Pontificia Universidad Católica de Chile
Escuela de ingeniería
Departamento de Ciencia de la Computación
Primer Semestre del 2014

# IIC1103 Introducción a la Programación

## Control Diccionarios-Listas-Tuplas-búsqueda-Ordenamiento

De manera muy simple se puede manejar información de contactos de una red social, algo similar a twitter. Esto se logra usando un diccionario de *contactos* que tenga como clave el rut de una persona y los seguidores, que serían los valores asociados a esa clave. Ejemplo podríamos ver el siguiente diccionario:

```
contactos = {"5.555.555-5":["2.222.222-2","7.777.777.7","3.333.333-3"],
             "1.111.111-1":["4.444.444-4","5.555.555-5"],
             "3.333.333-3":["1.111.111-1"],
             "7.777.777-7":["3.333.333-3","5.555.555-5","1.111.111-1"]}
```

Observemos que la persona con rut `"5.555.555-5"` tiene 3 seguidores almacenados en una lista `["2.222.222-2","7.777.777.7","3.333.333-3"], y` así el resto claves del diccionario *contactos*.

Para saber los datos de cada persona se cuenta con otro diccionario llamado *personas*, donde la clave está dada por el rut y los valores son una tupla que tiene la información asociada a ese rut que *nombre*, *país* y *edad*, como se muestra a continuación:

```
personas = {"1.111.111-1":("juan","Chile",20),"2.222.222-2":("mike","Cuba",18),
           "3.333.333-3":("betty","Venezuela",25),"4.444.444-4":("milena","Cuba",23),
           "5.555.555-5":("fabian","Chile",34),"6.666.666-6":("teodoro","Argentina",28),
           "7.777.777-7":("josefina","Suiza",31)}
```

Observemos que la persona con clave `"5.555.555-5"` tiene una tupla como valor que contiene sus datos `("fabian","Chile",34)`, y así el resto de las personas.

Dada esta información, tu misión es:

1. Crear la función ***verifica_existencia*** que reciba como parámetros un *rut* y el diccionario *contactos*. La función debe retorne verdadero en caso de que el rut que ingresa a la función esté como clave del diccionario *contactos*, o que retorne falso en caso contrario.

   Solución

   ```
   def verifica_existencia(cont,rut):
       for pers in cont:
           if pers == rut:
               return True
       return False
   ```

2. Crear la función ***muestra_contactos*** que reciba como parámetro un *rut* y los diccionarios *contactos* y *personas*. La función debe desplegar los seguidores que tiene la persona con el rut que se ingresó.
   Si la función recibe el rut 7.777.777-7 y los diccionarios antes mencionados, entonces debe desplegar:

```
josefina  es seguida por betty
josefina  es seguida por fabian
josefina  es seguida por juan
```

   Antes de desplegar, verifica si la persona con el rut que ingresa a la función pertenece a una de las claves del diccionario *contactos*. Para ello puede usar la función ***verifica_existencia*** implementada anteriormente.

   Solución:

```
def muestra_contactos(cont,pers,rut):
    if verifica_existencia(cont,rut):
        for r in cont[rut]:
            print(pers[rut][0]," es seguida de ",pers[r][0])
    else:
        #no se pedia este else
        print("Persona no tiene red social")
```

3. Crear la función ***obtiene_contactos*** que reciba como parámetro un *rut* y los diccionarios *contactos* y *personas*. La función debe retornar una lista con la información completa de cada seguidor de la persona cuyo *rut* ingresó a la función.

   Ejemplo:
   Si la función recibe el rut 5.555.555-5 y los diccionarios antes mencionados, vemos que la lista que contiene a los seguidores del rut 5.555.555-5 es:
   `["2.222.222-2","7.777.777.7","3.333.333-3"]`, por lo que se debe retornar la lista con la información completa de cada unos de los ***rut***s de esa lista:
   `[("mike","Cuba",18), ("josefina","Suiza",31), ("betty","Venezuela",25)]`

   Solución:

```
def obtiene_contactos(cont,pers,rut):
    lista=[]
    for r in cont[rut]:
        lista.append(pers[r])
    return lista
```

4. Crear la función ***ordena_contactos_por_edad*** que reciba como parámetro la lista que genera ***obtiene_contactos*** para poder ordenar esa lista en forma ascendente de acuerdo a la edad de las personas. Esta función retorna la misma lista, pero ordenada.
   Ejemplo:
   Si la lista es `[("mike","Cuba",18),("josefina","Suiza",31),("betty","Venezuela",25)]` entonces la lista ordenada ascendentemente por la edad queda:
   `[("mike","Cuba",18),("betty","Venezuela",25),("josefina","Suiza",31)]`

Solución:

```
def ordena_contactos_por_edad(lista):
    if lista != 0:
        for i in range(0,len(lista)-1):
            posicion = i
            for j in range(posicion+1,len(lista)):
                if lista[posicion][2] > lista[j][2]:
                    posicion = j
            auxiliar = lista[posicion]
            lista[posicion] = lista[i]
            lista[i] = auxiliar
    return lista
```