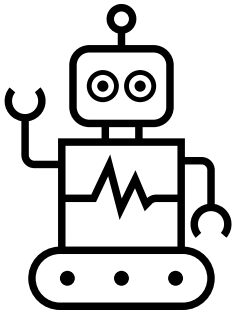
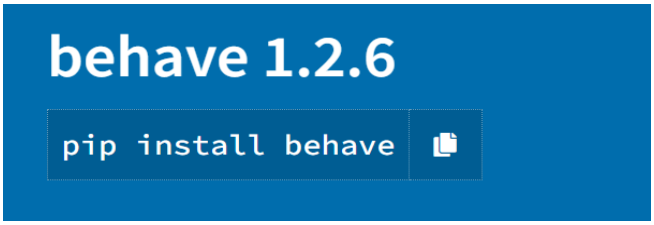
User stories : Bobo, Coco sur InstaBo. L’aventure continue !



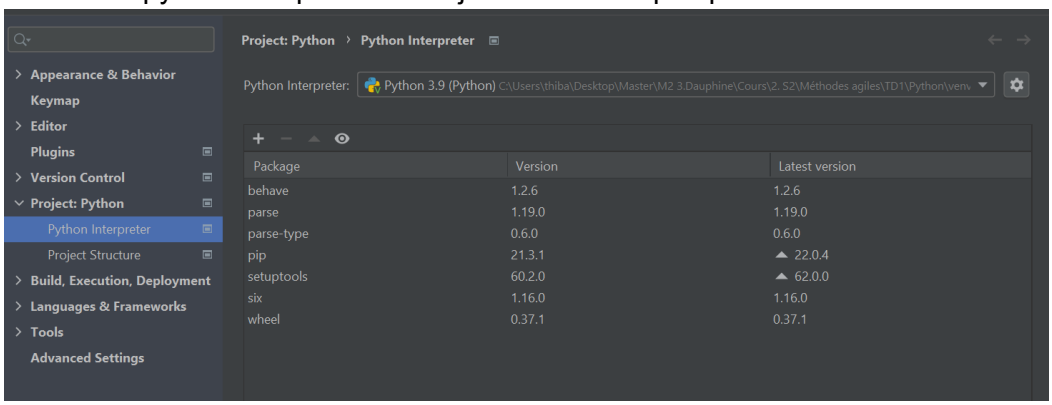
Thibault Bougerol, Bastien Lesouef

Dans ce court tutoriel nous allons voir comment créer des user stories. Une **user storie** est une description simple du besoin fonctionnel pour répondre à un problème utilisateur.

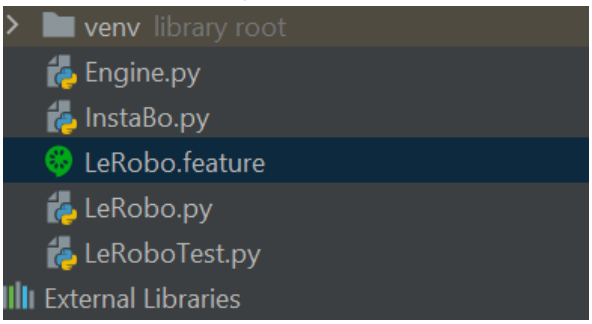
Tout d'abord, télécharger Behave en ouvrant la commande et en tapant "pip install behave":



Ensuite, se rendre dans l'IDE Pycharm et l'ajouter en tant qu'interpréteur :
Appuyer sur Ctrl + Alt + S :
Aller dans python interpréteur et l'ajouter s'il n'est pas présent :



Ensuite, créer un fichier .feature, Pycharm va nous proposer automatiquement de télécharger le plug in nécessaire (Gherkin)



Créer les stories dans le fichier feature : On veut **vérifier que nos Robo peuvent bel et bien se parler s'ils sont inscrits au même réseau social.**
On va pour cela les créer, leur donner un nom à chacun et tenter de faire parler le premier LeRobo au deuxième LeRobo. Ce que l'on veut tester sont les scénarios suivants :
-Si deux robots sont connectés au même réseau Instabo, alors ils peuvent se parler
-S'ils ne sont pas connectés sur le même réseau (scénario 2), alors ils ne peuvent pas se parler.

Ainsi, ces deux tests présents dans Outline doivent renvoyer True puis False et c'est ce qui est spécifié dans les variables de fin (talk).

ATTENTION : Les valeurs renseignées pour les variables "connected" et "talk" sont lues comme des strings. Ainsi, si l'on veut tester la variable dans un if, il faut bien penser à la tester en tant que string.

```
Scenario: Discussion in Instabo
  Given a robo robo1 and a robo robo2
  When we are connected to the same social network
  Then we can talk

Scenario Outline:Outline
  Given a robo and another
  When they are connected to the same network <connected>
  Then they can <talk>
  Examples:
    | connected | talk  |
    | False     | False |
    | True      | True  |
```

On spécifie ensuite nos tests fonctionnels :

```
from LeRobo import LeRobo
from InstaBo import InstaBo
from behave import *
from dataclasses import dataclass


@dataclass
class ManageLeRoboSteps:
    _robo1 = LeRobo()
    _robo2 = LeRobo()
    _result = False
```

```python
@given("a robo robo1 and a robo robo2")
def two_robots_robo1_robo2(self):
    self.robo1 = LeRobo()
    self.robo1.set_name("robo1")
    self.robo2 = LeRobo()
    self.robo2.set_name("robo2")
    names = self.robo1 != "unknown" and self.robo2 != "unknown"
    assert names
```

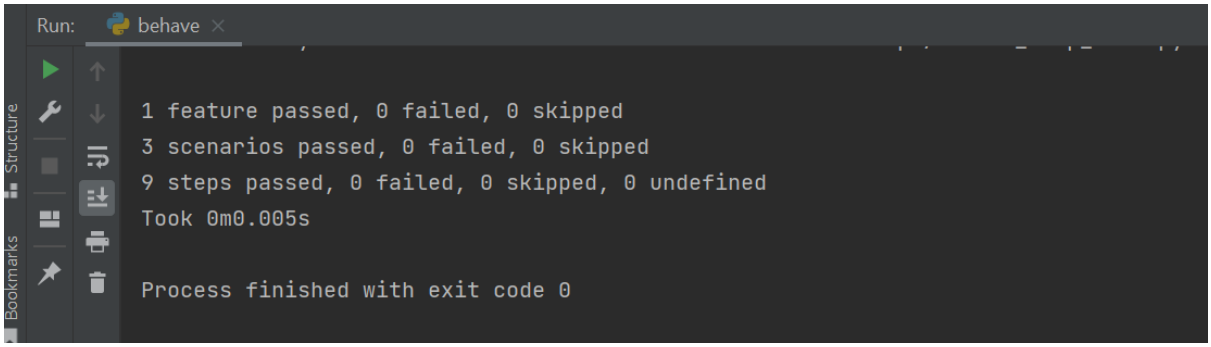
```python
@when("we are connected to the same social network")
def robo1_and_robo2_connect_to_instabo(self):
    self.insta = InstaBo()
    self.robo1.subscribe_to_instabo(self.insta)
    self.robo2.subscribe_to_instabo(self.insta)
    assert hasattr(self.robo1, 'insta') and hasattr(self.robo2, 'insta')

@then("we can talk")
def robo1_and_robo2_talk(self):
    self._result = self.robo1.robo_talks(self.robo2)
    assert self._result
```

```python
@given("a robo and another")
def two_robots(self):
    self.robo1 = LeRobo()
    self.robo1.set_name("robo1")
    self.robo2 = LeRobo()
    self.robo2.set_name("robo2")
    names = self.robo1 != "unknown" and self.robo2 != "unknown"
    assert names
```

```python
@when("they are connected to the same network {connected}")
def two_robots_connexion(self, connected):
    if connected == "True":
        self.insta = InstaBo()
        self.robo1.subscribe_to_instabo(self.insta)
        self.robo2.subscribe_to_instabo(self.insta)
        assert hasattr(self.robo1, 'insta') == True and hasattr(self.robo2, 'insta') == True
    else:
        assert hasattr(self.robo1, 'insta') == False and hasattr(self.robo2, 'insta') == False
```

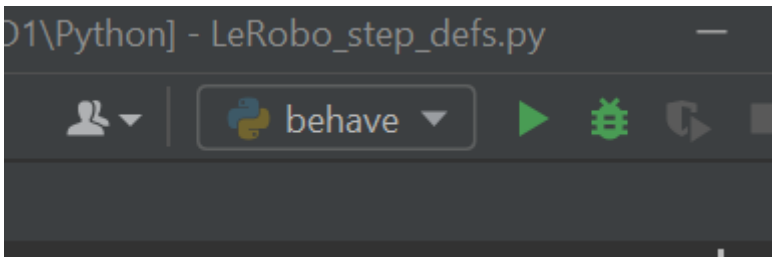
```python
@then("they can {talk}")
def two_robots_talk(self, talk):
    self._result = self.robo1.robo_talks(self.robo2)
    print("For this test, the wanted result is {} and the output is {}".format(talk, self._result))
    assert str(self._result) == str(talk)
```

Une fois tout renseigné, grâce à un tour de passe passe (voir ci-après) et quelques recherches stackoverflow il est possible de run le fichier *.feature* sur Pycharm community (c'est une option facilement utilisable uniquement pour Pycharm pro):

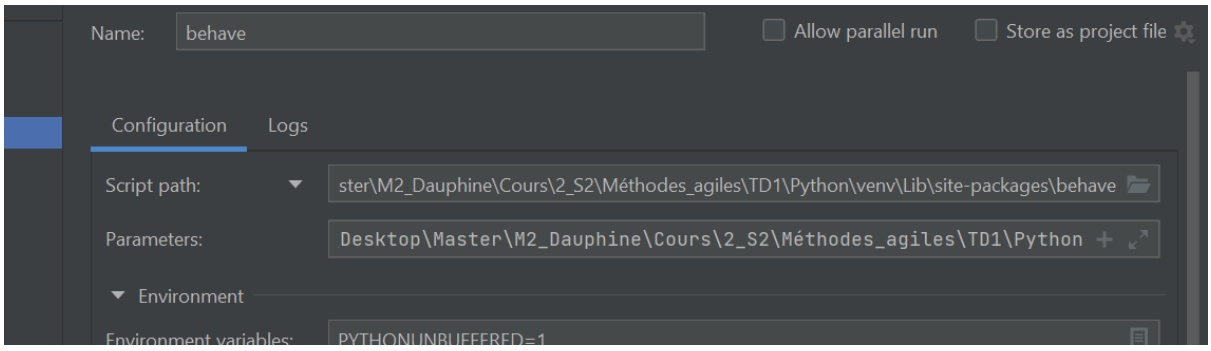


On voit que nos 3 scénarios se sont bien déroulés et que tous nos tests sont validés!

Méthode pour lancer les fichiers .feature sur Pycharm community :
https://stackoverflow.com/questions/40520301/how-to-run-a-feature-file-using-pycharm-community
Cliquez en haut à droite et configurer un interpréteur :



On met comme nom behave, on change le script path en mettant le path de l'interpréteur behave et on met comme paramètre le path de notre dossier où se trouve notre fichier *.feature.*
Attention, Pycharm n'accepte pas les paths avec des espèces (oui oui, vous avez bien lu), donc veillez à changer vos noms de dossiers avant…



Merci d'avoir suivis ce tutoriel. On se voit bientôt pour de nouvelles aventures