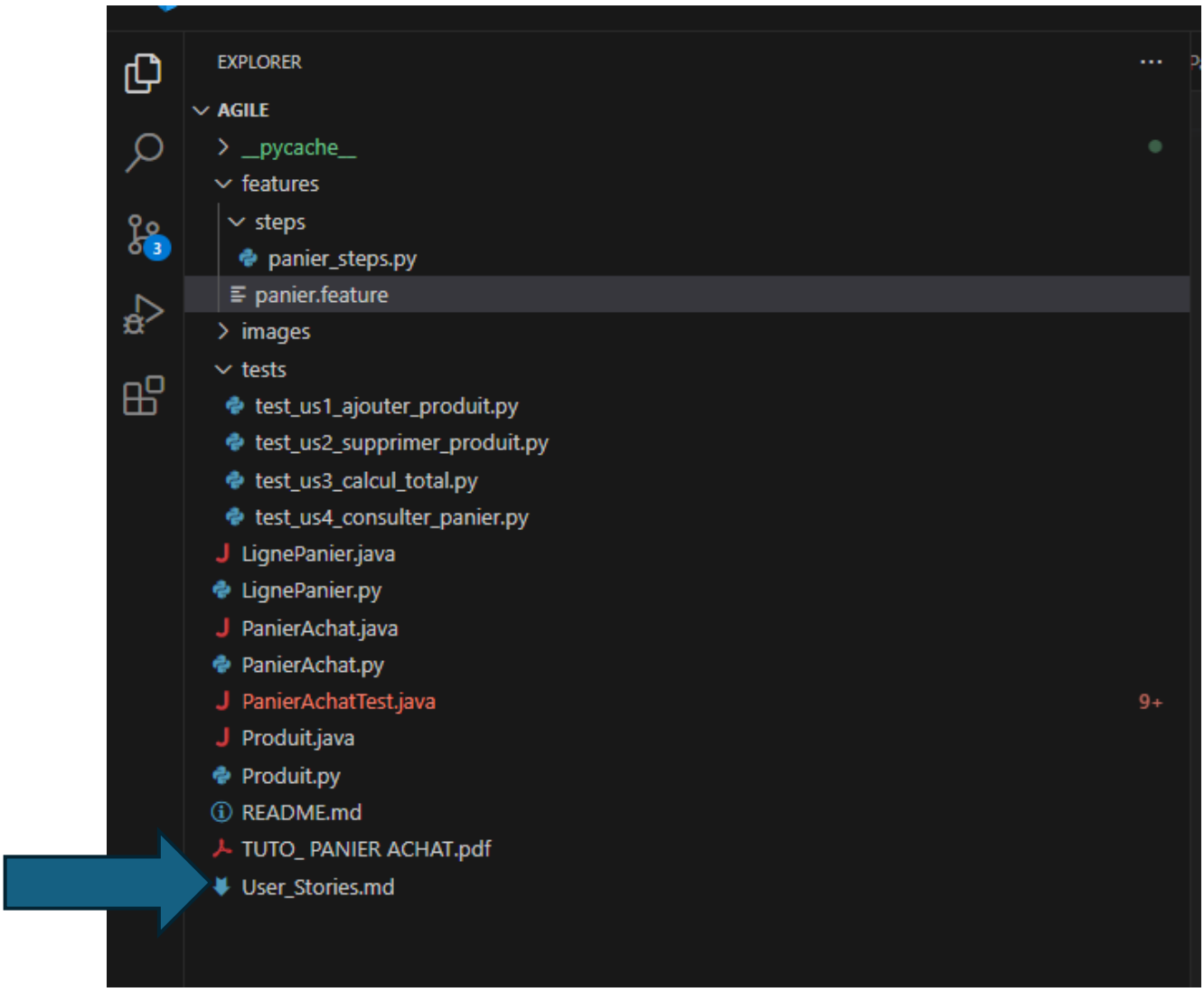
1-User Stories

Les User Stories ont été formalisées dans un fichier Markdown (User_Stories.md).
Ce fichier est disponible dans le dépôt Git du projet.

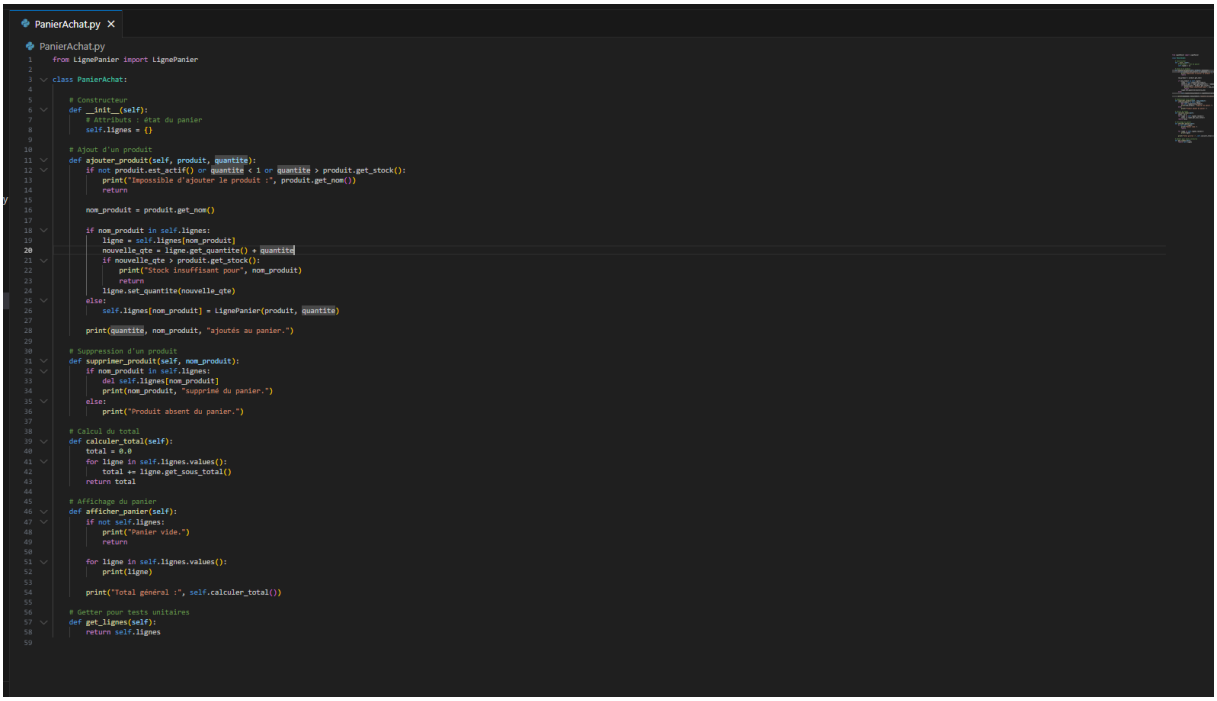


2-ECRITURES DES CODES + TEST UNITAIRES PLUS FONTIONNELS

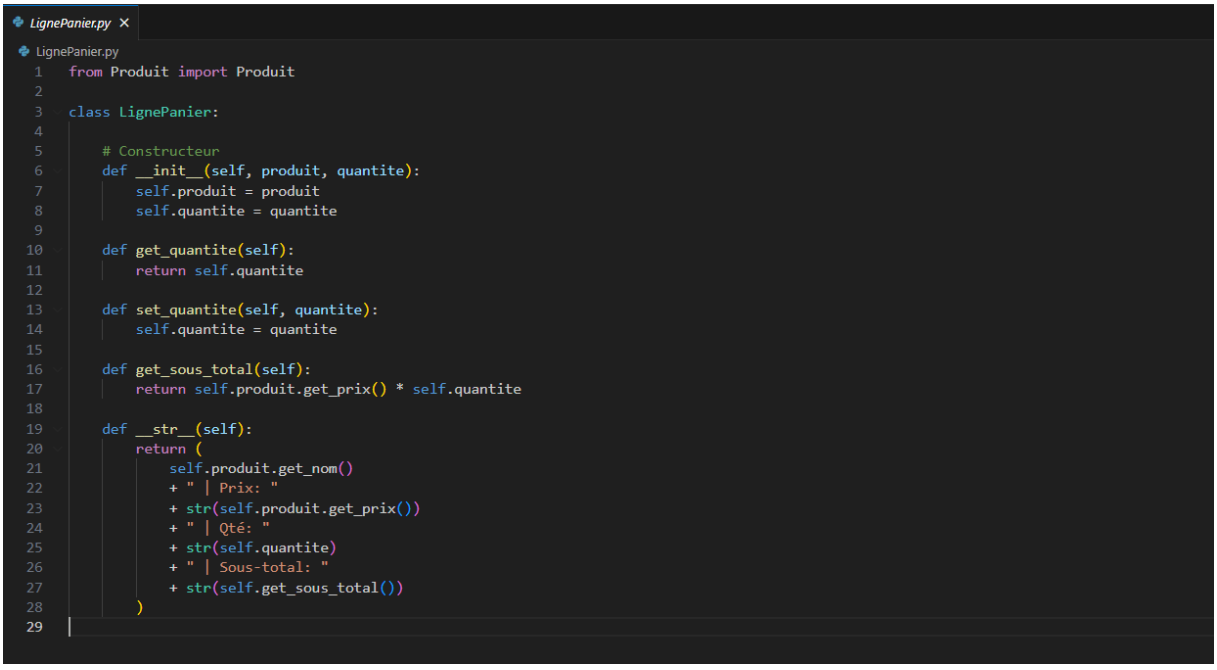
2.1. Traduction du code en Python

Le code initialement développé en Java a été traduit en Python

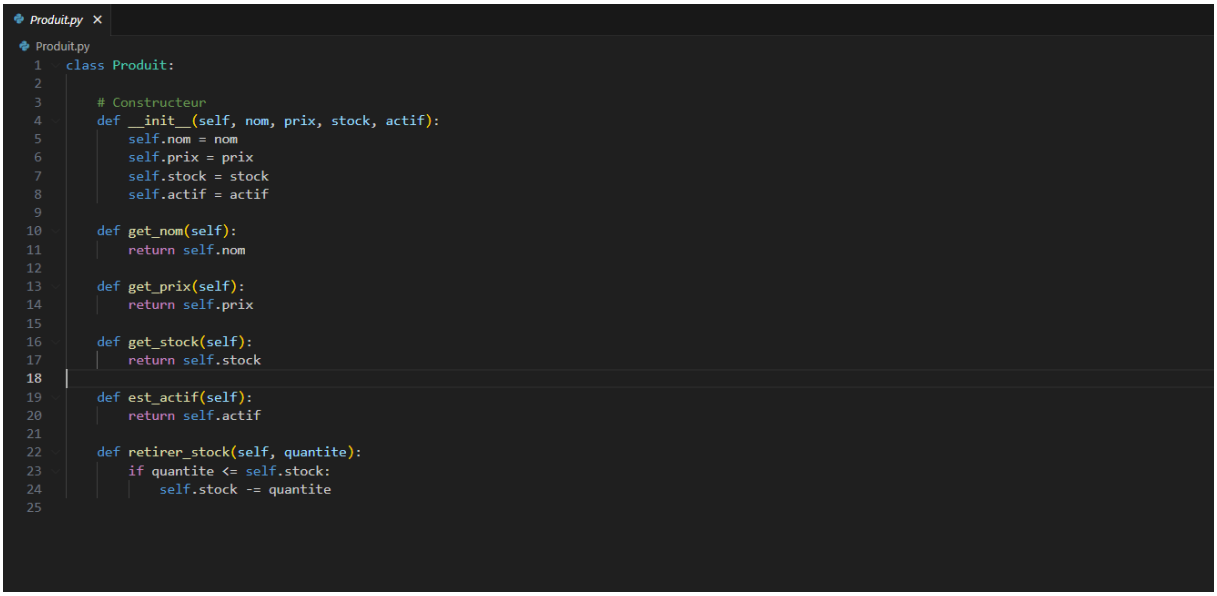
PanierAchat:



Lignepanier:



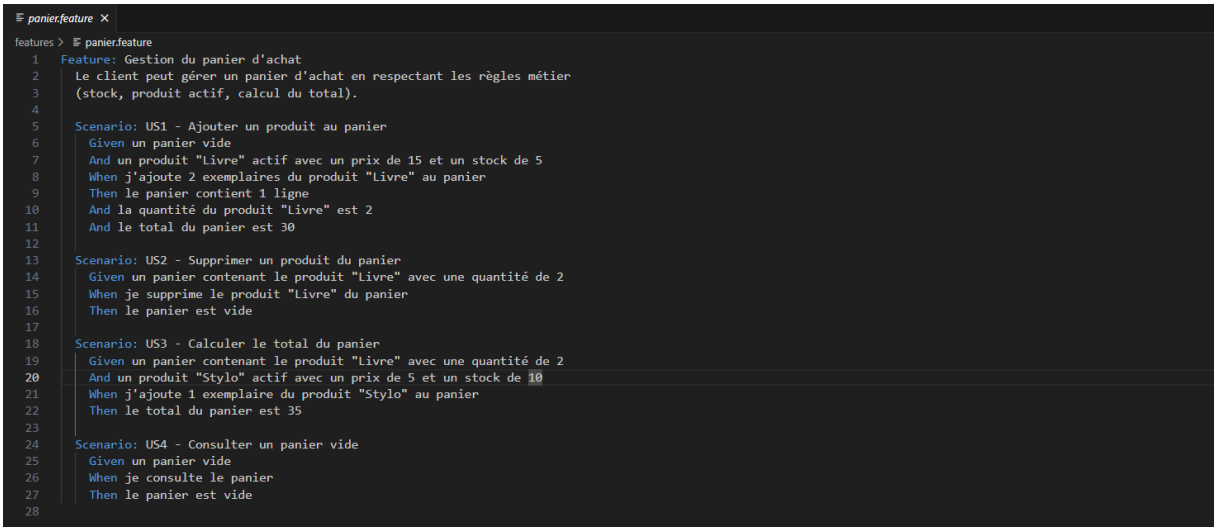
Produit:



```python
class Produit:

    # Constructeur
    def __init__(self, nom, prix, stock, actif):
        self.nom = nom
        self.prix = prix
        self.stock = stock
        self.actif = actif

    def get_nom(self):
        return self.nom

    def get_prix(self):
        return self.prix

    def get_stock(self):
        return self.stock

    def est_actif(self):
        return self.actif

    def retirer_stock(self, quantite):
        if quantite <= self.stock:
            self.stock -= quantite
```

2-2-Features(.feature)

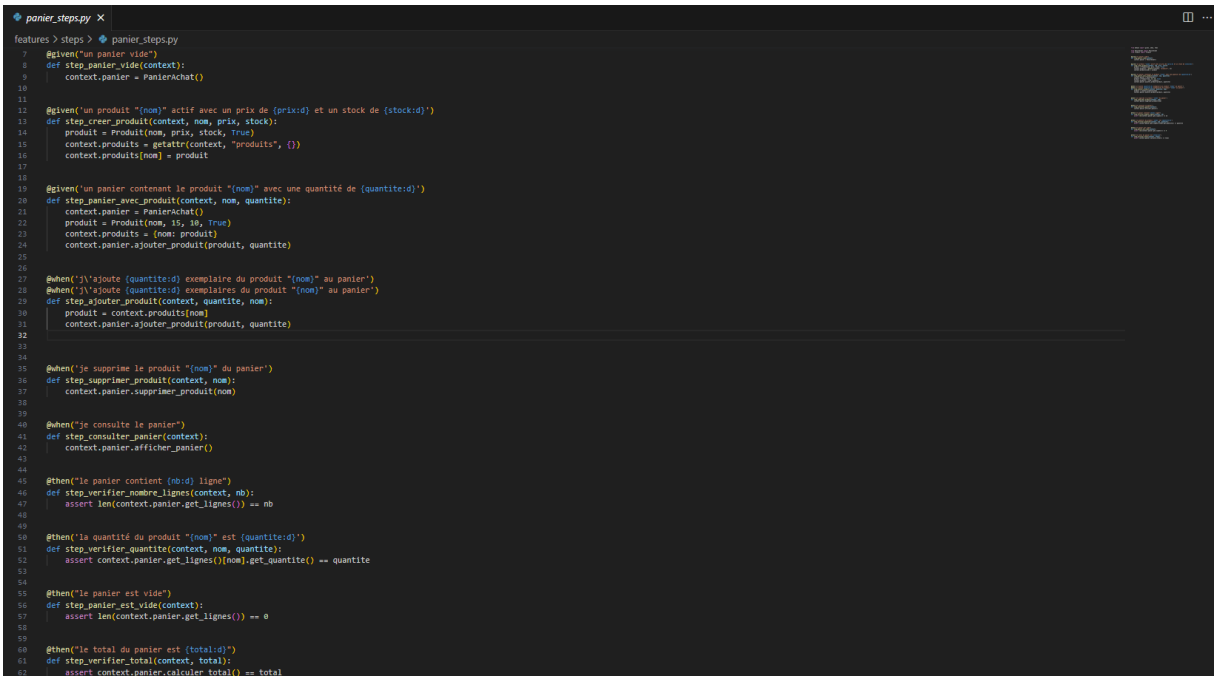
Les fonctionnalités du panier ont été décrites sous forme de scénarios BDD dans un fichier .feature. Ces scénarios traduisent les User Stories en langage naturel structuré (Given / When / Then) et servent de référence fonctionnelle.



```gherkin
Feature: Gestion du panier d'achat
  Le client peut gérer un panier d'achat en respectant les règles métier
  (stock, produit actif, calcul du total).

  Scenario: US1 - Ajouter un produit au panier
    Given un panier vide
    And un produit "Livre" actif avec un prix de 15 et un stock de 5
    When j'ajoute 2 exemplaires du produit "Livre" au panier
    Then le panier contient 1 ligne
    And la quantité du produit "Livre" est 2
    And le total du panier est 30

  Scenario: US2 - Supprimer un produit du panier
    Given un panier contenant le produit "Livre" avec une quantité de 2
    When je supprime le produit "Livre" du panier
    Then le panier est vide

  Scenario: US3 - Calculer le total du panier
    Given un panier contenant le produit "Livre" avec une quantité de 2
    And un produit "Stylo" actif avec un prix de 5 et un stock de 10
    When j'ajoute 1 exemplaire du produit "Stylo" au panier
    Then le total du panier est 35

  Scenario: US4 - Consulter un panier vide
    Given un panier vide
    When je consulte le panier
    Then le panier est vide
```

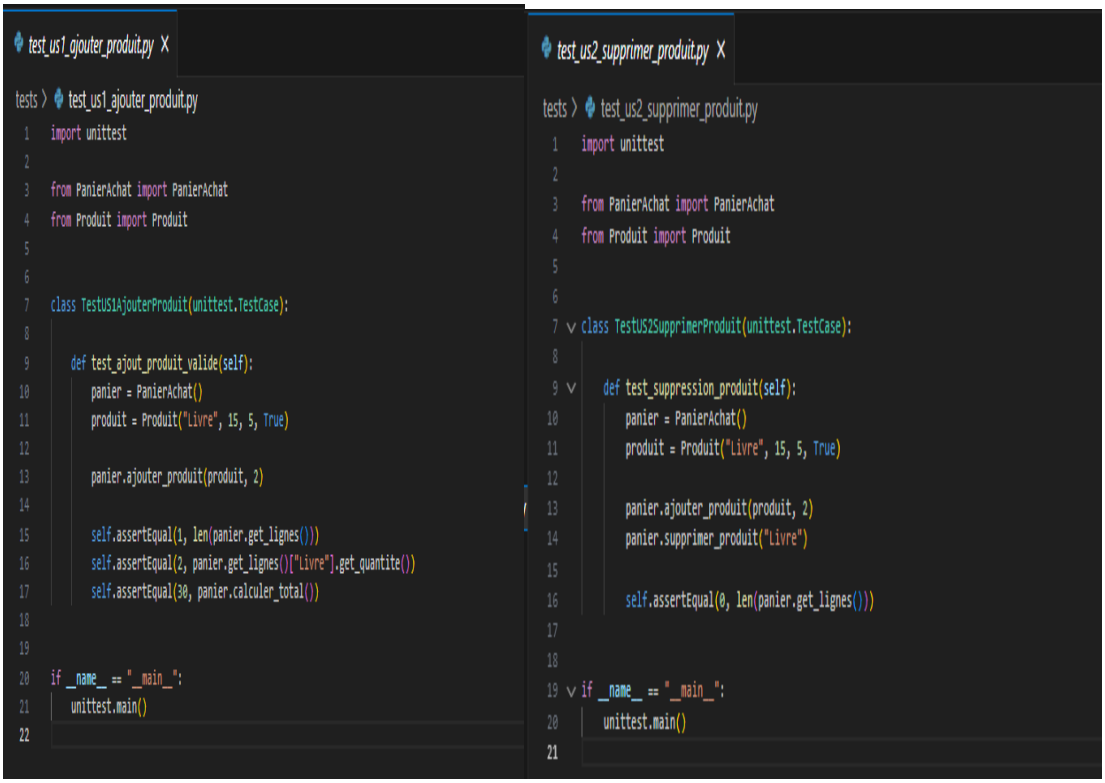
2 .3- Steps Behave

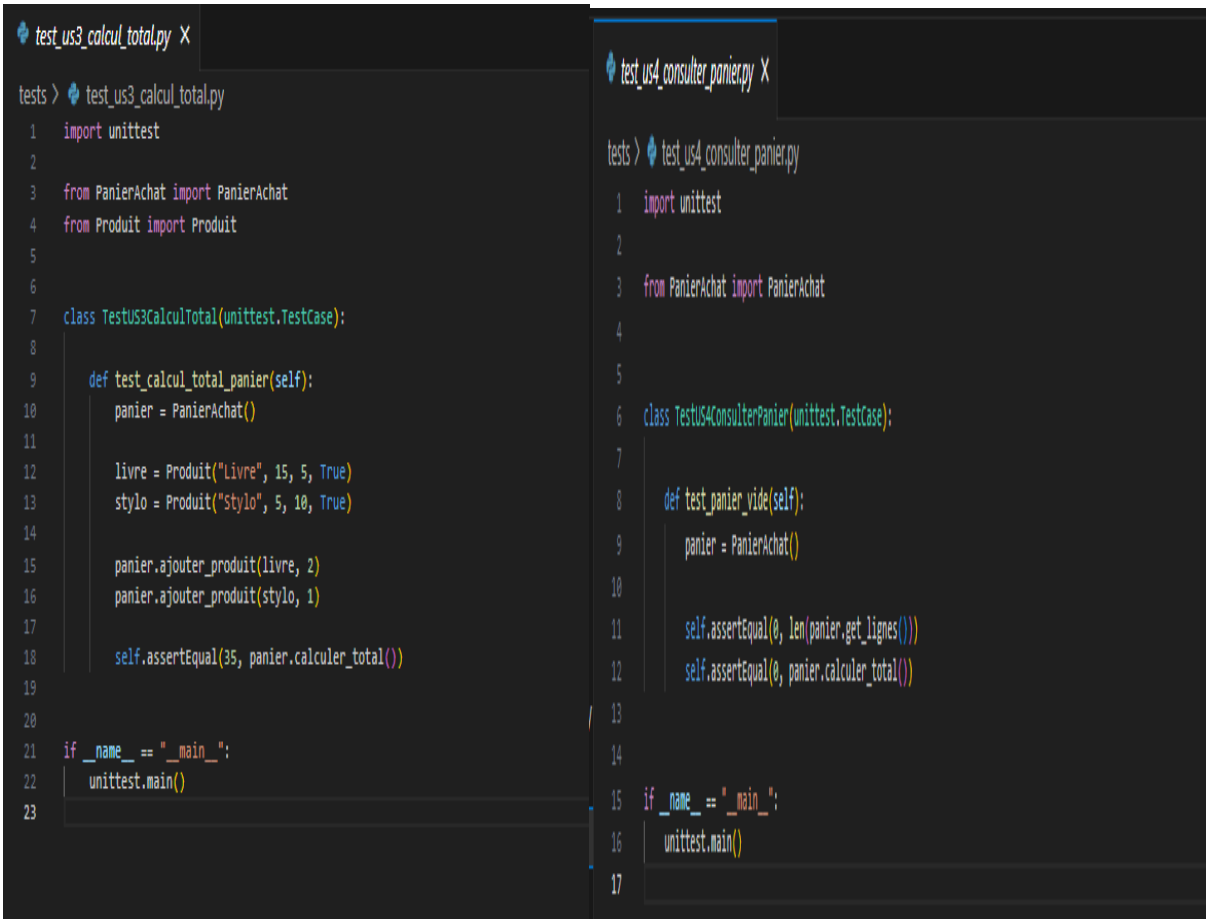
Les étapes définies dans les fichiers .feature ont été reliées au code Python via des **steps Behave**.
Chaque étape appelle directement les méthodes du code métier afin de valider le comportement du système.



2.4- Test

Ces tests permettent de garantir la cohérence fonctionnelle et la fiabilité du panier d'achat.





test_us3_calcul_total.py

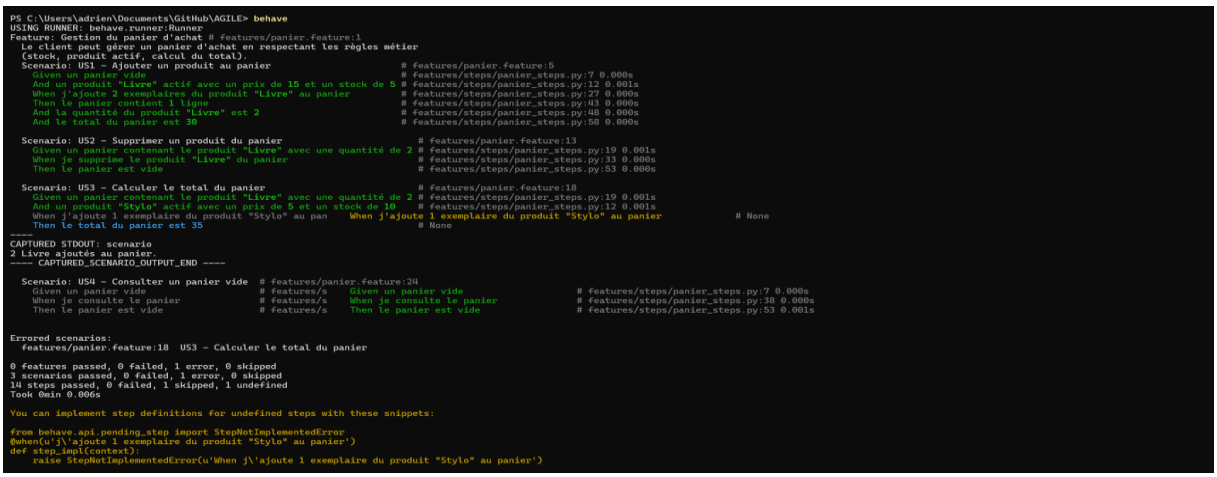
```python
import unittest

from PanierAchat import PanierAchat
from Produit import Produit


class TestUS3CalculTotal(unittest.TestCase):

    def test_calcul_total_panier(self):
        panier = PanierAchat()

        livre = Produit("Livre", 15, 5, True)
        stylo = Produit("Stylo", 5, 10, True)

        panier.ajouter_produit(livre, 2)
        panier.ajouter_produit(stylo, 1)

        self.assertEqual(35, panier.calculer_total())


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

test_us4_consulter_panier.py

```python
import unittest

from PanierAchat import PanierAchat


class TestUS4ConsulterPanier(unittest.TestCase):

    def test_panier_vide(self):
        panier = PanierAchat()

        self.assertEqual(0, len(panier.get_lignes()))
        self.assertEqual(0, panier.calculer_total())


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

3. RUN

Exécution des tests fonctionnels avec Behave.
Les scénarios US1, US2 et US4 sont validés avec succès.
Le scénario US3 échoue initialement en raison d’une étape non définie dans les steps Behave (1 exemplaire au singulier).
Le problème a été résolu en généralisant la définition de l’étape afin de gérer à la fois le singulier et le pluriel.



```
PS C:\Users\adrien\Documents\GitHub\AGILE> behave
USING RUNNER: behave.runner:Runner
Feature: Gestion du panier d'achat # features/panier.feature:1
  Le client peut gérer un panier d'achat en respectant les règles métier
  (stock, produit actif, calcul du total).
  Scenario: US1 - Ajouter un produit au panier                    # features/panier.feature:5
    Given un panier vide                                          # features/steps/panier_steps.py:7 0.000s
    And un produit "Livre" actif avec un prix de 15 et un stock de 5 # features/steps/panier_steps.py:12 0.001s
    When j'ajoute 2 exemplaires du produit "Livre" au panier      # features/steps/panier_steps.py:27 0.000s
    Then le panier contient 1 ligne                               # features/steps/panier_steps.py:43 0.000s
    And la quantité du produit "Livre" est 2                      # features/steps/panier_steps.py:48 0.000s
    And le total du panier est 30                                 # features/steps/panier_steps.py:58 0.000s

  Scenario: US2 - Supprimer un produit du panier                  # features/panier.feature:13
    Given un panier contenant le produit "Livre" avec une quantité de 2 # features/steps/panier_steps.py:19 0.001s
    When je supprime le produit "Livre" du panier                 # features/steps/panier_steps.py:33 0.000s
    Then le panier est vide                                       # features/steps/panier_steps.py:53 0.000s

  Scenario: US3 - Calculer le total du panier                     # features/panier.feature:18
    Given un panier contenant le produit "Livre" avec une quantité de 2 # features/steps/panier_steps.py:19 0.001s
    And un produit "Stylo" actif avec un prix de 5 et un stock de 10 # features/steps/panier_steps.py:12 0.001s
    When j'ajoute 1 exemplaire du produit "Stylo" au pan    When j'ajoute 1 exemplaire du produit "Stylo" au panier          # None
    Then le total du panier est 35                                # None
____
CAPTURED STDOUT: scenario
2 Livre ajoutés au panier.
---- CAPTURED_SCENARIO_OUTPUT_END ----

  Scenario: US4 - Consulter un panier vide  # features/panier.feature:24
    Given un panier vide                    # features/s    Given un panier vide                  # features/steps/panier_steps.py:7 0.000s
    When je consulte le panier              # features/s    When je consulte le panier            # features/steps/panier_steps.py:38 0.000s
    Then le panier est vide                 # features/s    Then le panier est vide               # features/steps/panier_steps.py:53 0.001s


Errored scenarios:
  features/panier.feature:18  US3 - Calculer le total du panier

0 features passed, 0 failed, 1 error, 0 skipped
3 scenarios passed, 0 failed, 1 error, 0 skipped
14 steps passed, 0 failed, 1 skipped, 1 undefined
Took 0min 0.006s

You can implement step definitions for undefined steps with these snippets:

from behave.api.pending_step import StepNotImplementedError
@when(u'j\'ajoute 1 exemplaire du produit "Stylo" au panier')
def step_impl(context):
    raise StepNotImplementedError(u'When j\'ajoute 1 exemplaire du produit "Stylo" au panier')
```

```
PS C:\Users\adrien\Documents\GitHub\AGILE> behave
USING RUNNER: behave.runner:Runner
Feature: Gestion du panier d'achat # features/panier.feature:1
  Le client peut gérer un panier d'achat en respectant les règles métier
  (stock, produit actif, calcul du total).
    And le total du panier est 30                          # features/steps/panier_steps.py:60 0.000s
    And le total du panier est 30                          # features/steps/panier_steps.py:60
    Then le panier est vide                                   # features/steps/panier_steps.py:55 0.000s
    Then le panier est vide                                   # features/steps/panier_steps.py:55
    Then le total du panier est 35                            # features/steps/panier_steps.py:60 0.000s
    Then le total du panier est 35                            # features/steps/panier_steps.py:60
  Scenario: US4 - Consulter un panier vide  # features/panier.feature:24
    Given un panier vide                    # features/steps/panier_steps.py:7 0.001s
    When je consulte le panier              # features/steps/panier_steps.py:40 0.000s
    Then le panier est vide                 # features/steps/panier_steps.py:55 0.000s

1 feature passed, 0 failed, 0 skipped
4 scenarios passed, 0 failed, 0 skipped
16 steps passed, 0 failed, 0 skipped
Took 0min 0.006s
```

Les quatre tests unitaires correspondant aux User Stories US1 à US4 ont été exécutés avec succès.
Tous les tests sont validés, confirmant le bon fonctionnement des méthodes et des règles métier du panier d'achat.

```
PS C:\Users\adrien\Documents\GitHub\AGILE> python -m unittest discover -s tests -p "test_*.py"
2 Livre ajoutés au panier.
.2 Livre ajoutés au panier.
Livre supprimé du panier.
.2 Livre ajoutés au panier.
1 Stylo ajoutés au panier.
..
----------------------------------------------------------------------
Ran 4 tests in 0.001s

OK
```