Fonaments de Programació (104337)
Curs 2020-21 - Examen FINAL (2 de febrer de 2021)

Nom estudiant: **NIU:**

Important: Recordeu que cal donar les millors solucions possibles en cada exercici. A més de funcionar, els procediments i funcions han d'estar ben programats (utilitzant les instruccions més adequades, sense operacions ni variables innecessàries, etc.)

Els exercicis de l'1 al 6 són funcions o procediments genèrics, però els necessitareu per fer l'exercici 7. Per tant, **la implementació feta als exercicis del 1 al 6 ha de ser coherent amb l'exercici 7**.

Exercici 1 (1 punt)

Els fitxers csv (comma-separated values) són un tipus de format de document per a representar dades en forma de taula . Cada columna es separa amb un caràcter, en l'exemple següent (`';'`):

$$ID_1;dada1_1;dada2_1;...;DadaN_1$$
$$ID_2;dada1_2;dada2_2;...;DadaN_2$$

Fer un funció anomenada `csvFile2dic` que rebi com a paràmetre el nom d'un fitxer (string) i el caràcter separador (en l'exemple `';'`) i retorni un diccionari on la clau sigui la primera columna i els valors siguin una llista amb la resta de les columnes. El fitxer estarà format per un conjunt de línies acabades amb un salt de línia (`'\n'`).

Important: Utilitzar les excepcions per controlar que el fitxer es pot obrir. Si es produeixi un error cal fer un `raise` amb l'error `FileNotFoundError` i el missatge `"Fitxer no trobat"`

```
def FileCSV2Dic(file_name,car):
    try:
        fileHandle=open(file_name,"r")
    except:
        raise FileNotFoundError("Fitxer no trobat")
    else:
        dictionary={}
        for line in fileHandle:
            llista=line[:-1].split(car)
            dictionary[llista[0]]=llista[1:]
        fileHandle.close()
        return dictionary
```

Exercici 2 (1 punt)

Fer un procediment anomenat `dic2csvFile` que rebi com a paràmetre el nom d'un fitxer (string), un diccionari i un caràcter. La clau del diccionari és un string i el valor és una llista de strings, per exemple:

{"ID_1":["$dada1_1$",..., "$dadaN_1$"],"ID_2":["$dada1_2$",..., "$dadaN_2$"],...}

La funció ha d'escriure el diccionari al fitxer. Si el fitxer conté dades, el fitxer es sobreescriu. El caràcter passat per paràmetre serà el separador de cada element del diccionari en el fitxer. El fitxer estarà format per un conjunt de línies acabades amb un salt de línia (`'\n'`). Per exemple, en el cas que `';'` fos el caràcter separador, el fitxer tindria la següent forma:

ID_1;$dada1_1$;$dada2_1$;...; $DadaN_1$
ID_2;$dada1_2$;$dada2_2$;...; $DadaN_2$
...

<u>Important</u>: Utilitzar les excepcions per controlar que l'arxiu tingui un problema d'escriptura. En cas que es produeixi un error cal fer un raise amb l'error `IOError`. i el missatge `"No s'ha pogut escriure al fitxer"`.

```
def Dic2FileCSV(file_name,dic,car):
    try:
        fileHandle=open(file_name,"w")
        for k,v in dic.items():
            fileHandle.write(k+car+car.join(v)+"\n")
    except:
        fileHandle.close()
        raise IOError("No es pot escriure al fitxer")
    else:
        fileHandle.close()
```

Exercici 3 (1 punt)

Fer una funció anomenada `get_column` que rebi com a paràmetre un diccionari anomenat `dictionary` i un valor enter anomenat `position` El diccionari està format per una clau de tipus string i com a valor una llista de strings (totes les llistes de la mateixa longitud).

$$\{\text{“ID}_1\text{”:[“str1}_1\text{”,..., “strN}_1\text{”],“ID}_2\text{”:[“str1}_{2}\text{”,..., “strN}_2\text{”],...}\}$$

La funció retornarà un diccionari amb la mateixa clau que `dictionary` i com a valor l'element de la posició `position` de la llista de valors de `dictionary`.

Nota: Podeu suposar que el valor de `position` sempre serà un valor vàlid, és a dir, sempre serà un valor més gran o igual que zero i més petit que la longitud de la llista associada a la clau de `dictionary`.

```
def get_column(dic,columna):
    out_dic={}
    for k,v in dic.items():
        out_dic[k]=v[columna]
    return (out_dic)
```

Exercici 4 (1 punt)

Fer una funció anomenada `conversion2int` que rebi com a paràmetre una llista on els valors són enters en format string (per exemple, `['16','18','23']`). La funció ha de retornar una llista amb els elements transformats a enters (en l'exemple anterior, `[16,18,23]`).

Nota: La SOLUCIÓ només es considerarà completament CORRECTA si no s'utilitza **CAP BUCLE**.

```
def conversion2int(l):
    return list(map(int, l))
```

Exercici 5 (1 punt)

Fer una funció anomenada `summation_values` que rebi com a paràmetre un diccionari anomenat `dictionary`. El diccionari està format per una clau de tipus string i com a valor una llista de números en format string.

La funció retornarà un diccionari amb la mateixa clau que `dictionary` i com a valor el sumatori de la llista associada a la clau. Utilitzeu la funció de l'exercici anterior per transformar la llista de números en format string a valors de tipus enter.

Nota: Si s'utilitza una **dictionary comprehension** es donarà 0.25 punts extra en la pregunta.

```
def summation_values(dic):
    out_dic={k : sum(conversion2int(v)) for k,v in dic.items()}
    return (out_dic)
```

Exercici 6 (1 punt)

Fer una funció anomenada `reverse` que rebi com a paràmetre un diccionari anomenat `dictionary` on la clau és un string i el valor un enter. La funció retornarà un nou diccionari on la clau sigui el valor de `dictionary` i el valor sigui una llista de les claus de `dictionary` que tenien associat aquell valor. En el procés de creació s'haurà de comprovar els casos en que hi hagi duplicitat de claus del nou diccionari i, en aquest cas, s'afegirà a la llista de valors de la clau en el nou diccionari. Per exemple:

`{'ID8':16,'ID7':18,'ID1':16}` → `{16:['ID8','ID1'],18:[ 'ID7']}`

```
def reverse(dic):
    out_dic={}
    for k,v in dic.items():
        if v in out_dic:
            out_dic[v].append(k)
        else:
            out_dic[v]=[k]
    return(out_dic)
```

Exercici 7 (2 punts)

Volem gestionar els temps d'una carrera ciclista per etapes. En un fitxer anomenat `temps.txt` es guarda el dorsal i el temps (en minuts) de cada etapa, separats per comes. Cada línia correspon a un corredor i acaba amb un salt de línia. Per exemple, si s'han disputat 5 etapes, el fitxer seria:

```
D01,124,210,125,179,174
D02,114,211,135,169,166
D03,110,213,155,159,178
...
```

També tenim un altre arxiu anomenat `equip.txt` on es guarda el dorsal, el nom i l'equip, separats per comes. Cada línia correspon a un corredor i acaba amb un salt de línia. Per exemple:

```
D01,Ramon Pamplina,EquipA
D02,Manolo Aixeca,EquipA
D10,Billy Borrego,EquipZ
...
```

Fer un programa que segueixi els passos que teniu a continuació. **Utilitzeu les funcions i procediments dels exercicis anteriors.**

1. Inicialitzacions de variables i constants.
2. Fer la lectura del fitxer `temps.txt` amb la informació dels participants i el temps per a cada etapa, i la lectura de l'arxiu `equip.txt` amb la informació de cada participant. Capturar si hi ha hagut algun error (`FileNotFoundError`) i, en aquest cas, escriure un missatge d'error i acabar el programa. En cas contrari, seguir amb el punt 3.
3. Utilitzar (NO IMPLEMENTAR) el procediment `menu_principal()` que permet imprimir per pantalla el següent menú:

```
---- MENU ----
1.- Introduir temps nova etapa
2.- Conèixer el temps etapa
3.- Temps total participant
4.- Classificació
5.- Finalitzar
```

4. Si l'opció és 1, el programa preguntarà el temps realitzat per cada dorsal i aquest temps s'afegirà al final de la llista de temps del dorsal en el diccionari corresponent.
5. Si l'opció és 2, el programa preguntarà l'etapa sobre la que es vol conèixer els temps. Si el número d'etapa és correcte, imprimirà línia per línia el dorsal i el temps emprat per cada corredor en l'etapa demanada. En el cas que l'etapa no existeixi (valor incorrecte) s'escriurà el següent missatge "`Etapa incorrecta`".
6. Si l'opció és 3, el programa demanarà un dorsal. Comprovarà que el dorsal existeixi i, si existeix, imprimirà el nom del corredor, l'equip i el temps total emprat pel corredor. Si el dorsal no existeix, retornarà el missatge "`Dorsal inexistent`".
7. Si l'opció és 4, s'imprimirà el temps total i el nom dels participants que han aconseguit aquest temps. El format d'impressió serà:

   `<temps> "minuts ->" <nom_corredor> <nom_corredor> ....`

   Exemple:

   ```
   134 minuts -> Ramon Pamplina Billy Borrego
   153 minuts -> Manolo Aixeca
   ```

   Nota: En aquesta opció s'ha d'utilitzar la funció `reverse` (ex. 6).
8. Si l'opció és 5, mostrar el missatge:" `Sortint...`" i guardar les dades del diccionari de dorsal i temps de les etapes al fitxer `temps.txt`. Capturar si hi hagut algun error (`IOError`) i , en aquest cas, escriure el missatge d'error que li retorni la funció d'escriptura i acabar el programa.
9. Per qualsevol altra opció, es mostrarà el missatge: `Opció no permesa`.
10. Repetir els passos 3 a 9, fins que l'opció de menú escollida sigui la 5.

```
FITXER_TEMPS="temps.txt"
FITXER_EQUIP="equip.txt"
CAR_SEP=','
try:
    dict_Temps = FileCSV2Dic(FITXER_TEMPS,CAR_SEP)
    dict_Equips = FileCSV2Dic(FITXER_EQUIP,CAR_SEP)
except FileNotFoundError as missatge:
    print("Error:",missatge)
else:
    final = False
    while not final:
        menu_principal()
        op=input("Introdueix opció: ")
        if op == "1":
            for c in dict_Temps:
                missatge = "Introdueix temps de "+ c +": "
                t = input(missatge)
                dict_Temps[c].append(t)
        elif op == "2":
            etapa = int (input("Introdueix etapa: "))
            k=list(dict_Temps.keys())
            if etapa>0 and etapa<=len(dict_Temps[k[0]]):
                columna = get_column(dict_Temps,etapa-1)
                for k,v in columna.items():
                    print(k,"->",v)
            else:
                print("Etapa incorrecta")
        elif op =="3":
            dorsal = input("Introduiex dorsal: ")
            if dorsal in dict_Temps:
                temps = sum(conversion2int(dict_Temps[dorsal]))
                print(dict_Equips[dorsal][0], dict_Equips[dorsal][1],temps)
            else:
                print("Dorsal inexistent")
        elif op =="4":
            temps = summation_values(dict_Temps)
            r_temps = reverse(temps)
            for t,corredors in r_temps.items():
                missatge=str(t)+" minuts ->"
                for c in corredors:
                    missatge+=dict_Equips[c][0]+" "
                print(missatge)
        elif op == "5":
            final = True
            try:
                Dic2FileCSV(FITXER_TEMPS,dict_Temps,CAR_SEP)
            except IOError as missatge:
                print("Error:",missatge)
        else:
            print("Opció no permesa")
```

Exercici 8 (0.5 punts)
Fer una funció anomenada multiples_3_5 que rebi com a paràmetre una llista d'enters i **utilitzi una list comprehension** per obtenir una altra llista amb només els valors que són múltiples de 3 o de 5 de la llista passada com a paràmetre. La funció ha de retornar la llista obtinguda.

Nota: La pregunta NOMÉS serà CORRECTA si s'implementa amb una LIST COMPREHENSION.

```
def multiples_3_5(llista):
    return [x for x in llista if x%3==0 or x%5==0]
```

Exercici 9 (1.5 punts)

Definir la classe `Corredor` que tindrà com a atributs el dorsal (`enter`), el nom del corredor (`string`), l'equip del corredor (`string`) i una llista amb els temps emprats en totes les etapes de la carrera.

Definiu les funcions membre:

- `__init__` En el cas que no és passi la llista, es crearà una llista buida.
- `__str__` per poder fer print's dels objectes de tipus `corredor` amb el format:

  `<dorsal> <nom> (<equip>) <temps_total>`
- l'operador __ge__ (greater or equal >=) que ens permetrà comparar dos corredors per saber quin corredor és ràpid. Direm que un corredor és "més gran o igual" que un altre si el **temps total és més gran o igual** que el temps total de l'altre.
- `afegir_etapa` que rep un enter `temps_etapa` i ens permet afegir el temps d'una nova etapa al final de la llista dels temps de les etapes.
- `get_temps_etapa` que rep un enter `num_etapa` i retorna el temps realitzat en una determinada etapa. En cas que `num_etapa` no no sigui un valor vàlid, generar un error del tipus `IndexError` amb el missatge d'error "`L'etapa <num_etapa> no existeix`".

```
class Corredor():
    def __init__(self,dorsal,nom,equip,temps_etapes=None):
        self.dorsal=dorsal
        self.nom=nom
        self.equip=equip
        if temps_etapes==None:
            self.temps_etapes=[]
        else:
            self.temps_etapes=temps_etapes

    def __str__(self):
        out = str(self.dorsal)+" "+self.nom+" ("+self.equip+") "+str(sum(self.temps_etapes))
        return out

    def __ge__(self,other):
        temps_self = sum(self.temps_etapes)
        temps_other = sum(other.temps_etapes)
        return  temps_self >= temps_other

    def afegir_etapa(self,temps_etapa):
            self.temps_etapes.append(temps_etapa)

    def get_temps_etapa(self,num_etapa):
        try:
           temps = self.temps_etapes[num_etapa]
        except IndexError:
            raise IndexError("L'etapa "+str(num_etapa)+" no existeix")
        else:
            return temps
```