Exercice 1. Il suffit de saisir dans la console `somme(n)` pour différentes valeurs de `n`.

Exercice 2.
a) Les deux appels renvoient une valeur approchée de $\sqrt{2}$.
b) Le premier appel fait ce que l'on attend : l'expression est évaluée en $1+\sqrt{2}$ et affche une valeur approchée. Le deuxième appel produit une erreur de type ; ceci car la fonction ne renvoie rien : un `Nonetype`, que l'interpréteur ne parvient pas à ajouter à 1 pour évaluer l'expression :

```
TypeError: unsupported operand type(s) for +: 'int'and 'NoneType'
```

Exercice 3.

```
## Exo 3.a)
def factorielle(n):
    P = 1
    for k in range(2,n+1):
        P *= k
    return P

# Exo 3.b)
for k in range(11):
    print(k,'! =', factorielle(k))
```

Exercice 4.

```
## Exo 4.a)
def fibonacci(n):
    u, v = 0, 1
    for k in range(n-1):
        u, v = v, u+v
    return v

# Exo 4.b)
print("F_ 0 = 0")
for k in range(1,20):
    print("F_",k,'=', fibonacci(k))

# Exo 4.c)
def sommeFibonacci(n):
    S = 0
    for k in range(1,n+1):
        S += fibonacci(k)
    return S
```

```
## Solution d'exécution beaucoup plus rapide :
def sommeFibonacci(n):
    u, v = 0, 1
    S = u+v
    for k in range(1,n+1):
        u, v = v, u+v
        S = S + v
    return S
```

Exercice 5.
L'appel de `mystere(7)` va demander à l'utilisateur de saisir les résultats de la table de multiplication de 7. Tant que l'utilisateur n'aura pas saisi le bon résultat, il ne pourra pas passer à la ligne suivante.

Exemple d'utilisation. L'utilisateur a eu du mal à se rappeler du résultat de 7×8 : il lui a fallu 3 tentatives (54, 58, et finalement 56) !

```
In [2]: mystere(7)
0 x 7 = 0
1 x 7 = 7
2 x 7 = 14
3 x 7 = 21
4 x 7 = 28
5 x 7 = 35
6 x 7 = 42
7 x 7 = 49
8 x 7 = 54
8 x 7 = 58
8 x 7 = 56
9 x 7 = 63
10 x 7 = 70
```

Plus généralement, avec en paramètre un entier n, `mystere(n)` permet de tester sa connaissance de la table de n.

Exercice 6.

```
## Algorithme d'Euclide du calcul de PGCD
def pgcd(a,b):
    while b!=0:
        a,b = b,a%b
    return a
```

Le PGCD de 26 221 et 42 357 est 2017 :

```
In [4]: pgcd(26221,42357)
```

```
Out[4]: 2017
```

Une fonction **ppcm** se déduit en utilisant la propriété : $pgcd(a,b)\times ppcm(a,b) = a\times b$:

```
def ppcm(a,b):
        return a*b//pgcd(a,b)
```

Exercice 7.

```
## Exo 4
for i in range(1,5):
    for k in range(i+1):
        print(' '*(5-(k+1))+'o'*(2*k+1))
```