Python

Cours 6 - Docs, Erreurs, Tests Unitaires

Table des matières

Documentation du code (commentaires)

En Python la documentation est simple et rapide.

Il y a deux types de commentaires :

- Par ligne (unitaire) : avec le “hashtag” (#)
- Par bloc (multiple : avec les guillemets simples ou doubles en suite de 3 (”’ ou ”””)

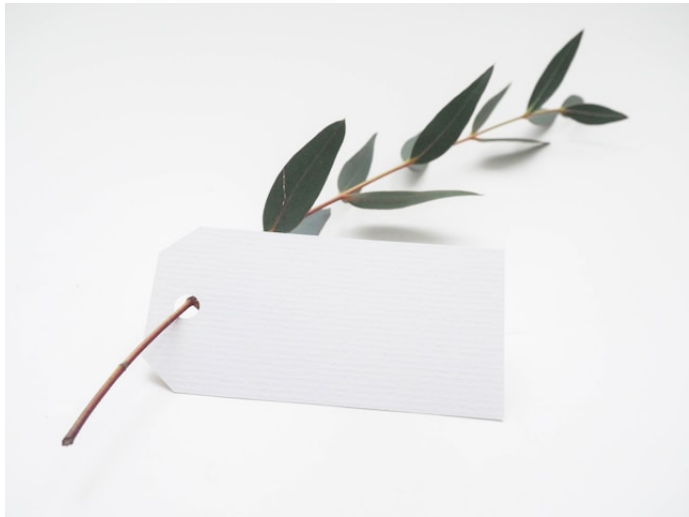

Figure: Un commentaire est comme une étiquette : elle ne fait pas partie du produit en tant que tel, elle y ajoute simplement de l’information pour référence.

Documentation du code (commentaires)

Listing: https://www.w3schools.com/python/python_comments.asp

```
1  # Steve Levesque, All rights reserved
2
3  """
4  A comment in block
5  1
6  2
7  """
8
9  # Comment for singular lines
10 # 1
11
12 """
13     This is a wrong comment evaluated as code...
14               ^^^^^
15 SyntaxError: invalid syntax
16 """
17 This is a wrong comment evaluated as code...
```

Erreurs (compilation et logique)

Puisque le code est écrit par des humains (ou maintenant par l'intelligence artificielle), celui-ci est sujet aux erreurs.

Il est possible de s'équiper avec des méthodes pour empêcher deux types d'erreurs courantes :

- Compilation : le code (une partie) n'exécute pas du tout et on voudrait quand même continuer l'exécution du programme
  - Solution : Exceptions
- Logique : le code exécute, mais le retour voulu est erroné
  - Solution : Tests unitaires

Exceptions (compilation)

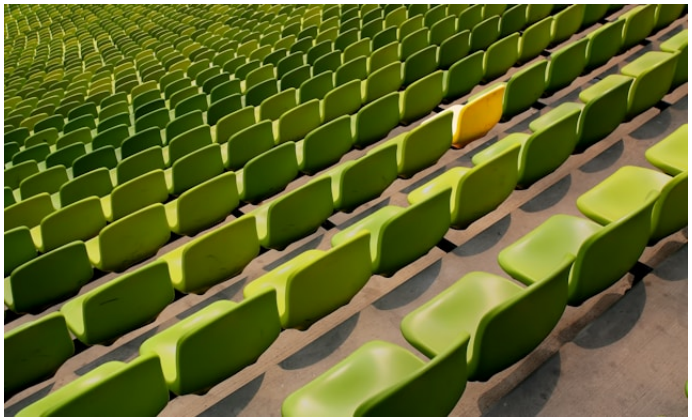

Figure: Une exception survient lors de l'exécution du programme.

En Python, il est possible de gérer les exceptions avec “try” et “expect”.

Les erreurs de compilation peuvent survenir de manière planifiée dans certains cas et seulement une gestion d'exception est la solution pour résoudre le problème.

Par contre, il est généralement non recommandé d'utiliser trop souvent un gestionnaire d'exception et de trouver une solution à la source lorsque possible.

Exceptions (compilation)

Listing: https://www.w3schools.com/python/python_comments.asp

```
1  # Steve Levesque, All rights reserved
2
3  # The variable "x" is not defined.
4  # In a normal execution, there would be a compilation error
5  # and the program would stop executing.
6  try:
7    print(x)
8  except:
9    print("An exception occurred")
10
11 # Result:
12 # An exception occurred
```

Tests unitaires (logique)

Si un programme retourne une valeur erronée, celui-ci compile tout de même.

L'ordinateur (le compilateur) n'a pas de compréhension vis-à-vis la logique du code à son domaine d'application.

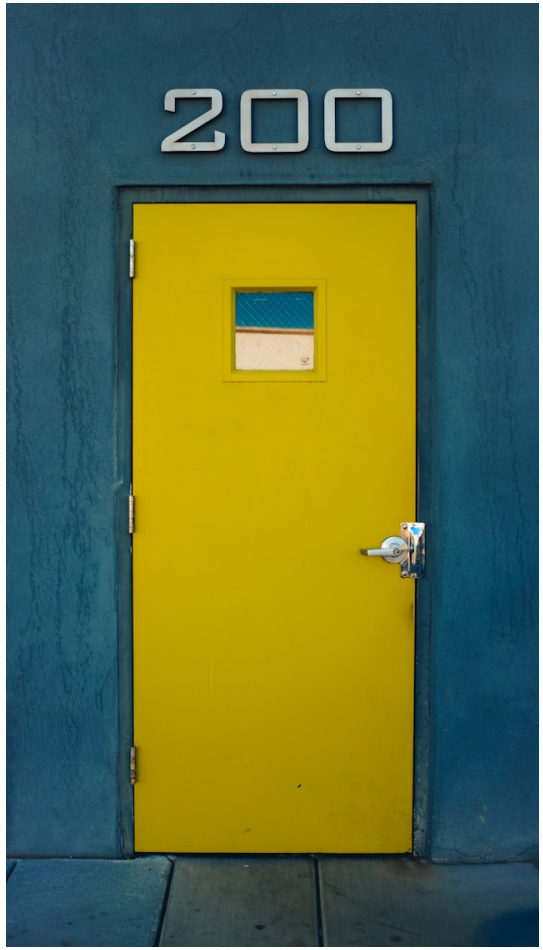

Figure: Un test unitaire est distinct et isolé.

Tests unitaires (logique)

Il est important de faire des tests unitaires pour valider et notifier les développeurs futurs si un changement non approprié brise le code.

Exemple : Nous allons tester la fonction “addition” utilisée dans l’exemple d’import de module local.

Tests unitaires (logique)

```python
# Steve Levesque, All rights reserved
import unittest
from calculator import addition

class TestCalculatorAddition(unittest.TestCase):

    def test_add_from_array(self):
        nums = [1, 3, 4]
        res = 8
        self.assertEqual(addition(nums), res)

    def test_add_from_array_large(self):
        nums = [1, 3, 4, 1, 1, 2, 3, 4, 5, 6, 7, 1, 3]
        res = 41
        self.assertEqual(addition(nums), res)

if __name__ == '__main__':
    unittest.main()
```

Tests unitaires (logique)

Si la fonction est mauvaise, voici un log avec les erreurs et de l'information additionnelle pertinente.

```
1  # Steve Levesque, All rights reserved
2  """
3  FAIL: test_add_from_array (__main__.TestCalculatorAddition.test_add_from_array)
4  ----------------------------------------------------------------------
5  Traceback (most recent call last):
6      self.assertEqual(addition(nums), res)
7  AssertionError: -8 != 8
8
9  ======================================================================
10 FAIL: test_add_from_array_large (__main__.TestCalculatorAddition.test_add_from_
11 ----------------------------------------------------------------------
12 Traceback (most recent call last):
13     self.assertEqual(addition(nums), res)
14 AssertionError: -41 != 41
15
16 ----------------------------------------------------------------------
17 Ran 2 tests in 0.001s
18
19 FAILED (failures=2)
20 """
```

Bibliographie

- https://www.w3schools.com/python/