# Taller 5

Métodos Computacionales para Políticas Públicas - URosario

**Entrega: viernes 6-mar-2020 11:59 PM**

**Santiago Ortiz Ortiz**

[santiago.ortizo@urosario.edu.co (mailto:santiago.ortizo@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

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

In [34]:
```
diccionario = {"andrea": 19, "ramiro": 20, "lina": 16, "jose":10}

forma = input("Ingrese la forma (A/D): ")

def ordenar(forma):
    if forma == "A":
        lista = sorted(diccionario.items(), key = lambda x: x[1], reverse = Fals
        print("Se imprimirá de forma ascendente")
    elif forma == "D":
        lista = sorted(diccionario.items(), key = lambda x: x[1], reverse = True
        print("Se imprimirá de forma descendente")
    for i in lista:
        print(i[0], "->", i[1])

ordenar(forma)
```

```
Ingrese la forma (A/D): A
jose -> 10
lina -> 16
andrea -> 19
ramiro -> 20
```

***Explicación del código:***

El método list("lista".items()) me permite obtener una lista cuyos elementos son tuplas que contienen en la posición 0 la llave y en la posición 1 el valor.

Si utilizamos la función sorted() podemos obtener los valores de forma ascendente. Esta función cuenta con una opción "key =", en la cual uno ingresa el elemento por el cual quiere que se ordene. Note que en la LISTA creada, los valores son cada tupla, por lo tanto yo quiero que coja CADA tupla pero utilice el VALOR, es decir, la posición 1. La opción reverse, organiza de mayor a menor.

## 2

Escriba una función que agregue una llave a un diccionario.

In [36]:
```
# Utilizamos el mismo diccionario creado: nombres y edades
print("Su diccionario inicial es este:", diccionario)
nombre, edad = input("Ingrese el nombre en minúsculas: "), int(input("Ingrese la
# edad = int(input("Ingrese la edad: "))
def agregar(a, b):
    diccionario[a] = b

agregar(nombre, edad)
print("Su diccionario final es este:", diccionario)
```

```
Su diccionario inicial es este: {'andrea': 19, 'ramiro': 20, 'lina': 16, 'jos
e': 10}
Ingrese el nombre en minúsculas: irene
Ingrese la edad: 10
Su diccionario final es este: {'andrea': 19, 'ramiro': 20, 'lina': 16, 'jose':
10, 'irene': 10}
```

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

concatenada = {}

concatenada.update(dicc1)
concatenada.update(dicc2)
concatenada.update(dicc3)
print("Estes es el resultado:", concatenada)
```

```
Estes es el resultado: {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 [42]:
```
# Utilicemos el diccionario "diccionario"
print("Su diccionario es: ",diccionario)
```

```
Su diccionario es:  {'andrea': 19, 'ramiro': 20, 'lina': 16, 'jose': 10, 'iren
e': 10}
```

In [49]:
```
nombre = input("Ingrese el nombre en minúsculas: ")
def verificar(a):
    try:
        edad = diccionario[a]
        print(a, "se encuentra en el diccionario")
    except KeyError:
        print(a, "NO se encuentra en el diccionario")
verificar(nombre)
```

```
Ingrese el nombre en minúsculas: julio
julio NO se encuentra en el diccionario
```

## 5

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

In [52]:
```
def tuplas_diccionario(dictionary):
    lista_tupas = list(dictionary.items())
    for x in range(len(lista_tupas)):
        print("El par (llave,valor) es: ", lista_tupas[x],"\n", "La llave es:",
tuplas_diccionario(diccionario)
```

```
El par (llave,valor) es:  ('andrea', 19)
 La llave es: andrea
 El valor es: 19
El par (llave,valor) es:  ('ramiro', 20)
 La llave es: ramiro
 El valor es: 20
El par (llave,valor) es:  ('lina', 16)
 La llave es: lina
 El valor es: 16
El par (llave,valor) es:  ('jose', 10)
 La llave es: jose
 El valor es: 10
El par (llave,valor) es:  ('irene', 10)
 La llave es: irene
 El valor es: 10
```

## 6

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

In [63]:
```
numero = int(input("Ingrese un número entero: "))
diccionario = {}
def square_dict(numero):
    for i in range(1, numero + 1):
        diccionario[i] = i ** 2
    return diccionario
square_dict(numero)
```

```
Ingrese un número entero: 7
```

Out[63]:
```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49}
```

## 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 [73]:
```
# Llaves y valores
from functools import reduce
c = {1 : 2, 2 : 3}
def sum_llaves(dictionary):
    llaves = list(dictionary.keys())
    suma = reduce(lambda x, y: x + y, llaves)
    return suma
print("La suma es:", sum_llaves(c))
```

```
La suma es: 3
```

## 8

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

In [74]:
```
# Llaves y valores
from functools import reduce
c = {1 : 2, 2 : 3}
def sum_valores(dictionary):
    valores = list(dictionary.values())
    suma = reduce(lambda x, y: x + y, valores)
    return suma
print("La suma es:", sum_valores(c))
```

```
La suma es: 5
```

## 9

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

In [75]:
```
# Llaves y valores
from functools import reduce
c = {1 : 2, 2 : 3}
def sum_llaves(dictionary):
    llaves = list(dictionary.keys())
    valores = list(dictionary.values())
    llaves_valores = llaves + valores
    suma = reduce(lambda x, y: x + y, llaves_valores)
    return suma
print("La suma es:", sum_llaves(c))
```

```
La suma es: 8
```

## 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 [90]:
```
nombres = ["sandra", "margarita", "sofia"]
edad = [34, 67, 2]

dictionary = {}
def diccionario(nombres, edad):
    for i,j in zip(range(len(nombres)), range(len(edad))):
        dictionary[nombres[i]] = edad[j]
    return dictionary
diccionario(nombres, edad)
```

Out[90]:
```
{'sandra': 34, 'margarita': 67, 'sofia': 2}
```

## 11

Escriba una función que elimine una llave de un diccionario.

In [89]:
```
print("Este es el diccionario inicial:", dictionary)
eliminable = input("Elija una llave a eliminar: ")
def eliminar(a):
    dictionary.pop(a)
    return dictionary
print("Se ha eliminado la llave", eliminable)
eliminar(eliminable)
```

```
Este es el diccionario inicial: {'margarita': 67, 'sofia': 2}
Elija una llave a eliminar: sofia
Se ha eliminado la llave sofia
```

Out[89]:
```
{'margarita': 67}
```

## 12

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

In [100]:
```
print("Su diccionario inicial es:", dictionary)
def max_min(diccionario):
    maximo = max(diccionario.keys(), key = lambda i: diccionario[i])
    minimo = min(diccionario.keys(), key = lambda i: diccionario[i])
    return [maximo, minimo]
print(f" {max_min(dictionary)[0]} es el valor máximo con {dictionary[max_min(dic
      f"{max_min(dictionary)[1]} es el valor mínimo con {dictionary[max_min(dict
```

```
Su diccionario inicial es: {'sandra': 34, 'margarita': 67, 'sofia': 2}
 margarita es el valor máximo con 67
 sofia es el valor mínimo con 2
```

## 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 [114]:
```
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 [115]:
```
word_lengths
```

Out[115]:
```
[5, 5, 3, 5, 4, 4, 3]
```

In [121]:
```
sentence = "the quick brown fox jumps over the lazy dog"
words = sentence.split()
word_lengths = [len(word) for word in words if word != "the"]
word_lengths
```

Out[121]:
```
[5, 5, 3, 5, 4, 4, 3]
```

## 14

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

In [125]:
```
a = [1,2,3,4,5,6,7,8,9,10]
a_prima = list(map(lambda x: x,filter(lambda x: x % 2 == 0, a)))
a_prima
```

Out[125]:
```
[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 [131]:
```
reduce(lambda x, y: x*y, a_prima)
```

Out[131]:
```
3840
```

## 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 [135]:
```
[(x,y) for x in range(1,7) for y in range(1,7)]
```

Out[135]:
```
[(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)]
```