RÉSUMÉ 16 - DOCTESTS : DOCUMENTER ET TESTER

Lien vers l'activité : Doctests : documenter et tester
www.infoforall.fr - Dernière modif. : 10 12 2020

1 DÉMARCHE HABITUELLE DE CRÉATION DE FONCTIONS :

- 1 - On écrit ou reçoit la demande documentée
- 2 - On écrit une base pertinente de tests documentés. Ces tests doivent permettre :
  - de comprendre la fonction d'utiliser la fonction
  - de vérifier les cas un peu particulier qu'on a prédétecté comme problématique
- 3 - On écrit le code

```
 1  def note_valide(note) :
 2      '''Renvoie True si note est dans l'intervalle [ 0 ; 20 ]
 3
 4      :: param note (int) :  la note à tester
 5      : :return (bool) :: True si note dans [ 0 ; 20 ]
 6
 7      :Exemple:
 8
 9      >>> note_valide(5)
10      True
11      >>> note_valide(-5)
12      False
13
14      '''
15      if note > 20 :
16          return False
17      elif note < 0 :
18          return False
19      else :
20          return True
```

- 4 - On rajoute l'activation du doctest

```
22  if __name__ == "__main__":
23      import doctest
24      doctest.testmod()
```

2 ASSERT

Pour rendre une condition plus contraignante qu'une simple indication dans une documentation, on peut utiliser une **assertion**.
Pour cela, on utilise le mot-clé **assert**.
Son utilisation basique se résume à une succession de trois éléments :

- Le mot-clé **assert**
- Un espace suivi de la condition qu'on veut vérifier sous forme d'une expression booléenne
- Une virgule suivi du texte à afficher si l'expression est évaluée à False

En résumé :

```
1  assert note >= 0 and note <= 20, "Pas dans [0;20] !"
```

3 ASSERT POUR RÉALISER DES TESTS

Finalement, on peut même se passer des modules de tests automatiques et n'utiliser que des asserts contenus dans une fonction qu'on lancera pour tester nos fonctions.

```
1  def addition(a, b) :
2      return a+b
3
4  def test_maison() :
5      assert addition(10,5) == 15, 'Echec du test : addition(10,5) == 15'
6      assert addition(10,-5) == 5, 'Echec du test : addition(10,5) == 15'
7
8  if __name__ == '__main__' :
9      test_maison()
```