Partie I :

1. On considère un fichier nommé *Voyelles.txt* constitué des voyelles de l'alphabet en minuscule sur une seule ligne. Que renvoie le programme suivant ?

```
txt=open('Voyelles.txt','r')
C=txt.read()
txt.close()
print(C)
```

Le programme renvoie `'aeiouy'`.

Écrire un code permettant d'ajouter une deuxième ligne au fichier *Voyelles.txt* contenant les voyelles en majuscules.

```
txt=open('Voyelles.txt','w')
txt.write('aeiouy\n AEIOUY')
txt.close()
```

2. On rappelle que la suite de Fibonacci $(F_n)_{n\in\mathbb{N}}$ est la suite définie par la relation de récurrence suivante

$$\forall n \in \mathbb{N}, \quad F_{n+2} = F_{n+1} + F_n \,.$$

Écrire une fonction `Fibo(F0,F1,n)` qui prend en argument trois entiers naturels `F0`, `F1` et `n`, et qui renvoie le n-ième terme de la suite de Fibonacci ayant pour premiers termes $F_0 =$ `F0` et $F_1 =$ `F1`.

```
def Fibo(F0,F1,n):
    a=F0
    b=F1
    if n==0:
        return F0
    if n==1:
        return F1
    for i in range(2,n):
        a,b=b,a+b
    return a+b
```

3. Considérons la fonction suivante prenant en argument une liste de flottants `a` et un flottant `x`.

```
def mystere(a,x):
    e=0
    for i in range(len(a)):
       e=a[len(a)-1-i]+x*e
    return e
```

Déterminer une expression mathématique de `mystere([2.,3.,1.],x)` en fonction de `x`.
La commande `mystere([2.,3.,1.],x)` renvoie $2 + x(3 + x(1 + x \times 0)) = x^2 + 3x + 2$.

Quel est l'objectif de cette fonction ?
Cette fonction renvoie l'évaluation en x du polynôme $a_0 + a_1X + \ldots a_nX^n$.

Écrire une nouvelle fonction `mystere2` dont les arguments et la sortie sont identiques à celles de `mystere`.

```
def mystere2(a,x):
    e=0
    for i in range (len(a)):
        e+=a[i]*x**i
    return e
```

Partie II : Nombres narcissiques

1. Montrer que 153 est un nombre narcissique en base 10.
   $1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$ donc 153 est un nombre narcissique en base 10.
2. Écrire une fonction `chiffres(n)` qui prend en argument un entier naturel `n` et renvoie une liste composée de ses chiffres en base 10. Par exemple, `chiffres(153)` renvoie la liste `[1,5,3]`.

```
def chiffres(n):
    a=n
    L=[]
    while a!=0:
        L=[a%10]+L
        a=a//10
    return L
```

3. Écrire une fonction `narcissique(n)` qui prend en entrée un entier naturel `n` et renvoie `True` si ce dernier est narcissique et `False` sinon.

```
def narcissique(n):
    L=chiffres(n)
    m=0
    for i in range(len(L)):
        m+=L[i]**len(L)
    if m==n:
        return True
    else:
        return False
```

4. Montrer que 17 est un nombre narcissique en base 3.
   $17 = 1 \times 3^2 + 2 \times 3^1 + 2 \times 3^0$ donc $17 = \overline{122}^3$. Or $17 = 1^3 + 2^3 + 2^3$ donc 17 est un nombre narcissique en base 3.
5. Écrire une fonction `chiffres_base(n,k)` qui prend en argument un entier naturel `n` et et renvoie une liste composée de ses chiffres en base `k`.

```
def chiffres_base(n,k):
    a=n
    L=[]
    while a!=0:
        L=[a%k]+L
        a=a//k
    return L
```

6. Écrire une fonction `narcissique_tous(n)` qui prend en entrée un entier `n` et renvoie une liste de taille 9 composée de nombres narcissiques. Le i-élément de cette dernière sera une liste contenant tous les nombres narcissiques inférieurs ou égaux à n en base $i + 2$.

```
def narcissique_tous(n):
    L=[]
    for k in range (2,11):
        Q=[]
        for j in range(n+1):
            c=chiffres_base(j,k)
            m=0
            for i in range(len(c)):
                m+=c[i]**len(c)
                if m==j:
                    Q.append(j)
        L.append(Q)
    return L
```