Fonctions, par l'exemple

Un programme court mais incompréhensible

```
x = int(input())
y = int(input())
a = x
b = y
while  a % b != 0:
    r = a % b
    a = b
    b = r
z = b
print(x//z)
print(y//z)
```

```
num = int(input("Numérateur :"))
den = int(input("Dénominateur :"))
a = num
b = den
while  a % b != 0:
    r = a % b
    a = b
    b = r
d = b
print(f"Fraction réduite: {num//d}
   / {den//d}")
```

- À gauche, les noms de variables ne disent rien,
- on ne sait pas ce que fait la boucle `while` et le calcul qui va avec,
- les `print` et `input` ne sont pas explicites : on n'en comprendra pas davantage au moment d'exécuter.

Extraire la fonction

```
def pgcd(a,b):
    """
    a, b: deux entiers positifs, b non nul
    renvoie le pgcd de a, b
    """
    while  a % b != 0:
        r  % b
        a = b= a
        b = r
    return b

num = int(input("Numérateur :"))
den = int(input("Dénominateur :"))
d = pgcd(num, den)
print(f"Frac réduite: {num//d} / {den//d}")
```

- **signature**
- **documentation**
- **définition de la fonction**
- **exécution de la fonction**
- **principal ou main**

Bloc de programme qui s'exécute sur demande. À la définition, on indique les entrées utiles. Il faudra les fournir à l'exécution. La fonction renvoie *éventuellement* une (des) sortie(s).

Typer ? *Pas exigé en NSI*

Dans la signature, on **peut** donner des indications de type. On n'est pas obligé mais c'est bien de le faire, quand ça reste simple, car cela aide à la lecture.

```
def pgcd(a:int, b:int) -> int:
```

`return`, pas `print`

- Une fonction ne contient **jamais** de `print`...
- **SAUF** si la fonction est fait exprès pour l'affichage.
- La sortie d'une fonction est faite par `return`

- Certaines fonctions peuvent ne rien renvoyer et ne contenir aucun `return` (toutefois en Python, une fonction renvoie `none` à défaut d'autre chose)
- Après l'exécution d'un `return`, la fonction se termine immédiatement.
- Il peut y avoir plusieurs `return` dans une fonction.

Modulariser

```
# monmodule.py
def pgcd(a:int,b:int) -> int:
    ...
```

```
# main.py
from monmondule import pgcd
num = int(input("Numérateur :"))
...
```

On peut déplacer les fonctions dans un **module** séparé. Quand un programme compte beaucoup de fonctions, c'est très bienvenu.

Il faut utiliser des fonctions ! Beaucoup de fonctions.

Préconditions – tests simples → assertions

`assert` permet d'indiquer une condition qui **doit** être remplie.

- Précondition

```
def pgcd(a:int,b:int) -> int:
    """
    renvoie le pgcd de a et b
    """
    assert b !=0
```

- Test simple : on sait d'avance que `pgcd(15,35) == 5` alors on peut écrire, en tant que test simple :

```
def pgcd(a:int,b:int) -> int:
    """
    renvoie le pgcd de a et b
    """
    ...

assert pgcd(15,35) == 5
```

Portée des variables

La fonction `pgcd` utilise les variables `a`, `b`, `r`. Ces variables n'existe que le temps de l'exécution de la fonction.

cf. exercice pour en savoir plus sur cet aspect des fonctions.