Documentons les fonctions

Il est essentiel de documenter les fonctions que l'on crée afin de pouvoir les utiliser à bon escient. Pour l'utilisateur (qui n'est pas forcément celui qui a écrit la fonction), il est important de savoir ce que fait la fonction (une description rapide est requise), la liste des arguments de la fonction (avec les types, les conditions éventuelles à respecter, les valeurs par défaut des arguments optionnels...) ainsi que la liste des sorties de la fonction.

De plus, lorsqu'une fonction est bien documentée, il est possible de lire l'aide sans avoir besoin de lire le code. Il n'est bien souvent pas nécessaire de savoir comment la fonction est codée pour l'utiliser : c'est ce que l'on fait tous les jours lorsque l'on utilise les fonctions de base du langage ou celles de modules particuliers.

La documentation se fait **obligatoirement** juste après la déclaration de la fonction (avec l'indentation) en utilisant les triples quotes (ou triples doubles quotes). Voici le prototype de documentation :

```
def ma_fonction(arg1, arg2, arg3=arg3_default):
    """
    description de ce que fait la fonction à l'impératif

    Parameters
    ----------

    arg1: float
        à quoi sert l'argument 1

    arg2: int
        à quoi sert l'argument 2

    arg3: bool (optional)
        à quoi sert l'argument 3 (default = arg3_default)

    Returns
    -------

    out1: float
        description de l'output 1

    out2: int
        description de l'output 2

    ...

    """
    # code de ma fonction avec éventuellement des commentaires
    # pour expliquer le code
    ...
    return ...
```

Mais il est possible de faire plus en ajoutant des exemples d'utilisation, une image, ...

In [55]:
```python
def puissance(x, n=1):
    """
    retourne la puissance d'un réel

    Parameters
    ----------

    x: float or int
        le nombre qui sera élevé à la puissance

    n: int
        la puissance désirée (default = 1)
        n doit être un entier positif

    Returns
    -------

    y: float or int
        y = x**n

    Examples
    --------

    >>> puissance(10, 2)
    100

    >>> puissance(10, -1)
    -1 doit être un entier positif !
    """
    if not isinstance(n, int) or n < 0:
        print(f"{n} doit être un entier positif !")
        return None
    if n == 0:
        return 1
    return x * puissance(x, n-1)
```

In [56]:
```
help(puissance)
```

```
Help on function puissance in module __main__:

puissance(x, n=1)
    retourne la puissance d'un réel

    Parameters
    ----------

    x: float or int
        le nombre qui sera élevé à la puissance

    n: int
        la puissance désirée (default = 1)
        n doit être un entier positif

    Returns
    -------

    y: float or int
        y = x**n

    Examples
    --------

    >>> puissance(10, 2)
    100

    >>> puissance(10, -1)
    -1 doit être un entier positif !
```

In [57]:
```
puissance(10, -1)
```

```
-1 doit être un entier positif !
```

In [ ]:

In [ ]: