Faculté des Sciences & Techniques — Université de Limoges

Master 1ère année

Protocoles & Prog. Réseau

Corrections TD n°1

Maîtrise de Python

Manipulation de fichiers

1 – Écrire un programme qui compte le nombre de lignes d'un fichier sur disque.

```
#!/usr/bin/python3

import sys

try:
    fichier = open("td1_exo1.py","r")
except Exception as e:
    print (e.args)
    sys.exit(1)

nb_lignes = 0

while 1:
    ligne = fichier.readline()
    if not ligne:
        break
    nb_lignes += 1

print ("Nb Lignes",nb_lignes)
```

2 – Écrire un programme qui ouvre un premier fichier et crée un nouveau fichier contenant une ligne sur deux du premier fichier.

```
#!/usr/bin/python3

import sys

try:
    entree = open("td1_exo1.py","r")
    sortie = open("td1_exo1.py_recopie","w")
except Exception as e:
    print (e.args)
    sys.exit(1)

while 1:
    ligne = entree.readline()
    if not ligne:
        break
    sortie.write(ligne)
    ligne = entree.readline()
entree.close()
sortie.close()
```

Gestion des listes

3 – Écrire un programme prenant la liste des fichiers contenus dans un répertoire, et qui ouvre et affiche la première ligne de chacun de ces fichiers.

```
#!/usr/bin/python3

import sys, subprocess

résultat = subprocess.run('ls *.py', shell=True, stdout=subprocess.PIPE)
liste_fichiers_ls = résultat.stdout

liste_fichiers = liste_fichiers_ls.splitlines()

for nom_fichier in liste_fichiers:
    try:
        entrée = open(nom_fichier,'r')
    except Exception as e:
        print(e.args)
        continue

    ligne = entrée.readline()
    if not ligne:
        continue
    print(nom_fichier,' ', ligne, end='')
    entrée.close()
```

4 – Écrire un programme qui réalise l'insertion d'une liste d'éléments dans une liste existante à un emplacement donné par son indice.

```
#!/usr/bin/python3

liste_a = [1,2,3,4,5]
liste_b = ['a','b','c']

indice = int(input("Donnez l'indice :"))

assert(indice<len(liste_a))
nouvelle_liste= liste_a[:indice]+liste_b+liste_a[indice:]
print (liste_a,liste_b)
print (nouvelle_liste)
```

Utilisation des expressions rationnelles ou *Regular Expression* & Connexion TCP

5 –

```
#!/usr/bin/python3

import os,socket,sys
import re

re_adresse_electronique = re.compile(r'ACCESS:\s*([a-z0-9\.\-]+)@[a-z0-9\.\-]+')

numero_port = 6688
ma_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM, socket.IPPROTO_TCP)
ma_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR,1)
ma_socket.bind(('', numero_port))
ma_socket.listen(socket.SOMAXCONN)
while 1:
    (nouvelle_connexion, TSAP_depuis) = ma_socket.accept()
    print ("Nouvelle connexion depuis ", TSAP_depuis)
    texte = str(nouvelle_connexion.recv(1000))
    print (">",texte)
    resultat = re_adresse_electronique.search(texte)
    if resultat :
        print ("trouve !")
        print ("Acces de ",resultat.group(1))
    nouvelle_connexion.close()
ma_socket.close()
```

6 – Écrire un programme réalisant du « *banner grabbing* », c-à-d de la capture de bannière d'accueil :

```
#!/usr/bin/python3

import os,socket,sys

liste_services = [ ('smtp.unilim.fr',25),('imap.unilim.fr',995)]

for un_service in liste_services:
    (nom_serveur,port_serveur) = un_service
    adresse_serveur = socket.gethostbyname(nom_serveur)
    ma_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    try:
        ma_socket.connect((adresse_serveur, port_serveur))
    except Exception as e:
        print ("Probleme de connexion", e.args)
        continue
    ligne = ma_socket.recv(1024) #reception d'une ligne d'au plus 1024 caracteres
    if  ligne:
        print ("Banniere: ",ligne)
    ma_socket.close()
```

7 – Écrire un programme qui récupère le titre d'une page html dans un fichier au format HTML.

```
#!/usr/bin/python3

import sys,re

re_debut_titre = re.compile(r'<title>(.*)$',re.I)
re_fin_titre = re.compile(r'^(.*)</title>',re.I)
titre = ""

try:
    fichier = open("page.html","r")
except Exception as e:
    print(e.args)
    sys.exit(1)

while 1:
    ligne = fichier.readline()#.rstrip('\n')
    if not ligne:
        break
```

```
    ligne = ligne.rstrip('\n')
    resultat = re_debut_titre.search(ligne)
    if resultat :
        ligne = resultat.group(1)
        while 1:
            resultat = re_fin_titre.search(ligne)
            if resultat :
                titre += resultat.group(1)
                break
            titre += ligne
            ligne = fichier.readline().rstrip('\n')
            if not ligne :
                break
        break
print (titre)
```

8 – Écrire un motif pour la décomposition d'une URL (récupération des différents champs qui la compose) :

```
#!/usr/bin/python3
import re

#re_decompose_url = re.compile(r'([^:]+)://([a-zA-Z0-9\.\-]+)(?::(\d+))?/(.*)$')
re_decompose_url_avec_port = re.compile(r'([^:]+)://([a-zA-Z0-9\.\-]+):(\d+)/(.*)$')
re_decompose_url_sans_port = re.compile(r'([^:]+)://([a-zA-Z0-9\.\-]+)/(.*)$')
url1 = "http://mon-adresse:789/mon_rep/ma_ressource"
url2 = "http://mon-adresse/mon_rep/ma_ressource"

resultat = re_decompose_url_avec_port.search(url1)

if resultat:
    print ("avec port",resultat.groups())
resultat = re_decompose_url_sans_port.search(url1)
if resultat :
    print ("sans port",resultat.groups())
```

9 – Écrire un programme affichant le contenu d'une page HTML récupérée à l'aide d'une connexion TCP.

```
#!/usr/bin/python3

import os,socket,sys

nom_serveur = "www.unilim.fr"
port_serveur = 80

adresse_serveur = socket.gethostbyname(nom_serveur)
ma_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
try:
    ma_socket.connect((adresse_serveur, port_serveur))
except Exception as e:
    print ("Probleme de connexion", e.args)
    sys.exit(1)
requete = b"GET / HTTP/1.0\r\nHost: www.unilim.fr\r\n\r\n"
ma_socket.sendall(requete)
while 1:
    ligne = ma_socket.recv(1024) #reception d'une ligne d'au plus 1024 caracteres
    if  not ligne:
        break
    print (str(ligne))
ma_socket.close()
```

Manipulation des dictionnaires & opérations d'éclatement et de recomposition (split & join)

10 – Écrire un programme qui détermine le nombre d'occurrence de chaque mot d'un fichier texte.

```
#!/usr/bin/python3
# coding= utf8

import re,sys

# valeurs par défaut
nom_fichier_defaut = "td1_exo10.py"

# programme
saisie = input("Nom du fichier à traiter : [%s]"%nom_fichier_defaut)
nom_fichier = saisie or nom_fichier_defaut

try:
    entree = open(nom_fichier,"r")
except Exception as e:
    print (e.args)
    sys.exit(1)

dico = {}
re_separateurs = re.compile(r"[ ,.;:()?!]+")

while True:
    ligne = entree.readline()
    if not ligne:
        break
    ligne = ligne.rstrip('\n')
    les_mots = re_separateurs.split(ligne)
    for un_mot in les_mots:
        if un_mot in dico:
            dico[un_mot] += 1
            continue
        dico[un_mot]=1

les_cles = list(dico.keys())
les_cles.sort()
for une_cle in les_cles:
    print (une_cle,":",dico[une_cle])

entree.close()
```

Représentation hexadécimale

11 – Écrire un programme qui reproduit le traitement de la commande « `xxd` » du shell :

```
#!/usr/bin/python3
import sys
nom_fichier = 'xxd.py'
try:
    f = open(nom_fichier,'rb')
except Exception as e:
    print(e.args)
    sys.exit(1)

décalage = 0
ligne = b''
while 1:
    if len(ligne) == 0:
        print(format(décalage,'04X'),': ',end='')
    car = f.read(1)
    if not car:
        break
    if (car[0]<127) and (car[0]>31):
        ligne += car
    else :
        ligne += b'.'
    print(format(car[0],'02X'),' ',end='')
    if len(ligne) == 16:
        print(' ',str(ligne,encoding='utf8'))
        ligne = b''
    décalage += 1

if len(ligne):
    print('    '*(16-len(ligne)),end='')
    print(' ', str(ligne,encoding='utf8'))
f.close()
```

Mise en œuvre des instructions de manipulation binaire

12 – Inversion des bits de rang 3 & 4 :

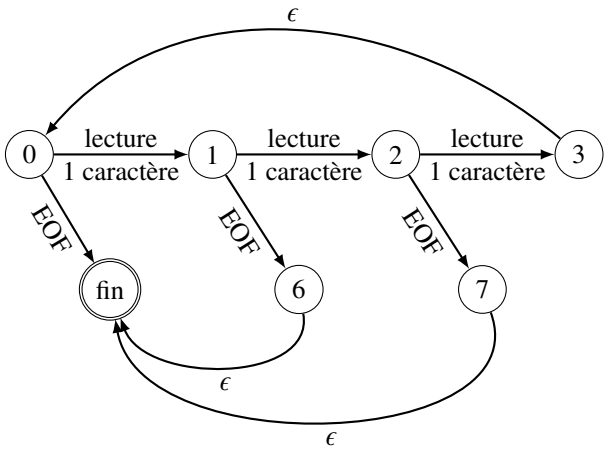
```
#!/usr/bin/python3

import re,sys

try:
    entree = open("exo12.py","rb")
    sortie = open("exo12.py_inverse","bw")
except Exception as e:
    print(e.args)
    sys.exit(1)

while 1:
    caractere = entree.read(1)
    if not caractere:
        break
    rep_binaire = format(caractere[0],'08b')
    rep_inverse = rep_binaire[:3]+rep_binaire[4]+rep_binaire[3]+rep_binaire[5:]
    print(rep_binaire,'->',rep_inverse)
    caractere_inverse = bytes([int(rep_inverse,2)])
    sortie.write(caractere_inverse)
entree.close()
sortie.close()
```

13 – Générateur à congruence linéaire :

```
#!/usr/bin/python3

graine = 5
octet_courant = 8
a = 6364136223846793005
m = 2**64
b = 1442695040888963407
xn = (a*graine+b) % m

def gcl():
    global xn,octet_courant
    if (octet_courant == 8):
        xn = (a*xn+b)%m
        octet_courant= 0
    rep_hexa_gcl = format(xn,'016X')
    rep_hexa_valeur = rep_hexa_gcl[octet_courant*2:octet_courant*2+2]
    octet_courant += 1
    print(rep_hexa_gcl," [",octet_courant,"] -> ",rep_hexa_valeur)
    return int(rep_hexa_valeur,16)

message = b"bonjour a tous"
chiffre = ''
for c in message:
    car_chiffre = gcl() ^ c
    chiffre += format(car_chiffre,'02X')
print(chiffre)

xn = (a*graine+b) % m
octet_courant = 8
message = [int(chiffre[x:x+2],16) for x in range(0,len(chiffre),2)]
chiffre = b''
for c in message:
    car_chiffre = gcl() ^ c
    chiffre += bytes([car_chiffre])
print(chiffre)
```

14 – Écrire un programme réalisant l'encodage base64 d'un fichier conformément à la RFC 2045.

Automate gérant les cas de terminaison d'abord :



état	opérations
1	1 caractère lu : 8bits, écriture d'un caractère sur 6bits, reste 2bits
6	reste 2bits: écriture d'un caractère sur 6bits avec les 2bits complétés avec '0000' et de '=='
2	1 caractère lu : 8bits, écriture d'un caractère sur 6bits avec les 2bits précédents, reste 4bits
7	reste 4bits: écriture d'un caractère sur 6bits avec les 4bits complétés avec '00' et de '='
3	1 caractère lu : 8bits, écriture de deux caractères sur 6bits avec les 4bits restants

```
#!/usr/bin/python3

import sys

alphabet_base64 =  bytes(range(b'A'[0],b'Z'[0]+1))+bytes(range(b'a'[0],b'z'[0]+1))
               +bytes(range(b'0'[0],b'9'[0]+1))+b'+/'

try:
    entree = open("exo14.py","br")
    sortie = open("exo14.py.base64","bw")
except Exception as e:
    print (e.args)
    sys.exit(1)

while True:
    car1 = entree.read(1)
    if not car1:
        break
    rep_binaire = format(car1[0],'08b')
    sortie.write(bytes([alphabet_base64[int('00'+rep_binaire[:6],2)]]))
    bits_restants = rep_binaire[6:]
    car2 = entree.read(1)
    if not car2:
        sortie.write(bytes([alphabet_base64[int('00'+bits_restants+'0000',2)]]))
        sortie.write(b'==')
        break
    rep_binaire = bits_restants + format(car2[0],'08b')
    sortie.write(bytes([alphabet_base64[int('00'+rep_binaire[:6],2)]]))
    bits_restants = rep_binaire[6:]
    car3 = entree.read(1)
    if not car3:
        sortie.write(bytes([alphabet_base64[int('00'+bits_restants+'00',2)]]))
        sortie.write(b'=')
        break
    rep_binaire = bits_restants + format(car3[0],'08b')
    sortie.write(bytes([alphabet_base64[int('00'+rep_binaire[:6],2)]]))
    sortie.write(bytes([alphabet_base64[int('00'+rep_binaire[6:],2)]]))
entree.close()
sortie.close()
```