TD 5 -
Manipulation de Fichiers

Informatique
MPSI - Lycée Thiers

2014/2015

Exercice 1 : Enoncé

Ecrire un éditeur de texte minimaliste : à l'appel de `edt()` le programmera attendra la saisie de texte par l'utilisateur, dont il écrira toutes les lignes dans un fichier texte. Dès que l'utilisateur saisira une ligne vide (en appuyant sur ENTREE), la saisie de texte s'achèvera et le programme demandera le nom du fichier à sauvegarder.

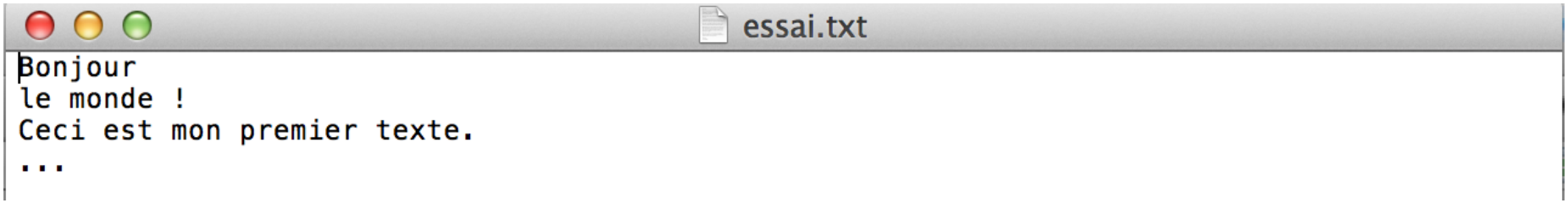
Exercice 1 : Editeur de texte

```
def edt():
    liste = []
    while True:
        ligne = input()
        if ligne == "":
            break
        else:
            liste.append(ligne + '\n')
    if len(liste) > 0:
        nom = input("Nom du fichier à sauvegarder : ")
        f = open(nom,'w')
        f.writelines(liste)
        f.close()
```

Exercice 1 : Editeur de texte

```
>>> edt()
Bonjour
le monde !
Ceci est mon premier texte.
...

Nom du fichier à sauvegarder : essai.txt
```



Exercice 2 : Enoncé

Ecrire une fonction `find()` qui recherche si un mot apparaît dans un fichier texte.

Exercice 2 : Corrigé

```
def find(file, mot):
    f = open(file,'r')
    liste = f.readlines()
    n = len(mot)
    for ligne in liste:
        N = len(ligne)
        for i in range(N-n+1):
            if ligne[i:i+n] == mot:
                f.close()
                return True
    f.close()
    return False
```

```
>>> find('essai.txt', 'Bonjour')
True
```

Exercice 3 : Enoncé

Ecrire une fonction `count()` qui compte le nombre de mots dans un fichier texte. Pour simplifier les mots seront délimités par des espaces et des sauts à la ligne.

Exercice 3 : Corrigé

```
def count(file):
    f = open(file,'r')
    n = 0
    liste = f.readlines()
    for ligne in liste:
        flag = False    # flag == True quand on a commencé un mot
        for char in ligne:
            if flag == False and char != ' ' and char != '\n':
                flag = True
            elif flag == True and (char == ' ' or char == '\n'):
                n += 1
                flag = False
    if flag == True:
        n += 1
    f.close()
    return n
```

Exercice 4 : Enoncé

On se propose de programmer un test de primalité efficace pour des nombres entiers ne dépassant pas une certaine valeur seuil.

1. Programmer un crible d'erathostène (cf. TD4 ou Cours 4) et l'utiliser pour construire la suite des nombres premiers inférieurs à 1000.
2. Une fois la liste obtenue, effectuer le produit de ses termes P et sauvegarder sa valeur comme première ligne dans un fichier texte suivie d'une ligne pour la liste des nombres premiers inférieurs à 1000.
3. Soit N un entier strictement supérieur à 1000 et inférieur à 10^6. A quelle condition nécessaire et suffisante sur le $pgcd(N, P)$, le nombre N est-il premier ?
4. Pouvez-vous en déduire un algorithme efficace qui teste si un entier inférieur à 10^6 est ou non premier ?
5. L'essayer pour différentes valeurs de N, notamment pour les *nombres de Mersenne* : de la forme $2^p - 1$ avec p premier. Déterminer lesquels sont premiers pour p un nombre premier variant de 2 à 19.
   Remarque (facile) : Sauriez-vous démontrer que si q n'est pas premier, alors $2^q - 1$ n'est pas premier ?

Exercice 4 : Corrigé

1) Crible d'Erathostène :

```
# Crible d'Erathostene :
def suppr0(L):
    M = []
    for x in L:
        if x != 0:
            M.append(x)
    return M

def erathostene(n):
    L = [x for x in range(n+1)]
    L[1] = 0
    for k in L:
        if k!= 0:
            t = 2
            while k*t <= n:
                L[k*t] = 0
                t+= 1
    return suppr0(L)
```

Exercice 4 : Corrigé

2)

```
L = erathostene(1000)
P = 1
for x in L:
    P = P*x

objet_fichier = open('prime.txt','w')
objet_fichier.write(str(P)+'\n')
objet_fichier.write(str(L))
objet_fichier.close()
```

3) On rappelle qu'un entier naturel N est premier si et seulement si : $N \geqslant 2$ et N n'a aucun diviseur autres que 1 et N.

• Si et seulement si N n'a aucun diviseur compris entre 2 et $\sqrt{N}$:

en effet si N avait une diviseur d_1 autre que 1 et lui-même il en aurait forcément un deuxième : $d_2 = N/d_1$ avec $d_2 \in [[2, N-1]]$.

Parmi les 2 diviseurs propres d_1 et d_2 au moins un est inférieur ou égal à $\sqrt{N}$. Autrement :

$$N = d_1 \cdot d_2 > \sqrt{N} \cdot \sqrt{N} = N$$

on obtiendrait $N > N$ ce qui serait contradictoire.

• On en déduit une condition nécessaire et suffisante pour qu'un entier $1000 < N \leqslant 10^6$ soit premier :

$$\boxed{PGCD(N, P) = 1}$$

Exercice 4 : Corrigé

On applique l'algorithme d'Euclide pour le calcul de pgcd :

```
def pgcd(a,b):
    while b!= 0:
        a,b=b,a%b
    return a
```

- On en déduit un test de primalité efficace pour tout entier $\leqslant 10^6$:

```
def test_prime(N):
    if N > 1.e6:
        return "Nombre trop grand"
    objet_fichier = open('prime.txt','r')
    P = eval(objet_fichier.readline())    # première ligne
    listePrime = eval(objet_fichier.readline())    # Deuxième ligne
    objet_fichier.close()
    if N <= 1000:
        return (N in listePrime)
    R = pgcd(P,N)
    if R == 1:
        return True
    else:
        return False
```

On prendra bien soin de ne pas recalculer P et L à chaque fois, mais bien une seule fois, pour aller ensuite récupérer leur valeur dans un fichier texte à chaque appel de la fonction.

Exercice 4 : Corrigé

```
# Test de primalité pour les nombres de Mersenne de 2^2-1 à 2^19-1 :
for p in L:
    if p>19:
        break
    if test_prime(2**p-1):
        print('Nombre de Mersenne : 2^',p,'-1 : premier')
    else:
        print('Nombre de Mersenne : 2^',p,'-1 : non premier')
```

Ce qui produit :

```
Nombre de Mersenne : 2^2 -1 : premier
Nombre de Mersenne : 2^3 -1 : premier
Nombre de Mersenne : 2^5 -1 : premier
Nombre de Mersenne : 2^7 -1 : premier
Nombre de Mersenne : 2^11 -1 : non premier
Nombre de Mersenne : 2^13 -1 : premier
Nombre de Mersenne : 2^17 -1 : premier
Nombre de Mersenne : 2^19 -1 : premier
```

Remarque : Lorsque q n'est pas premier :
Premier cas : $q = 1$. Dans ce cas $2^q - 1 = 2 - 1 = 1$ n'est pas premier.
Deuxième cas : $q > 1$. Dans ce cas $q = a \times b$ avec $a, b \in \mathbb{N} \setminus \{0, 1\}$:

$$2^q - 1 = 2^{ab} - 1 = (2^a)^b - 1^b = \underbrace{(2^a - 1)}_{>1} \times \underbrace{\sum_{k=0}^{b-1} (2^a)^q}_{>1} \quad \text{donc n'est pas premier}$$