## Manipulation de fichiers

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

```
#!/usr/bin/python
import sys
try:
	entre=open("fichier.txt","r")
except Exception, message:
	print message
	sys.exit(1)
cpt=0
while 1:
	une_ligne=entre.readline()
	if not une_ligne :
		break
	cpt=cpt+1
print cpt
entre.close()
```

**2 –** Écrire un programme qui effectue l'écriture de données dans un fichier texte :

```
#!/usr/bin/python
import sys
try:
	entre=open("fichier_autre.txt","a")
except Exception, message:
	print message
	sys.exit(1)
nom = raw_input("Entrer votre nom")
prenom = raw_input("Entrer votre prenom")
adresse = raw_input("Entrer votre adresse")
entre.write("%s : %s : %s\n" %(nom, prenom, adresse))
entre.close()
```

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

```
#!/usr/bin/python
import sys
try:
	entree=open("fichier.txt","r")
	sortie=open("nouveau_fichier.txt","w")
except Exception, message:
	print message
	sys.exit(1)
while 1:
	une_ligne = entree.readline()
	if not une_ligne:
		break
	sortie.write(une_ligne)
	une_ligne = entree.readline()
sortie.close()
entree.close()
```

## Gestion des listes

**4 –** 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/python
import sys, commands

resultat = commands.getoutput("ls *.py")
listeresult = resultat.splitlines()
for nomfic in listeresult:
        try:
                entree=open(nomfic,"r")
        except Exception, message:
                print message
                continue
        chaine = entree.readline()
        if not chaine :
                entree.close()
                continue
        print chaine
        entree.close()
```

**5 –** Écrire un programme qui réalise l'insertion d'un élément à un emplacement donné par son indice.
Pour faire cet exercice, il est possible d'utiliser une fonction.
Pour définir en Python, il faut utiliser le mot clé `def` :

```
def inserer_valeur(liste = [], indice = 0, valeur = 'x') :
        #contenu de la fonction
        return resultat
```

Lors de la déclaration de la fonction, il est possible en Python de nommer les arguments et de leur affecter des valeurs par défaut. Ainsi, lors de l'appel, il sera possible de :

```
>>> inserer_valeur([1,2,3], 2, 5)
[1, 2, 5, 3]
>>> inserer_valeur(indice=1,valeur='r')
['r']
>>> inserer_valeur()
['x']
>>> inserer_valeur(valeur='a',indice=0)
['x']
```

On peut :
- donner l'ensemble des paramètres lors de l'appel de la fonction ;
- seulement ceux que l'on veut dans n'importe quel ordre en utilisant des couples `nom=valeur` (on utilisera pour les autres les valeurs par défaut).

Soit la correction :

```
def inserer_valeur(liste = [], indice = 0, valeur = 'x') :
        if (indice >= len(liste)):
                liste.append(valeur)
                return liste
        nouvelle_liste = liste[:indice] # on récupère la première tranche
        nouvelle_liste.append(valeur)
        nouvelle_liste.extend(liste[indice:])
        return nouvelle_liste
```

## Utilisation des structures

**6 –** Écrire un programme lisant des données dans un fichier binaire enregistré suivant la structure C suivante :

```
struct {
        int numero ;
        char nom[10];
        float moyenne; }
```

et qui les sauve dans un fichier texte sous le forme : `nom, numéro, moyenne`.

**Correction :**

Pour la lecture du fichier binaire sous environnement Windows, il est conseillé de mettre comme mode d'ouverture `'rb'`, pour indiquer que le fichier doit être traité comme un fichier binaire et non pas comme un fichier texte. Sous Unix, ce n'est pas nécessaire mais tolérée.

```
#!/usr/bin/python
import sys, struct
# des lignes pour obtenir les noms des fichiers
try :
       entree = open(nom_fichier_entree, 'rb')
       sortie = open(nom_fichier_sortie, 'w')
except Exception, message\,:
       print message
       sys.exit(1) # une valeur ≠ de zéro indique une erreur
format = 'i10sf' # l'usage du 's' au lieu de 'c' indique une chaîne et non une liste de
10 caractères
while 1:
       donnees = entree.read(struct.calcsize(format))
       if not donnees : break
       (numero, nom, moyenne) = struct.unpack(format, donnees)
       sortie.write('%s, %d, %f\n' % (nom, numero, moyenne))
entree.close()
sortie.close()
```

## Utilisation des expressions rationnelles ou *Regular Expression*

**7 –** Écrire un programme qui récupère les informations enregistrées par le programme de l'exercice 2 (nom, prénom et adresse de chaque individu) et créé un nouveau fichier.

```
#!/usr/bin/python
import sys,re
try:
       entre=open("carnet_adresse.txt","r")
       sortie=open("mon_original.traduit","w")
except Exception, message:
       print message
       sys.exit(1)
re_expr = re.compile(r"^(\w+)\s*:\s*(\w+)\s*:\s*(\w+)\s*$",re.I)
#re_expr = re.compile(r"^((\w+)\s*:\s*){2}(\w+)$",re.I)
while 1:
       une_ligne=entre.readline()
       if not une_ligne :
              break
       print une_ligne
       res=re_expr.search(une_ligne)
       if res :
              sortie.write("%s ; %s ; %s \n" %(res.group(1).upper(),res.group(
2).upper(),res.group(3).upper()))
              #print "<",res.group(0),">"
       else : print("erreur sur la ligne")
entre.close()
sortie.close()
```

**8 –** Écrire un programme qui ouvre un fichier structuré de la manière suivante :

```
Nom = un_nom   # un_commentaire
Prenom = un_prenom                          # un_commentaire
Nom machine = un_nom_machine              # un_commentaire
Mot de passe = un_mot_de_passe            # un_commentaire
Champ_supplémentaire= une_valeur
...
Nom = un_nom   # un_commentaire
Prénom = un_prénom                          # un_commentaire
Champ_supplémentaire= une_valeur
```

On considère que le mot de passe peut contenir un espace.

```
1  #!/usr/bin/python
2  import re,sys
3
4  nom_fichier = "document.txt"
5  try:
6          entree = open(nom_fichier, "r")
7         sortie_nom = open(nom_fichier+".name","w")
8         sortie_mdp = open(nom_fichier+".password","w")
9  except Exception,message:
10         print "Le fichier %s ne peut etre ouvert et l'erreur est %s" % (nom_fichier,message)
11        sys.exit(1)
12
13 er_nom = re.compile(r"^Nom\s*=\s*([^#]*)", re.I)
14 er_mot_de_passe = re.compile(r"^Mot de passe\s*=\s*([\w\s:;,()]*[\w:;,()])", re.I)
15
16 while 1:
17         ligne = entree.readline()
18         if not ligne: break
19         resultat = er_nom.search(ligne)
20         if resultat:
21                 sortie_nom.write('%s\n'%resultat.group(1))
22                 continue
23         resultat = er_mot_de_passe.search(ligne)
24         if resultat:
25                 sortie_mdp.write('%s\n'% resultat.group(1))
26 entree.close()
27 sortie_mdp.close()
28 sortie_nom.close()
```

### Utilisation des opérations d'éclatement et de recomposition (split & join)

**9 –** On va utiliser une expression régulière pour définir la notion de séparateurs, et le `split` :

```
#!/usr/bin/python
import re, sys
#configuration
nom_defaut = "document.txt"
#programme
saisie = raw_input("Entrer le nom du fichier [%s]" % nom_defaut)
nom_fichier = saisie or nom_defaut
try:
        entree = open(nom_fichier, "r")
except Exception, message:
        print message
        sys.exit(1)
nb_lignes = nb_caracteres = nb_mots = 0
re_separateurs = re.compile(r"[\.,;:?!\s]+")

while 1:
       ligne = entree.readline()
       if not ligne : break
       ligne = ligne.rstrip('\n')
       nb_lignes += 1
       liste_mots = re_separateurs.split(ligne)
       nb_mots = nb_mots + len(liste_mots)
       for un_mot in liste_mots:
               nb_caracteres += len(un_mot)
moy_taille_mot =  nb_caracteres / float(nb_mots)
moy_taille_ligne = nb_mots / float(nb_lignes)
print "nombre de caracteres :%d nombre de mots :%d nombre de lignes : %d" % (nb_caracteres,
nb_mots, nb_lignes)
print "taille moyenne des mots : %.2f, taille moyenne des lignes :%.2f" % (moy_taille_mot,
moy_taille_ligne)
```

### Manipulation des dictionnaires

**10 –** L'utilisation d'un dictionnaire va permettre d'associer un nombre d'occurrence (une valeur du dictionnaire) à un mot (une clé du dictionnaire) :

```
#!/usr/bin/python
import re, sys
# on dispose de nom_fichier
try:
        entree = open(nom_fichier, "r")
except Exception, message:
        print message
        sys.exit(1)
re_separateurs = re.compile(r"[\.,;:?!\s]+")
dictionnaire = {}
while 1:
       ligne = entree.readline()
       if not ligne : break
       ligne = ligne.rstrip('\n')
       liste_mots = re_separateurs.split(ligne)
       for un_mot in liste_mots:
               #variante
               #dictionnaire[un_mot] = dictionnaire.get(un_mot, 0) + 1
               if dictionnaire.has_key(un_mot):
                       dictionnaire[un_mot] += 1
               else: dictionnaire[un_mot] = 1
les_cles = dictionnaire.keys()
les_cles.sort()
for cle in les_cles:
       print "(%s : %s)" % (cle, dictionnaire[cle])
```

**11 –** Écrire un programme qui effectue la lecture et la construction d'un dictionnaire à partir du contenu d'un fichier d'options structuré de la façon suivante :

```
nom_option = valeur_option
nom_option_2 = valeur_option_2
...
```

```
#!/usr/bin/python
import sys, re

nom_fichier = "config.txt"

try:
       fic_config = open(nom_fichier, "r")
except Exception, m:
       print m
       sys.exit(1)

dico_config = {}
re_analyse = re.compile(r'^(.*[^\s])\s*=\s*(.*[^\s])')

while 1:
       ligne = fic_config.readline()
       if not ligne : break
       resultat = re_analyse.search(ligne)
       if resultat :
              dico_config[resultat.group(1)] = resultat.group(2)

print dico_config
fic_config.close()
```

## Mise en œuvre des instructions de manipulation binaire

**12 –**

```
#!/usr/bin/python

import sys
defaut = "255280.bmp"
lecture = raw_input("Entrer le nom du fichier [%s]" % defaut)
nom_fichier = lecture or defaut
try:
        entree = open(nom_fichier, "r")
       sortie = open("nouveau.bin","w")
except:
        print "Le fichier %s est introuvable" % nom_fichier

while 1:
        car = entree.readline(1)
       if not car : break
       car = bin(ord(car))[2:]
       car = "0"*(8-len(car))+car
       print ">",car
       car = list(car)
       tmp = car[3]
       car[3]=car[4]
       car[4]=tmp
       car=''.join(car)
       print "<",car
       car=chr(int(car,2))

       sortie.write(car)
entree.close()
sortie.close()
```

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

```
#!/usr/bin/python

import sys

compteur=0
table_base64 = [chr(ord('A')+x) for x in range(0,26)] +  [chr(ord('a')+x) for x in
range(0,26)] +  [chr(ord('0')+x) for x in range(0,10)]+['+']+['/']

def charToBin(char) :
   repBin = bin(ord(char))[2:]
   return '0'*(8-len(repBin))+repBin

def binToChar(binCode) :
   return table_base64[int(binCode, 2)]

def ecrire(desc,car) :
   desc.write(car)
   global compteur # définie une variable globale
   compteur +=1
   if compteur == 76:
      desc.write('\n')
      compteur=0
try :
   entree = open(sys.argv[1], "r")
   sortie = open(sys.argv[2], "w")
except Exception, descr :
   print descr
   sys.exit(1)

while 1 :
   car1 = entree.readline(1)
   if not car1 : break
   # 8 bits sont disponibles
   mon_buffer = charToBin(car1)
   car_base64_1 = binToChar(mon_buffer[:6])
   mon_buffer = mon_buffer[6:]
   ecrire(sortie,car_base64_1)
   car2 = entree.readline(1)
   if not car2 :
      mon_buffer += '0000'
      car_base64_2 = binToChar(mon_buffer[:6])
      ecrire(sortie,car_base64_2)
      sortie.write('==')
      break
   mon_buffer += charToBin(car2)
   car_base64_2 = binToChar(mon_buffer[:6])
   ecrire(sortie,car_base64_2)
   mon_buffer = mon_buffer[6:]
   car3 = entree.readline(1)
   if not car3 :
      mon_buffer += '00'
      car_base64_3 = binToChar(mon_buffer[:6])
      ecrire(sortie,car_base64_2)
      ecrire(sortie,'=')
      break
   mon_buffer += charToBin(car3)
   car_base64_3 = binToChar(mon_buffer[:6])
   ecrire(sortie,car_base64_3)
   car_base64_4 = binToChar(mon_buffer[6:])
   ecrire(sortie,car_base64_4)

sortie.close()
entree.close()
```