puissance4-test-annexe.zip

Puissance 4
Jeux de tests
Christophe Viroulaud
Première - NSI
Lang 09

Puissance 4
Jeux de tests

Christophe Viroulaud

Première - NSI

Lang 09

Le projet *Puissance 4* est composé de plusieurs fichiers et contient de nombreuses fonctions.

Le projet *Puissance 4* est composé de plusieurs fichiers et contient de nombreuses fonctions.

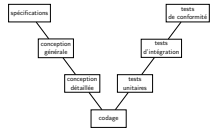
Comment détecter les erreurs dans un programme informatique ?

Comment détecter les erreurs dans un programme informatique ?

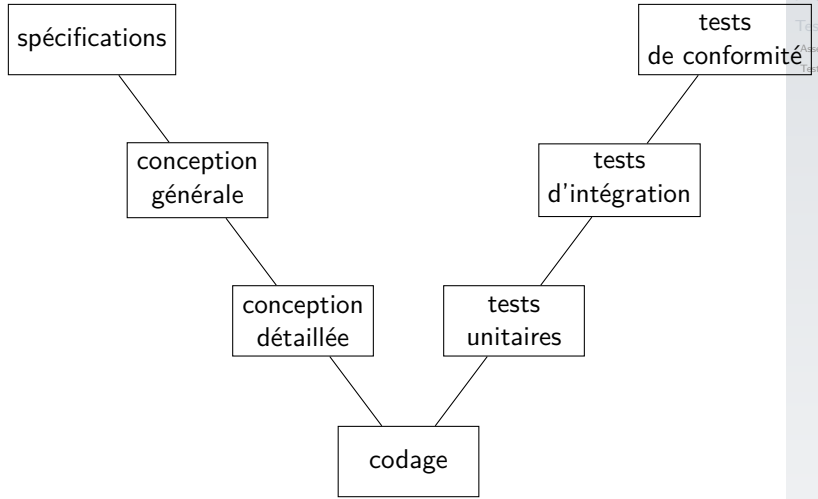
Sommaire
1. Cycle de vie d'un projet
2. Tests unitaires

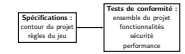
Sommaire

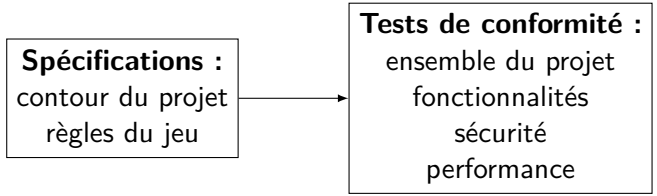
Cycle de vie d'un projet



Cycle de vie d'un projet

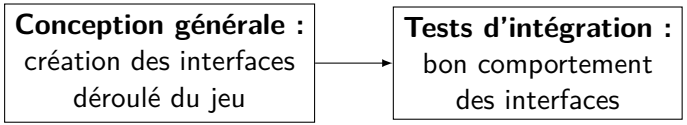






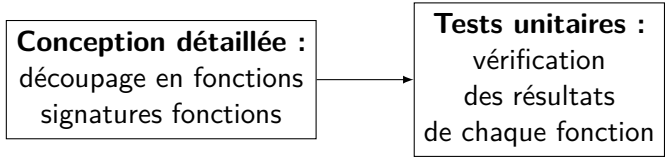
Exemple : Respect des règles du jeu

Conception générale : création des interfaces déroulé du jeu → Tests d'intégration : bon comportement des interfaces
Exemple : Respect des différentes séquences du jeu (positionnement graphique du jeton...)



Exemple : Respect des différentes séquences du jeu (positionnement graphique du jeton...)

Conception détaillée : découpage en fonctions signatures fonctions
Tests unitaires : vérification des résultats de chaque fonction
Exemple : Vérification de chaque fonction de calcul du gagnant (vertical, horizontal)

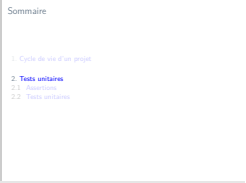


Exemple : Vérification de chaque fonction de calcul du gagnant (vertical, horizontal)

À retenir

- Un projet est découpé en plusieurs étapes.
- Chaque étape peut être réalisée par des équipes différentes.
- Les tests peuvent prendre plus de la moitié du temps de réalisation.

Sommaire



Sommaire

Test unitaires - assertions

Définition
assertion : Proposition que l'on avance et que l'on soutient comme vraie.

Test unitaires - assertions

Définition

assertion : Proposition que l'on avance et que l'on soutient comme vraie.

def placer_jeton(grille: list, colonne: int, joueur) -> int:
ligne = tomber_ligne(grille, colonne)
grille[ligne][colonne] = joueur
return ligne

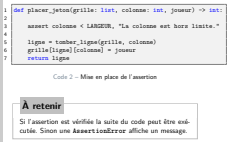
Code 1 – Une fonction utile du Puissance 4

Assertion : la valeur de la colonne ne doit pas dépasser la largeur du plateau.


```python
def placer_jeton(grille: list, colonne: int, joueur) -> int:
    ligne = tomber_ligne(grille, colonne)
    grille[ligne][colonne] = joueur
    return ligne
```

Code 1 – Une fonction utile du Puissance 4

Assertion : la valeur de la colonne ne doit pas dépasser la largeur du plateau.

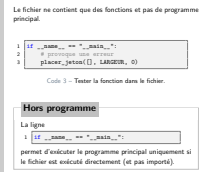



```python
def placer_jeton(grille: list, colonne: int, joueur) -> int:

    assert colonne < LARGEUR, "La colonne est hors limite."

    ligne = tomber_ligne(grille, colonne)
    grille[ligne][colonne] = joueur
    return ligne
```

Code 2 – Mise en place de l'assertion

À retenir

Si l'assertion est vérifiée la suite du code peut être exécutée. Sinon une **`AssertionError`** affiche un message.


Le fichier ne contient que des fonctions et pas de programme principal.

```
1 if __name__ == "__main__":
2     # provoque une erreur
3     placer_jeton([], LARGEUR, 0)
```

Code 3 – Tester la fonction dans le fichier.

Hors programme

La ligne

```
1 if __name__ == "__main__":
```

permet d'exécuter le programme principal uniquement si le fichier est exécuté directement (et pas importé).

Le fichier ne contient que des fonctions et pas de programme principal.

```
1 if __name__ == "__main__":
2     # provoque une erreur
3     placer_jeton([], LARGEUR, 0)
```

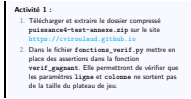
Code 3 – Tester la fonction dans le fichier.

Hors programme

La ligne

```
1 if __name__ == "__main__":
```

permet d'exécuter le programme principal uniquement si le fichier est exécuté directement (et pas importé).



Activité 1 :

1. Télécharger et extraire le dossier compressé **puissance4-test-annexe.zip** sur le site https://cviroulaud.github.io
2. Dans le fichier **fonctions_verif.py** mettre en place des assertions dans la fonction **verif_gagnant**. Elle permettront de vérifier que les paramètres **ligne** et **colonne** ne sortent pas de la taille du plateau de jeu.

Activité 1 :

1. Télécharger et extraire le dossier compressé **puissance4-test-annexe.zip** sur le site https://cviroulaud.github.io
2. Dans le fichier **fonctions_verif.py** mettre en place des assertions dans la fonction **verif_gagnant**. Elle permettront de vérifier que les paramètres **ligne** et **colonne** ne sortent pas de la taille du plateau de jeu.

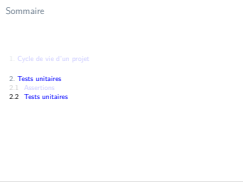
```python
def verif_gagnant(grille: list, joueur: int, ligne:
    int, colonne: int) -> bool:

    assert ligne < HAUTEUR and ligne >= 0, "ligne
    hors limite"

    assert colonne < LARGEUR and colonne >= 0, "
    colonne hors limite"

# reste du code
```

```python
if __name__ == "__main__":
    # provoque une erreur de ligne
    verif_gagnant([], 0, HAUTEUR, 0)
```

Code 4 – Tester la fonction dans le fichier.

Code 4 – Tester la fonction dans le fichier.

Sommaire



Sommaire

Tests unitaires

Réaliser des tests externes pour :
- ne pas alourdir le code des fonctions,
- automatiser les tests.

Tests unitaires

Réaliser des tests externes pour :

- ne pas alourdir le code des fonctions,
- automatiser les tests.

Réaliser des tests externes pour :
- ne pas alourdir le code des fonctions,
- automatiser les tests.

bibliothèque Python **unittest**

Réaliser des tests externes pour :

- ne pas alourdir le code des fonctions,
- automatiser les tests.

bibliothèque Python **unittest**

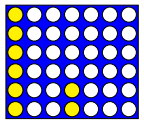

Hors programme
La construction complète du fichier de tests n'est pas à maîtriser en classe de première.
Activité 2 : Ouvrir le fichier tests_placement.py

Hors programme

La construction complète du fichier de tests n'est pas à maîtriser en classe de première.

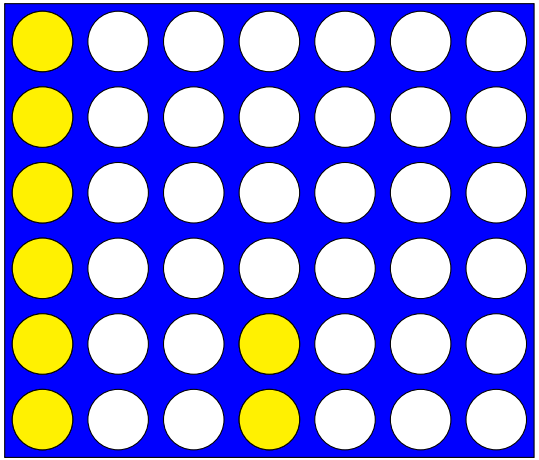
Activité 2 : Ouvrir le fichier `tests_placement.py`

À retenir
Pour tester les fonctions il faut maîtriser les données de tests.



À retenir

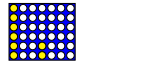
Pour tester les fonctions il faut maîtriser les données de tests.



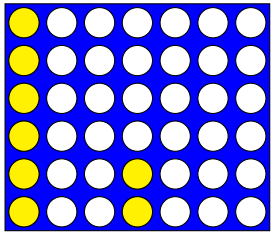
Code 5 – Initialiser des données pour les tests


```python
def setUp(self):
    """
    initialise la grille pour les tests
    """
    self.grille = [[VIDE for i in range(LARGEUR)]
                            for j in range(HAUTEUR)]
```

Code 5 – Initialiser des données pour les tests



```python
def setUp(self):
    """
    initialise la grille pour les tests
    """
    self.grille = [[VIDE for i in range(LARGEUR)]
                        for j in range(HAUTEUR)]

    # rempli la première colonne
    for i in range(HAUTEUR):
        self.grille[i][0] = JAUNE
    # place 2 jetons dans colonne 3
    self.grille[5][3] = JAUNE
    self.grille[4][3] = JAUNE
```




```python
def setUp(self):
    """
    initialise la grille pour les tests
    """
    self.grille = [[VIDE for i in range(LARGEUR)]
                        for j in range(HAUTEUR)]

    # rempli la première colonne
    for i in range(HAUTEUR):
        self.grille[i][0] = JAUNE
    # place 2 jetons dans colonne 3
    self.grille[5][3] = JAUNE
    self.grille[4][3] = JAUNE
```

```python
def test_remplie(self):
    # test OK si renvoie est True
    self.assertTrue(est_remplie(self.grille, 0))
    # test OK si renvoie est False
    self.assertFalse(est_remplie(self.grille, 1))
```

Code 6 – Réaliser des tests

Observations

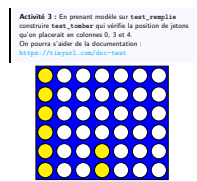
- La colonne 0 est pleine, la fonction doit renvoyer `True`.
- La colonne 1 est vide, la fonction doit renvoyer `False`.


```python
def test_remplie(self):
    # test OK si renvoie est True
    self.assertTrue(est_remplie(self.grille, 0))
    # test OK si renvoie est False
    self.assertFalse(est_remplie(self.grille, 1))
```

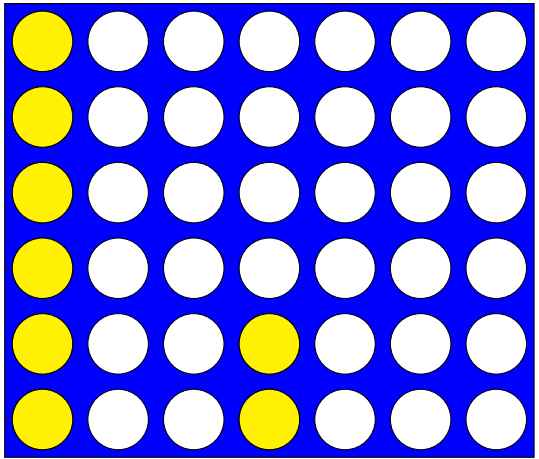
Code 6 – Réaliser des tests

Observations

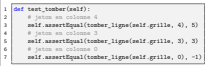
- La colonne 0 est pleine, la fonction doit renvoyer `True`.
- La colonne 1 est vide, la fonction doit renvoyer `False`.



Activité 3 : En prenant modèle sur `test_remplie` construire `test_tomber` qui vérifie la position de jetons qu'on placerait en colonnes 0, 3 et 4.
On pourra s'aider de la documentation :
https://tinyurl.com/doc-test



Correction



Correction

```python
def test_tomber(self):
    # jeton en colonne 4
    self.assertEqual(tomber_ligne(self.grille, 4), 5)
    # jeton en colonne 3
    self.assertEqual(tomber_ligne(self.grille, 3), 3)
    # jeton en colonne 0
    self.assertEqual(tomber_ligne(self.grille, 0), -1)
```

Remarque

Il est possible de créer un fichier de tests pour chaque fichier de fonctions.

À retenir
Les phases de tests sont une étape indispensable du cycle de vie d'un projet : elles permettent de maintenir la cohérence du code lors des modifications, ajouts...

À retenir

Les phases de tests sont une étape indispensable du cycle de vie d'un projet : elles permettent de maintenir la cohérence du code lors des modifications, ajouts...