TD 6 :
Nombres remarquables

MPSI - Lycée Thiers

Exercice 1

Exercice 1. Nombres parfaits, abondants, déficients.

Pour un entier $n \in \mathbb{N}$ un *diviseur strict* de n est est un diviseur de n différent de n. Par exemple :
6 a pour diviseurs : 1, 2, 3 et 6
6 a pour diviseurs stricts : 1, 2, et 3.

En arithmétique, un entier naturel > 0 est dit :

- **parfait** s'il est égal à la somme de ses diviseurs stricts. Par exemple 6 et 28 sont parfaits car :

  $6 = 1 + 2 + 3$
  $28 = 1 + 2 + 4 + 7 + 14$.

- **abondant** s'il est strictement inférieur à la somme de ses diviseurs stricts. Par exemple 12 est abondant puisque :

  $12 < 1 + 2 + 3 + 4 + 6$

- **déficient** s'il est strictement supérieur à la somme de ses diviseurs stricts. Par exemple : 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 13.

Ainsi tout entier naturel non-nul est soit parfait, soit abondant, soit déficient.

1. Ecrire une fonction `diviseurStricts(n)` prenant en paramètre un entier strictement positif `n` et qui retourne la liste de ses diviseurs stricts.
2. En déduire une fonction `parfait(n)` prenant en paramètre un entier positif `n` et qui retourne le booléen `True` ou `False` selon si `n` est un nombre parfait ou non.
3. Ecrire une fonction `nature(n)` prenant en paramètre un entier strictement positif `n` et qui retourne la chaîne de caractère `"parfait"`, `"abondant"` ou `"déficient"` selon si `n` est respectivement un nombre parfait, abondant ou déficient.
4. Ecrire une fonction `NATURE(n)` prenant en paramètre un entier $n > 0$ et qui inscit à l'écran pour chaque entier $0 < k \leqslant n$ s'il est parfait, abondant ou déficient. Par exemple `NATURE(6)` affichera :

```
1 est déficient
2 est déficient
3 est déficient
4 est déficient
5 est déficient
6 est parfait
```

5. On démontre que tout nombre parfait a pour chiffre des unités 6 ou 8. Ecrire une fonction `Parfait68(n)` prenant en paramètre un entier positif `n` et qui vérifie qui tout nombre parfait inférieur ou égal à `n` a pour chiffre

Exercice 1 : Corrigé

1.

```
def diviseursStricts(n):
    L = []
    for d in range(1,n):
        if n % d == 0:
            L.append(d)
    return L
```

2.

```
def parfait(n):
    L = diviseursStricts(n)
    S = 0
    for x in L:
        S += x
    return S == n
```

3.

```
def nature(n):
    L = diviseursStricts(n)
    S = 0
    for x in L:
        S += x
    if S == n:
        return "parfait"
    elif S > n:
        return "abondant"
    else:
        return "deficient"
```

4.

```
def NATURE(n):
    for k in range(1,n+1):
        print(k,"est",nature(k))
```

5.

```
def parfait68(n):
    for k in range(1,n+1):
        if parfait(k):
            u = k % 10
            if u != 6 and u != 8:
                return False
    return True
```

Exercice 2

Exercice 2. On cherche les solutions de l'équation diophantienne :

$$(x, y, z) \in \mathbb{N}^3,\ 0 < x \leqslant y \leqslant z \leqslant 100,\ x^2 + y^2 = z^2 \quad (*)$$

1. A l'aide d'une compréhension de liste créer la liste `L` constituée de toutes les t-uplets `(x,y,z)` pour lesquels (x, y, z) est solution de l'équation $(*)$.
2. En déduire la longueur des côtés du triangle rectangle qui parmi tous les triangles rectangles dont les côtés sont des entiers naturels inférieurs ou égaux à 100, a le plus grand périmètre.
3. Même question, mais pour celui dont l'aire est maximale.

Exercice 2 : Correction

1.

```
L = [ (x,y,z) for z in range(1,101) for y in range(1,z+1)
    for x in range(1,y+1) if x**2+y**2==z**2]
```

2. et 3.

```
m = M = 0
for x in L:
    p = sum(x)    # périmètre
    a = x[0]*x[1]/2    # aire
    if p > m:
        m = p
        T1 = x
    if a > M:
        M = a
        T2 = x
print("Périmètre maximal :",m)
print("obtenu pour",T1)
print("Aire maximale :",M)
print("obtenu pour",T2)
```

Exercice 3

On rappelle qu'une condition nécessaire et suffisante pour que 3 nombres strictement positifs a, b, c soient les longueurs des côtés d'un triangle non-plat est :

$$c < a + b \quad \text{et} \quad b < a + c \quad \text{et} \quad a < b + c$$

1. Ecrire une fonction `triangle(n)` prenant en paramètre un entier strictement positif `n`, et qui retourne la liste des triplets `(a,b,c)` tels que :
   - `a`, `b` et `c` sont des entiers avec : `0 < a ⩽ b ⩽ c` .
   - `a`, `b` et `c` sont les longueurs des côtés de triangles non-plats dont le périmètre est inférieur ou égal à `n`.
2. La *formule de Héron* permet de calculer l'aire d'un triangle à partir de la longueur de ses trois côtés. Soit un triangle de côtés de longueurs a, b et c, son aire est donnée par :

$$\mathscr{A} = \sqrt{p(p-a)(p-b)(p-c)} \qquad \text{où } p = \frac{a+b+c}{2}$$

   Ecrire une fonction `aireMax(n)` qui appelle la fonction `triangle(n)` et retourne le triplet des longueurs des côtés (à valeurs entières d'un triangle non-plat de périmètre au plus n) du triangle dont l'aire est maximale.

1.

```
def triangle(n):
    L = []
    for c in range(1,n):
        for b in range(1,c+1):
            for a in range(1,b+1):
                if c<a+b and b<a+c and a<b+c and a+b+c<=n:
                    L.append((a,b,c))
    return L
```

2.

```
def aireMax(n):
    L = triangle(n)
    m = 0
    for x in L:
        a,b,c = x
        p = (a+b+c)/2
        aire = (p*(p-a)*(p-b)*(p-c))**0.5
        if aire > m:
            m = aire
            T = x
    return T
```