Utilisation du module `doctest` afin de génerer des jeux de test avec 3 algorithmes classiques de parcours séquentiel.

```
import doctest
def appartient(tab, valeur):
    """
    retourne True si valeur est dans tab
    >>> appartient([10, 12, 3.14, 56, 3, 1], 12)
    True

    >>> appartient([10, 12, 3.14, 56, 3, 1], 0)
    False
    """
    for elem in tab:
        if elem == valeur:
            return True
    return False

def maximum(tab):
    """
    détermine le maximum de tab
    >>> maximum([100, 5, 3, 101, 12])
    101
    >>> maximum([14, 3, 12, 18, 3.5, -5])
    1
    """
    max = tab[0]
    for elem in tab:
        if elem > max:
            max = elem
    return max
```

On donne la version avec les indices également:

```
def moyenne(tab):
    """
    calcule la moyenne des
    valeurs de tab
    """
    S = 0
    for elem in tab:
        S = S+elem
    return S/len(tab)
```

```
def moyenne2(tab):
    """
    version avec le parcours des indices
    >>> moyenne2([1, 2, 3, 4])
    2.5
    """
    S = 0
    for i in range(len(tab)):
        S = S +tab[i]
    return S/len(tab)
```

On inclut l'appel à la fonction testmod:

```
doctest.testmod
```

En voici le résultat:

```
>>> %Run recherche_seq.py
**********************************************************************
File "recherche_seq.py", line 16, in __main__.maximum
Failed example:
    maximum([14, 3, 12, 18, 3.5, -5])
Expected:
    1
Got:
    18
**********************************************************************
1 items had failures:
   1 of   2 in __main__.maximum
***Test Failed*** 1 failures.
```