

2TI – Dev II

Testing

Introduction

- Dans vos scripts, vous définissez des spécifications :
 - Signature des méthodes
 - Préconditions
 - Postconditions
- Comment vérifier que l'implémentation est conforme aux spécifications ?

Types de tests

Tests unitaires
Vérification du fonctionnement d'une
méthode/fonction d'un programme

Tests d'intégration
Vérifie le bon fonctionnement
de l'ensemble du programme

Tests d'acceptance / fonctionnels

[Test d'interface graphique]
Exceptionnellement ici pour Kivy
Tests qu'un bouton est bien premier enfant de la vue, par exemple.
from kivy.tests.common import GraphicUnitTest

Encore des tests

- Boite blanche
- Boite noire
- test fonctionnel
- test de performance
- test d'intrusion
- test utilisateur
- Test d'utilisabilité
- Test de performance
- Test de compatibilité
- Gestion des exceptions
- Test de charge
- Test de volume
- Test de stress
- Test de sécurité
- Test d'évolutivité
- Test d'aptitude
- Tests fumigatoires
- Tests exploratoires
- Test ad hoc
- Test de régression
- Test d'installation
- Test de maintenance
- Test de récupération
- Test d'accessibilité

Les données de test

Plus il y a de cas représentatifs qui sont vérifiés, mieux c'est !

Si la fonction permet de faire l'addition de deux entiers :

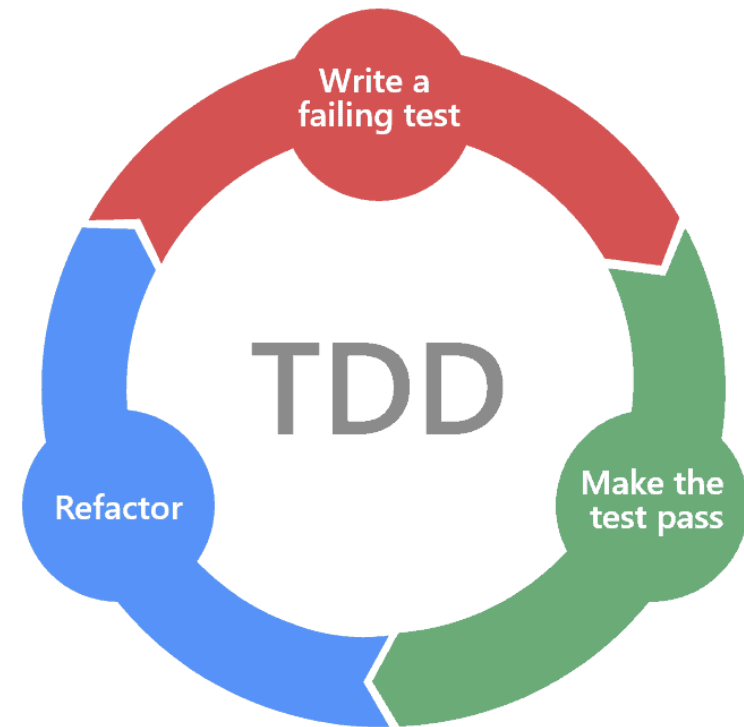
```
def addition (nbr1, nbr2):  
    return nbr1 + nbr2
```

Vous testerez alors, au minimum → → →

nbr1	nbr2	Résultat du test
0	10	
10	0	
-15	0	
0	6,10	
6,10	0	
12,5	6,10	

Test Driven Development

- Écrire les tests (vides)
- Vérifier qu'ils échouent donc existent
- Écrire les fonctionnalités liées aux tests
- Vérifier que les tests passent
- Amélioration du code si nécessaire



<https://marsner.com/blog/why-test-driven-development-tdd/>

TDD – Programmation par contrat


Reprenons la fonction d'addition :

```
def addition (nbr1, nbr2):  
    """  
    Cette fonction additionne deux nombres entiers.  
  
    :param nbr1: Un premier nombre entier  
    :type nbr1: int  
  
    :param nbr2: Un second nombre entier  
    :type nbr2: int  
  
    :returns: L'addition des deux nombres entiers  
    :rtype: int  
  
    """  
    pass
```

Idée du Black-Box testing (Basé sur les specifications)
[et du White-Box testing (Valeurs de même intervalle
&& test du code dans toutes ses possibilités)]



Restrictions sur les valeurs des paramètres



Ce qui est attendu comme résultat

Avec toutes ces informations, vous savez définir correctement vos tests

PRE & POST ne sont pas utilisés dans cet exemple, mais cette formulation représente la même chose, simplement d'une autre manière.

Tests unitaires - Exemple

```
# Importer les librairies nécessaires aux tests
import unittest

# Si besoin, importer les méthodes(class)/fonction à tester
from password_checker import check_password_length, check_password_characters

# Créer une classe de test qui hérite de unittest.TestCase
# Le nom de la classe définit la fonctionnalité qu'on teste
class TestPasswordChecker(unittest.TestCase):

    # Pour chacun des tests que l'on veut effectuer, on crée une méthode
    # Le nom de la méthode signifie le contexte qu'on teste

    def test_password_length(self):
        tst_pwd_short = 'ABC' # 3 caractères
        tst_pwd_long = 'ABCDEFGHJKLMNOPQRST' # 20 caractères

        self.assertTrue(check_password_length(tst_pwd_short))
        self.assertFalse(check_password_length(tst_pwd_long))
        self.assertEqual(check_password_length(tst_pwd_short), False)

    def test_password_contain_special_characters(self):
        # En TDD, on laisse tout cela à pass pour la première étape.
        pass

    def test_password_contain_numbers(self):
        pass

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


Tests unitaires - Résultats

Unittests in tests: 11 total, 11 passed			69 ms
			Collapse Expand
test_password_checker			43 ms
TestPasswordChecker			43 ms
test_password_contain_numbers	passed		43 ms
test_password_contain_special_characters	passed		0 ms
test_password_length	passed		0 ms
test_patient_key_generator			26 ms
TestPatientKeyGenerator			26 ms
test_generate_patient_key_is_only_letters_and_numbers	passed		26 ms
test_generate_patient_key_is_upper	passed		0 ms
test_generate_patient_key_length	passed		0 ms
test_professional_form			0 ms
TestPatientForm			0 ms
test_generate_patient_key_length	passed		0 ms
test_session_number_converter			0 ms
TestSessionNumberConverter			0 ms
test_GUI_list_of_companies	passed		0 ms
test_convert_session_number_to_text	passed		0 ms
test_session_screen			0 ms
TestSessionScreen			0 ms
test_get_a_random_video_number	passed		0 ms
test_video_order_check	passed		0 ms

Generated by PyCharm on 13/11/20 12:01