On écrit la fonction fibo dans le fichier fibonacci.py

```
def fibo(n) :
        fibo = [0]*(n)
        fibo[0] = 0   # Le premier est forcément à 0
        fibo[1] = 1   # Le deuxième est forcément à 1
        res = 0

        for i in range(2, n):
                fibo[i] = fibo[i-1] + fibo[i-2]
                res = fibo[i] + fibo[i-1]

        return res
```

1 - Assertions

On test l'assertion dans le fichier main.py

```
from fibonacci import fibo

assert fibo(3) == 2
assert fibo(7) == 13
```

On peut voir que l'assertion c'est bien passé car le programme ne retourne pas d'erreurs
Maintenant on va tester un cas d'erreur

```
from fibonacci import fibo

assert fibo(3) == 2
assert fibo(7) == 13
assert fibo(7) == 12, "Ne peut pas avoir cette valeur"
```

On obtient alors ceci en console

```
Traceback (most recent call last):
  File "...\main.py", line 5, in <module>
    assert fibo(7) == 12, "Ne peut pas avoir cette valeur"
AssertionError: Ne peut pas avoir cette valeur
```

On va ensuite faire le docstring de notre fonction fibonacci

```
def fibo(n) :
    """
    fibonacci[i] = fibonacci[i - 1] + fibonacci[i - 2]
    Le dernier nombre de la suite sera égale aux 2 nombres qui le précéde
    La suite commencera forcément par 0, 1
    puis vaudra donc 1 (0 + 1), 2 (1 + 1), 3 (2 + 1), 5 (3 + 2), 8 (5 + 3); etc...
    Ce qui nous donne 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...
    Et ce, à l'infinie...
    :param n: un entier naturel positif
    :return: la somme des 2 derniers entiers de la suite
    """
 fibo = [0]*(n)
 fibo[0] = 0   # Le premier est forcément à 0
 fibo[1] = 1   # Le deuxième est forcément à 1
 res = 0

 for i in range(2, n):
        fibo[i] = fibo[i-1] + fibo[i-2]
        res = fibo[i] + fibo[i-1]

 return res
```

2 - Doctest

Y rajouter 2 tests, dans le docstring

```
def fibo(n) :
        """
        fibonacci[i] = fibonacci[i - 1] + fibonacci[i - 2]
        Le dernier nombre de la suite sera égale aux 2 nombres qui le précéde
        La suite commencera forcément par 0, 1
        puis vaudra donc 1 (0 + 1), 2 (1 + 1), 3 (2 + 1), 5 (3 + 2), 8 (5 + 3); etc...
        Ce qui nous donne 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...
        Et ce, à l'infinie...
        >>> fibo(3)
        2
        >>> fibo(7)
        13
        :param n: un entier naturel positif
        :return: la somme des 2 derniers entiers de la suite
        """

        fibo = [0]*(n)
        fibo[0] = 0   # Le premier est forcément à 0
        fibo[1] = 1   # Le deuxième est forcément à 1
```

```
    res = 0

    for i in range(2, n):
        fibo[i] = fibo[i-1] + fibo[i-2]
        res = fibo[i] + fibo[i-1]

    return res
```

On va ensuite importer le module doctest

```
import doctest

def fibo(n) :
    """
    fibonacci[i] = fibonacci[i - 1] + fibonacci[i - 2]
    Le dernier nombre de la suite sera égale aux 2 nombres qui le précéde
    La suite commencera forcément par 0, 1
    puis vaudra donc 1 (0 + 1), 2 (1 + 1), 3 (2 + 1), 5 (3 + 2), 8 (5 + 3); etc...
    Ce qui nous donne 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...
    Et ce, à l'infinie...
    >>> fibo(3)
    2
    >>> fibo(7)
    13
    :param n: un entier naturel positif
    :return: la somme des 2 derniers entiers de la suite
    """

    fibo = [0]*(n)
    fibo[0] = 0   # Le premier est forcément à 0
    fibo[1] = 1   # Le deuxième est forcément à 1
    res = 0

    for i in range(2, n):
        fibo[i] = fibo[i-1] + fibo[i-2]
        res = fibo[i] + fibo[i-1]

    # print(fibo) #permet de visualiser la suite de Fibonacci
    return res

    doctest.testmod()
```

Dans le run de Python, au lancement, il ne se passe rien mais il n'y a pas d'erreur

On va donc modifier le test et remplacer le 3 du premier test, par un 2.

La, une erreur est levée

On remet le 3 à la place du 2, puis l'on va dans le terminal Python et on écrit :

python fibonacci.py

Puis

python fibonacci.py -v

pour avoir plus de précisions sur ce qu'il vient de se passer

On remplace à nouveau le 3 par un 2 puis ***python fibonacci.py -v***

3 – Pytest

Permet de faire des tests plus complets, et propose des diagnostics plus complets et explicite que les autres.

On peut démarrer l'utilisation de pytest en préfixant nos fonctions par *test_* , contenant chacune une assertion.

Dans le main, ajouter :

```
from fibonacci import fibo

import pytest

def test_0() :
    assert fibo(0) == 0
def test_3() :
    assert fibo(3) == 2
def test_7() :
    assert fibo(7) == 13
def test_neg() :
    with pytest.raises(ValueError) :
        fibo(-1)
```

Puis dans le terminal, écrire : pytest main.py

2 tests sont passé, 2 tests ont échoué. On peux voir les tests qui ont échoué.