Correction DS n° 02 – Concours blanc

I Question de cours – Algorithme d'Euclide

Voir Cours C03-Introduction à la programmation.

II Exercice de TP – Complexités

Voir Correction TP04-Boucles et complexité.

III Exercice – Nombres heureux

1. $n = 5271 = 527 \times 10 + 1 = q \times 10 + r$: le quotient est $q = 527$ et le reste est $r = 1$;
   $q = 527 = 52 \times 10 + 7$: le nouveau quotient est 52 et le nouveau reste est 7.
   On constate que les restes successifs sont les chiffres du nombre initial en partant de la droite.
2. Grâce à la question précédente, on sait comment obtenir les chiffres d'un nombre. Il suffit d'en sommer les carrés.

```
def somme_carre(n): #définition
    r=n%10          #ce reste est le chiffre le plus à droite de nombre n
    q=n//10         #ce quotient contient les chiffres à gauche de r
    somme=r**2      #initialisation de la somme
    while q>0:      #tant qu'il reste quelque chose à diviser
        r=q%10      #chiffre une position plus à gauche
        somme = somme + r**2 #on ajoute son carré à la somme
        q=q//10     #nouveau quotient
    return somme    #on retourne la somme des carrés des chiffres
```

Code moins explicite mais plus compact :

```
def somme_carre(n):
    somme = 0
    while n > 0:
        somme = somme + (n % 10)**2
        n = n//10
    return somme
```

3. L'idée est de calculer les sommes de carrés des chiffres successivement comme dans l'exemple de l'énoncé, en réutilisant la fonction de la question précédente.
   Code explicite :

```
def heureux(n):
    L=[89,145,42,20,4,16,37,58] #liste des sommes de carrés qui prouvent
                                #que le nombre est malheureux
    while n not in L:      #tant que n n'est pas dans la liste
        if n==1:           #si n=1
            return True    #le nombre est heureux, on retourne True
        n=somme_carre(n)   #sinon on détermine la nouvelle somme des carrés
    return False           #si on est sorti de la boucle while c'est que
                           #le nombre est malheureux puisqu'une des sommes
                           #est dans la liste
```

Sur la page suivante, un code moins explicite mais plus élégant :

```python
def heureux(n):
    while True:           #élégant mais dangereux : la boucle est infinie
                          #sauf si on est SÛR qu'on sortira grâce à un return !
        if n == 1:
            return True
        elif n == 42:     #il suffit qu'une des sommes soit égale
            return False  #à une des sommes de la liste pour que le
                          #nombre soit malheureux ; on choisit bien sûr 42
        else:
            n = somme_carre(n)
```