado', 'Paco': 'especialización', 'Luis': 'maestria'}
```

11

Escriba una función que elimine una llave de un diccionario.

In [13]:
```
compras={"Carne": "1 Kilo", "Platano": "2 unidades", "Jamón": "media libra"}
def del_key(dict, key):
    if key in dict:
        del dict[key]
        print("Nuevo diccionario :",dict)
    else:
        print("La llave ", key, " no existe en el diccionario :", dict)
```

In [15]:
```
del_key(compras, "Carne")
```

```
Nuevo diccionario :  {'Platano': '2 unidades', 'Jamón': 'media libra'}
```

In [16]:
```
del_key(compras, "Fruta")
```

```
La llave  Fruta  no existe en el diccionario :  {'Platano': '2 unidades', 'Jamón': 'media libra'}
```

12

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

In [19]:
```
numeros={1: 2, 3: 4, 5: 6, 7: 8, 9: 10}
def val_min_max(dict):
    minimo=min(dict.values())
    maximo=max(dict.values())
    print("El valor minimo es ", minimo, " y el valor máximo es ", maximo)
```

In [20]:
```
val_min_max(numeros)
```

```
El valor minimo es  2  y el valor máximo es  10
```

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 [26]:
```
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 [33]:
```
a=[1,2,3,4,5,6,7,8,9,10]
[x for x in a if x%2==0]
```

Out[33]:
```
[2, 4, 6, 8, 10]
```

15

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

In [41]:
```
from functools import reduce
print (reduce(lambda a,b : a*b,a))
```

```
3628800
```

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 [2]:
```
c=[(x,y) for x in range (1,7) for y in range (1,7)]
print(c)
```

```
[(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)]
```
