TD : introduction à Python et PyCharm

Romain Tavenard

Le but de cette séance est de réaliser vos premiers programmes en Python dans l'IDE (*Integrated Development Environment*) PyCharm. Sachez que la documentation Python est de très bonne qualité : utilisez-la (https://docs.python.org/3/tutorial/).

1 Avant-propos

Lors de ce TD, vous allez créer, sur votre disque `M:/`, un répertoire dans lequel vous travaillerez **tout au long du semestre**. Sauf indication contraire, vous créerez, **pour chaque séance de TD, un nouveau fichier Python** dans lequel vous écrirez votre code. Pour ce premier TD, vous serez guidés (un peu plus bas dans cet énoncé) dans la création de votre répertoire/projet Python et du fichier Python correspondant au TD1.

De plus, vous prendrez la bonne habitude de renseigner, en commentaire de votre code, les tests effectués pour vérifier le bon fonctionnement de votre code. Par exemple, plutôt que :

```
x = input("Entre une valeur :")
print(12)
```

vous écrirez :

```
x = input("Entre une valeur :")
print(x)
# [Entrée] Entre une valeur : 12
# [Sortie] 12
```

2 Échauffement

1. Écrivez **sur papier**, en pseudo-code, un algorithme permettant, étant données 3 longueurs `a`, `b` et `c` d'afficher s'il est ou non possible de construire un triangle ayant ces longueurs pour côtés et, le cas échéant, si ce triangle

est équilatéral, isocèle, rectangle ou quelconque. **Attention :** un triangle peut être à la fois isocèle et rectangle.

3 Prise en main de PyCharm

Démarrez PyCharm (qui se trouve dans le dossier Développement du menu Démarrer, sous l'item JetBrains) et créez un nouveau projet nommé L2_Python dans un endroit bien identifié (vous utiliserez ce projet tout au long du semestre) de votre disque `M:/`. Pour ce projet, sélectionnez l'interpréteur `Python 3.4.0` (dont le chemin est `C:/Python34/python.exe` sur les ordinateurs de l'Université).

Ajoutez le répertoire `data/` (dans lequel se trouvera le contenu dans le fichier de données mis à votre disposition sur CURSUS) à votre projet. Pour cela, il suffit de décompresser l'archive dans le répertoire de votre projet (en utilisant votre gestionnaire d'archive préféré) et le dossier apparaîtra dans l'arborescence de votre projet dans PyCharm.

Ajoutez un nouveau fichier Python à votre projet, que vous nommerez `td1`.

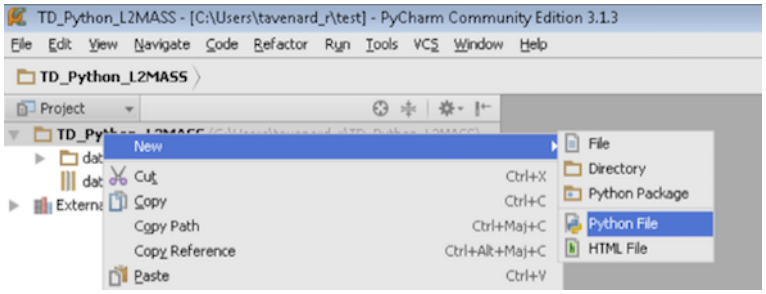

FIGURE 1 –

Dans ce nouveau fichier, entrez le morceau de code suivant :

```
print("Hello world !")
```

2. Que fait ce programme ? Vérifiez-le en l'exécutant.

En exécutant pour la première fois ce programme, vous avez créé dans PyCharm une configuration d'exécution (*Run Configuration*). Ainsi, il vous suffira par la suite de sélectionner cette configuration d'exécution dans la liste disponible en haut à droite de votre fenêtre PyCharm puis de cliquer sur le bouton *Run* (triangle vert).

Lorsque vous souhaiterez utiliser le débogueur de PyCharm, il suffira de cliquer sur l'icône symbolisant un insecte (sémantique originale du terme *bug*) au lieu de l'icône *Run*. Pour utiliser ce débogueur, plusieurs possibilités s'offrent à vous :

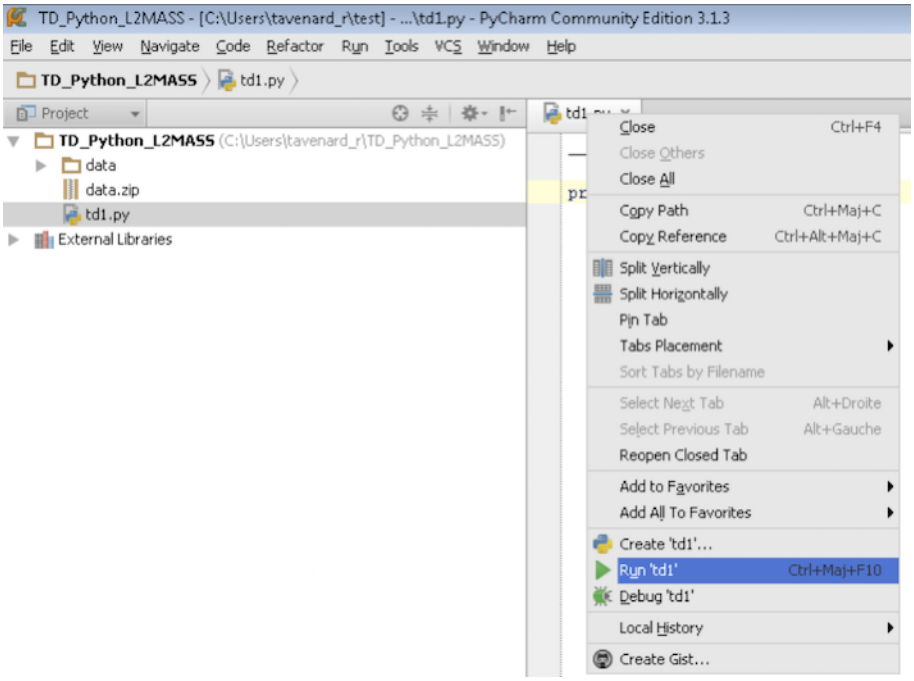

FIGURE 2 –

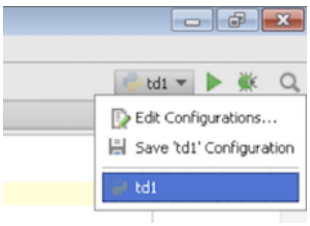

FIGURE 3 –

a. Vous souhaitez comprendre pourquoi un programme “plante” (*ie*, affiche une exception et termine) : lancez le débogueur et l'exécution s'interrompra au plantage voulu ;
b. Vous souhaitez examiner les valeurs prises par certaines variables au cours de l'exécution : vous pouvez alors disposer des points d'arrêts dans votre code (en cliquant dans la marge en face de la ligne concernée) et l'exécution s'interrompra dans ce cas à chaque fois qu'un point d'arrêt sera rencontré.

3. Utilisez le débogueur pour analyser le comportement du programme suivant (que vous aurez copié-collé dans votre fichier `td1.py`) et repérer les différents éléments de la console du débogueur :

```
x = 12
y = 2 * X
print(y)
```

Un autre outil très utile pour développer en Python dans PyCharm est la console Python. Comme vous le savez, il est possible d'exécuter des morceaux de code Python directement dans la console Python pour observer leur effet sans avoir à les inclure dans un programme (ou script Python).

4. Pour lancer la console Python, rendez vous dans le menu Outils (Tools) et choisissez l'item correspondant. Entrez quelques lignes de code élémentaire pour vous familiariser avec cette console.

4 Variables

Le langage Python est un langage fortement typé, ce qui signifie que chaque variable, à chaque instant, ne peut être que d'un type. Par contre, ce typage est dynamique, ce qui signifie qu'à la déclaration d'une variable, il n'est pas nécessaire de déclarer son type : c'est la valeur qu'on affectera à la variable qui déterminera son type. La déclaration d'une variable est donc une instruction du type :

```
ma_variable = 12
```

Il est possible de connaître le type d'une variable en utilisant la fonction `type` :

```
print(type(ma_variable))
```

5. Déclarez une variable `ma_variable` et affectez-lui la valeur `2`. Quel est le type de cette variable ? Répétez l'opération avec les valeurs suivantes :

— `2.5`
— `2.` (notez le caractère `.` final)
— `2 + 2.5`

— `'a'`
— `10 / 3`
— `10 / 2`

Il est possible, en Python, d'affecter des valeurs à plusieurs variables en même temps :

```
premiere_variable, deuxieme_variable = 12, "abc"
```

6. Utilisez cette astuce pour intervertir les valeurs de deux variables en une seule ligne de code. Pour afficher les valeurs des deux variables, on pourra utiliser la syntaxe :

```
print("{} {}".format(premiere_variable, deuxieme_variable))
```

5 Structures conditionnelles

La syntaxe des structures conditionnelles en Python est de la forme suivante :

```
if condition1:
    # [...]
elif condition2:
    # [...]
else:
    # [...]
```

7. Stockez, dans une variable `age`, un entier entré par l'utilisateur, puis affichez l'une des phrases suivantes, en fonction de la valeur stockée dans `age` :

— La classe d'âge est 0-19
— La classe d'âge est 20-29
— La classe d'âge est 30+

Pour récupérer une valeur entrée par l'utilisateur, on peut utiliser la fonction `input` :

```
valeur_entree = input("Entrez votre âge : ")
```

Pour pouvoir comparer cette valeur aux bornes des différents intervalles, il sera nécessaire de la convertir en entier (vous pouvez tester le type de cette variable pour vous en rendre compte) à l'aide de la fonction `int` qui prend en entrée une chaîne de caractères et la transforme, si cela est possible, en l'entier qu'elle représente.

6 Boucles

8. Écrivez une boucle qui permette d'afficher les valeurs contenues dans une liste. La sortie affichée dans la fenêtre *Run* de PyCharm devra être de la forme (pour une liste définie par `lst = [1, 7, 5, 3, 6]`) :

```
La liste lst contient : 1
La liste lst contient : 7
La liste lst contient : 5
La liste lst contient : 3
La liste lst contient : 6
```

Pour afficher une chaîne de caractères composée d'une partie de texte fixe et d'une autre correspondant au contenu d'une variable, on utilise la méthode `format()`, comme dans l'exemple suivant :

```
prenom = "Antoine"
nom = "Griezmann"
annee_naissance = 1991
print("{} {} est né en {}".format(prenom, nom, annee_naissance))
```

9. Écrivez une nouvelle boucle affichant, avec les valeurs contenues dans une liste, leur indice. On obtiendra alors une sortie du type :

   ```
   La liste l contient 1 à l'indice 0
   La liste l contient 7 à l'indice 1
   La liste l contient 5 à l'indice 2
   La liste l contient 3 à l'indice 3
   La liste l contient 6 à l'indice 4
   ```

10. Écrivez une nouvelle boucle affichant les carrés des valeurs contenues dans une liste. On obtiendra alors une sortie du type :

    ```
    1^2=1
    7^2=49
    5^2=25
    3^2=9
    6^2=36
    ```

11. Écrivez une nouvelle boucle permettant de calculer la somme des `n` premiers entiers. Notez que `range(1, n + 1)` retourne une liste contenant les `n` premiers entiers.

Dans la suite, les exemples présentés seront basés sur des boucles **`for`**, mais les mots-clés **`break`** et **`continue`** sont également utilisables avec des boucles **`while`**, exactement de la même façon. Lors de l'exécution d'une boucle, il est possible de vouloir en sortir si une certaine condition est remplie. Cela se fait à l'aide de l'instruction **`break`** :

```
for v in liste_valeurs:
    if condition:
        break
    # [...]
```

De même, il est possible de vouloir, à certaines étapes de l'exécution, ne rien faire et passer à l'itération suivante :

```
for v in liste_valeurs:
    if condition:
        continue
    # [...]
```

12. Parcourez, à l'aide d'une boucle, la liste des 30 premiers entiers et affichez un message pour chaque entier pair (pour tester la parité d'un entier, on s'intéressera au reste de sa division entière par deux). Lorsque la valeur 20 est dépassée, imposez de sortir de la boucle.

7 Exercice de synthèse

13. Écrivez un programme qui convertit une note scolaire entrée par l'utilisateur (on vérifiera que celle-ci est bien comprise entre 0 et 20 et on affichera un message d'erreur sinon) en appréciation selon le barème suivant :

Note	Appréciation
$N \geq 15$	A
$10 \leq N < 15$	B
$5 \leq N < 10$	C
$N < 5$	D