Iniciació a la Programació (104383)
Curs 2019-2020 - Examen Final (20 de gener de 2020)

Nom estudiant: **NIU:**

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

Els exercicis del 1 al 7 són funcions o procediments genèrics, però els necessitareu per a fer l'exercici 8. Per tant, **la implementació que feu dels exercicis 1 a 7 ha de ser coherent amb l'exercici 8**.

Exercici 1 (1 punt)
Fer una funció anomenada `file2list` que rebi com a paràmetre el nom d'un fitxer (string) i retorni una llista on cada element sigui una línia del fitxer. 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. En el cas que es produeixi un error cal fer un raise amb l'error `FileNotFoundError`.

```
def file2list(file_name):
    try:
        fileHandle=open(file_name,"r")

    except:
        raise FileNotFoundError("Fitxer no trobat")
    else:
        llista=[]
        for line in fileHandle:
            llista.append(line[:-1])
        fileHandle.close()
        return(llista)
```

Exercici 2 (1 punt)
Fer una funció anomenada `extract_element` que rebi com a paràmetre un string anomenat `phrase` i un valor enter anomenat `position`. La funció retornarà un string que serà un substring de `phrase` i que es troba entre els caràcters blancs `position` i `position+1`. Per exemple:

`extract_element('Hola com estàs?',1)` retornarà `'com'`

Nota: Cal comprovar que `position` no estigui fora de rang, és a dir, que no sigui negatiu ni més gran que el nombre de paraules que tingui `phrase`. En aquest cas utilitzeu algun mètode per la gestió d'error, amb el missatge `"Position fora de rang"`.

```
def extract_element (phrase,position):
    llista = phrase.split()
    long=len(llista)
    assert (position>=0 and position<long),"Position fora de rang"
    return (llista[position])
```

ó

```
def extract_element (phrase,position):
    llista = phrase.split()
    long=len(llista)
    if (position>=0 and position<long):
        return (llista[position])
    else:
        raise IndexError("Position fora de rang")
```

Exercici 3 (0,5 punts)
Fer un procediment anomenat `histogram` que rebi com a paràmetre un diccionari anomenat `dictionary` i un valor que anomenarem `key`. El procediment comprovarà que `key` sigui una clau del diccionari. En el cas que no sigui així, s'afegirà `key` al diccionari i s'assignarà 1 com a valor. En cas que `key` ja existeixi, s'augmentarà en 1 el valor associat a `key`.

```
def histogram(dictionary,key):
    if key in dictionary:
        dictionary[key]+=1
    else:
        dictionary[key]=1
```

Exercici 4 (0,5 punts)
Fer un procediment anomenat `add_dictionary` que rebi com a paràmetre un diccionari anomenat `dictionary` i dos valors que anomenarem `key` i `value` El procediment comprovarà que `key` sigui una clau del diccionari. En el cas que no sigui així, s'afegirà `key` al diccionari i es crearà un conjunt amb un únic element que serà `value`. En cas que `key` ja existeixi, s'afegirà value al conjunt associat a `key`.

Nota: Si utilitzeu llistes en lloc de conjunts (com a valor associat a les claus del diccionari) haureu de assegurar-vos que no hi hagi elements repetits a les llistes.

```
def add_dictionary (dictionary,key,value):
    if key in dictionary:
        if value not in dictionary[key]:
            dictionary[key].append(value)
    else:
        dictionary[key]=[value]
```

Exercici 5 (0,5 punts)
Fer una funció anomenada `maximum_value` que rebi com a paràmetre un diccionari. Els valors associats a les claus del diccionari seran valors enters. La funció haurà de retornar el valor màxim de tots els valors del diccionari.

```
def maximum_value (dictionary):
    llista_max=dictionary.values()
    m=max(llista_max)
    return(m)
```

Exercici 6 (0,5 punts)
Fer una funció anomenada `equal_value` que rebi com a paràmetre un diccionari i un valor enter anomenat `value`. Els valors associats a les claus del diccionari seran valors enters. La funció haurà de retornar una llista amb les claus que tinguin associat el valor igual a `value`.

```
def equal_value(dictionary,value):
    llista=[]
    for k,v in dictionary.items():
        if v==value:
            llista.append(k)
    return (llista)
```

´0

```
def equal_value(dictionary,value):
    return([k for k in dictionary.keys if dictionary[k]==value])
```

Exercici 7 (1 punt)
Fer una funció anomenada `intersection` que rebi com a paràmetres dues llistes i retorni una llista que sigui la intersecció de les dues llistes passades com a paràmetres d'entrada.

Nota: La SOLUCIÓ sols es considerarà com a CORRECTA si està feta utilitzant una list comprehension.

```
def intersection(lst1, lst2):
    lst3 = [value for value in lst1 if value in lst2]
    return lst3
```

Exercici 8 (3 punts)
Volem gestionar un fitxer de log[1] on es guarden intents d'accés fallits a un determinat servidor. La informació que es guarda té el següent format amb un salt de línia al final de cada línia:

```
From 158.109.10.234 stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008
From 123.23.10.214 gsilver@umich.edu Sat Jan 5 19:23:46 2019
...
```

Aquesta informació correspon a l'adreça IP[2] des d'on s'ha fet l'intent d'accés, el compte de mail al que s'ha intentat accedir i la data/hora de l'intent d'accés.

També disposem d'un fitxer amb adreces IP de servidors responsables d'atacs informàtics (una IP per línia).

Fer un programa que segueixi els passos que teniu a continuació. **Per a fer els passos indicats, necessiteu les funcions i procediments dels exercicis anteriors.**

1. Inicialitzacions de variables i constants.
2. Fer la lectura del fitxer `log.txt` amb la informació dels intents d'accés al servidor, i la lectura del fitxer `llista_negra.txt` amb les IP's de servidors responsables d'atacs informàtics. Capturar si hi hagut algun error (`FileNotFoundError`), i en aquest cas escriure un missatge d'error (`"Fitxer no trobat"`) 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.- Direct Attack
2.- Spoofing
3.- Phishing
4.- Temporal Attack
5.- Finalitzar
```

4. Si l'opció és 1, s'han de mostrar per pantalla les IPs que han fet algun intent d'accés fallit i que estan a la llista negra de servidors.
5. Si l'opció és 2 (Spoofing és quan s'intenta suplantar una IP), volem saber si una IP existeix en el fitxer de log.
   5.1. S'ha de demanar a l'usuari que introdueixi l'adreça IP que vol comprovar.
   5.2. En el cas que la IP estigui en el fitxer de log es retornarà el següent missatge `"S'ha intentat suplantar la IP"`.
   5.3. En cas contrari el missatge serà: `"No hi ha cap intent de suplantar la IP"`.
6. Si l'opció és 3 (Phishing és quan s'intenta suplantar la identitat), volem imprimir per pantalla la llista de emails que han rebut més intents de suplantació, és a dir, els emails que apareixen més vegades a fitxer de log (n'hi pot haver més d'un)
7. Si l'opció és 4, volem imprimir per pantalla la relació d'adreces IP que han atacat el servidor per cada dia del log. No cal ordenar per data. El format del text a imprimir ha de ser:

```
5 / Jan / 2019 --> 45.109.10.234 158.109.10.234 ...
25 / Dec / 2018 --> 345.46.87.235 23.234.22.132 ...
...
```

8. Si l'opció és 5, mostrar el missatge: `"Sortint del programa..."` i acabar.
9. Qualsevol altra opció, escriure el missatge: `Opció no permesa`.
10. Repetir els passos 3 a 9, fins que l'opció del menú escollida sigui la 5.

---

[1] Un fitxer de log és un fitxer on es guarda informació històrica d'un determinat esdeveniment en un sistema informàtic.

[2] Una adreça IP és una seqüència de 4 valors entre 0 i 255 separats per un punt que identifiquen una màquina connectada a la xarxa.

```
def main():
    try:
        log=file2list("log.txt")
        ip_black=file2list("llista_negra.txt")
    except FileNotFoundError:
        print("ERR")
    else:

        fi=False
        while(not fi):
            menu_principal()
            opcio=int(input())
            if(opcio==1):
                ip_log =[]
                for register in log:
                    ip_log.append(extract_element(register,1))
                ip_intersection = intersection(ip_log,ip_black)
                print(ip_intersection)

            elif(opcio==2):
                ip=input("Introdueix IP: ")
                trobat=False
                i=0
                while (i<len(log) and not trobat):
                    if (ip in log[i]):
                        trobat=True
                    else:
                        i+=1
                if(trobat):
                    print("S'ha intentat suplantar la IP")
                else:
                    print("No s'ha produït cap intent de suplantar la IP")
```

ó

```
def main():
    try:
        log=file2list("log.txt")
        ip_black=file2list("llista_negra.txt")
    except FileNotFoundError:
        print("ERR")
    else:

        fi=False
        while(not fi):
            menu_principal()
            opcio=int(input())
            if(opcio==1):
                ip_log =[extract_element(register,1) for register in log]
                ip_intersection = intersection(ip_log,ip_black)
                print(ip_intersection)

            elif(opcio==2):
                ip=input("Introdueix IP: ")
                ip_log =[extract_element(register,1) for register in log]
                if(ip in ip_log):
                    print("S'ha intentat suplantar la IP")
                else:
                    print("No s'ha produït cap intent de suplantar la IP")
```

```
            elif(opcio==3):
                email_dic={}
                for register in log:
                    histogram(email_dic,extract_element(register,2))
                maxim =maximum_value(email_dic)
                email_list=equal_value(email_dic,maxim)
                print(email_list)

            elif(opcio==4):
                date_dic={}
                for register in log:
                    dia=extract_element(register,5)
                    mes=extract_element(register,4)
                    ano=extract_element(register,7)
                    data=dia+"/"+mes+"/"+ano
                    add_dictionary(date_dic,data,extract_element(register,1))
                for k,v in date_dic.items():
                    out=k + " --> "
                    for ip in v:
                        out+=ip+" "
                    print(out)
            elif(opcio==5):
                fi=True
                print("Sortint del programa...")
            else:
                print("Opció no permesa")
```

Exercici 9 (1 punt)
Definir la classe `Pais`, que tindrà com a atributs el nom del país, la seva superfície i una llista dels ministres del seu govern.

Definiu la funció membre de inicialització i les següents funcions membre:

- `__str__` per tal de poder fer print's dels objectes de tipus `Pais` amb el format:

  `<nom país> (sup: <superfície>) Govern: <ministre_1>,..., <ministreN>`
- l'operador `__ge__` (greater or equal >=) que ens permetrà comparar dos països. Direm que una país és "més gran o igual" que un altre si la superfície del país es més gran o igual que la de l'altre.
- `PushMinister(m)` que ens permetrà afegir un nou ministre `m` a la llista de ministres.
- `PopMinister(m)` per eliminar el ministre `m` de la llista. Cal controlar que l'element que es vol eliminar estigui a la llista.

  El paràmetre `m` de `PushMinister` i `PopMinister` és un string amb el nom d'un ministre.

```
class Pais():
    def __init__(self,Nom,Superficie,llista_Ministres):
        self.Nom=Nom
        self.Superficie=Superficie
        self.llista_Ministres=llista_Ministres
    def __str__(self):
        mis=self.Nom
        mis += "(sup: " + str(self.Superficie) + ") "
        mis += " Govern: "+ ", ".join(self.llista_Ministres)
        return(mis)
    def __ge__(self,other):
        return(self.Superficie>=other.Superficie)
    def PushMinister(self,Ministre):
        self.llista_Ministres.append(Ministre)
    def PopMinister(self,Ministre):
        if Ministre in self.llista_Ministres:
            self.llista_Ministres.remove(Ministre)
        else:
            print("Err")
```

Exercici 10 (1 punt)
Fer una funció que calculi l'equivalència d'una xifra segons en anys de cadascun dels diferents planetes del sistema solar (Mercuri, Venus, Terra, Mart, Júpiter, Saturn, Urà, Neptú). La funció rebrà com a paràmetres una xifra en segons i una llista amb les equivalències dels anys de cada planeta respecte l'any terrestre, i retornarà una llista amb els anys als que equival la xifra en segons en cadascun dels diferents planetes del sistema solar.

Un any a la terra (365.25 dies) correspon a 31.557.600 segons. Les dades per la resta de planetes són:

Mercuri: 1 any = 0.24 anys terrestres
Venus: 1 any = 0.62 anys terrestres
Mart: 1 any = 1.88 anys terrestres
Júpiter: 1 any = 11.86 anys terrestres
Saturn: 1 any = 29.45 anys terrestres
Urà: 1 any = 84.02 anys terrestres
Neptú: 1 any = 164.79 anys terrestres

Llavors, si partim d'una xifra de 1.000.000.000 de segons terrestres i de la llista d'equivalències [0.24, 0.62, 1, 1.88, 11.86, 29.45, 84.02, 164.79], la funció hauria de retornar la llista:

`[132.03370, 51.10982, 31.68808, 16.85537, 2.67185, 1.07600, 0.37715, 0.19229]`

Això vol dir que 1.000.000.000 segons equivalen a 132 anys a Mercuri, 51 anys a Venus, etc.

```
equiv=(0.24, 0.62, 1, 1.88, 11.86, 29.45, 84.02, 164.79)

def conversio(segons,llista):
    return list(map(lambda x: segons/31557600/x,llista))


res=conversio(1000000000,equiv)
print(res)
```