TP – Environnement de travail, premiers pas

Objectifs

- S'approprier l'environnement de travail Spyder et l'installer à la maison. Distinguer code source (visible dans l'éditeur) et programme exécutable (dont les résultats peuvent s'observer dans la console)
- Distinguer l'interprétation en mode interactif (console) et en mode différé (script)
- Comprendre la notion de variable (= contenant)
- Connaître à chaque instant le type des valeurs manipulées
- Repérer la portée d'une variable
- Définir une fonction
- Connaître la signature des fonctions que l'on écrit
- Appeler une fonction avec des valeurs d'entrée choisies
- Demander l'affichage d'un résultat dans la console
- Rendre son code source lisible et modulaire en regroupant chaque tâche dans une fonction.
- Comprendre comment imbriquer des appels de fonctions et la portée des variables dans ce contexte
- Respecter les usages de nommage, d'écriture de code et de documentation des fonctions (signature, préconditions, documentation).

1. Découvrir l'environnement de travail

1.1 Le réseau du lycée et l'arborescence de fichiers

Vous disposez d'un accès au réseau pédagogique du lycée Montaigne. Cet accès vous donne notamment accès à un espace de stockage personnel en réseau, que vous pouvez retrouver depuis n'importe quel ordinateur du lycée.

Question 1. Se connecter au réseau à l'aide de votre *identifiant* et de votre *mot de passe* personnels. En général, votre identifiant est *prenom.nom* sans accent, et votre mot de passe est votre date de naissance au format *jj/mm/aaaa*.

Question 2. Repérer votre *dossier personnel*. C'est dans ce dossier que vous pouvez enregistrer vos fichiers de travail. Créez dès maintenant un dossier *ITC*, puis un sous-dossier `TP0`. Où sont physiquement stockés ces fichiers ?

Question 3. Repérer également le *dossier partagé* de la classe. Il est accessible en lecture aux étudiants et en lecture/écriture aux enseignants. Vous y trouverez, pour certaines séances, des fichiers déposés par les professeurs.

Question 4. Pour cette séance, repérez le fichier `tp0.py`, puis le copier dans votre dossier personnel, dans le sous-dossier `TP0`.

1.2 Installer la distribution scientifique Anaconda

Pour travailler en Python, nous utiliserons une distribution scientifique appelée *Anaconda*. Elle installe l'interpréteur Python ainsi qu'un ensemble d'outils utiles pour le calcul scientifique et la programmation, en particulier l'environnement de développement *Spyder*.

Remarque – Qu'est-ce qu'une distribution Python ?

Une *distribution Python* est un ensemble cohérent comprenant :

- l'interpréteur de langage Python, dont le rôle est de traduire votre code Python en instructions machine ;
- des bibliothèques (appelés *packages*) qui contiennent des fonctions utiles déjà écrites par d'autres programmeurs du monde entier ;
- des outils de développement (éditeur de texte pour écrire le code source, outils de débogage, console pour exécuter et lire les résultats affichés).

Anaconda est disponible pour les systèmes d'exploitation Windows, MacOS et Linux.

Sous Windows. Télécharger l'installateur d'Anaconda pour Windows depuis le site officiel d'Anaconda. Lancer l'installateur, puis conserver les options proposées par défaut. Une fois l'installation terminée, ouvrir *Anaconda Navigator* depuis le menu Démarrer.

Sous MacOS. Télécharger l'installateur d'Anaconda pour macOS depuis le site officiel d'Anaconda. Choisir la version adaptée à votre ordinateur, selon qu'il utilise une puce Apple Silicon ou Intel. Lancer l'installation, puis ouvrir *Anaconda Navigator* depuis le dossier Applications ou depuis le Launchpad.

Sous Linux. Dans un Terminal de commande, tapez

```
sudo apt install spyder
```

Entrez votre mot de passe adminstrateur et répondez `O` (oui) à toutes les questions. Pour lancer `spyder`, il suffit de taper

```
spyder
```

dans un terminal

Question 5. Votre premier travail pour la semaine prochain : installez Anaconda sur votre ordinateur personnel. Si vous ne possédez pas d'ordinateur personnel, et souhaitez être conseillé ou aidé (prêt), n'hésitez pas à contacter au plus tôt Mme OLIVIER (geraldine.olivier@ac-bordeaux.fr). Avoir un ordinateur personnel vous permettra de vous entrainer régulièrement (en ITC, mais aussi en mathéamtiques, en physique-chimie, en SI, en option informatique et en TIPE).

Remarque – Et au lycée ?

Sur les ordinateurs du lycée, Anaconda et Spyder peuvent déjà être installés. Dans ce cas, il est inutile de refaire l'installation : il suffit de lancer Spyder.

1.3 Utiliser Spyder

Spyder est un environnement de développement intégré (IDE, *Integrated Development Environment*) pour Python. Il facilite la vie du programmeur en mettant à sa disposition :

- un éditeur de texte simple pour écrire le code source
- une interface graphique permettant de lancer facilement l'interprétation du code par l'interpréteur Python
- une console qui permet d'interagir avec le programme en cours d'exécution
- des outils d'analyse de l'exécution (suivi de variables) et de débogage

Question 6. Repérer les différents outils intégrés dans l'interface Spyder :

- Ouvrir *Anaconda Navigator*.
- Dans la fenêtre d'accueil, repérer *Spyder*, puis cliquer sur *Launch*.
- Après quelques instants, Spyder ouvre une fenêtre contenant plusieurs zones. Les deux zones les plus importantes sont :
  - l'*éditeur*, dans lequel on écrit le code source en Python ;
  - la *console*, dans laquelle on peut lancer l'exécution d'instructions Python et observer les résultats.

Question 7. La console permet de lancer l'interprétation d'instructions écrites en Python et de visualiser le résultat de l'exécution. Taper successivement les instructions suivantes dans la console, en appuyant à chaque fois sur la touche Entrée pour lancer l'exécution :

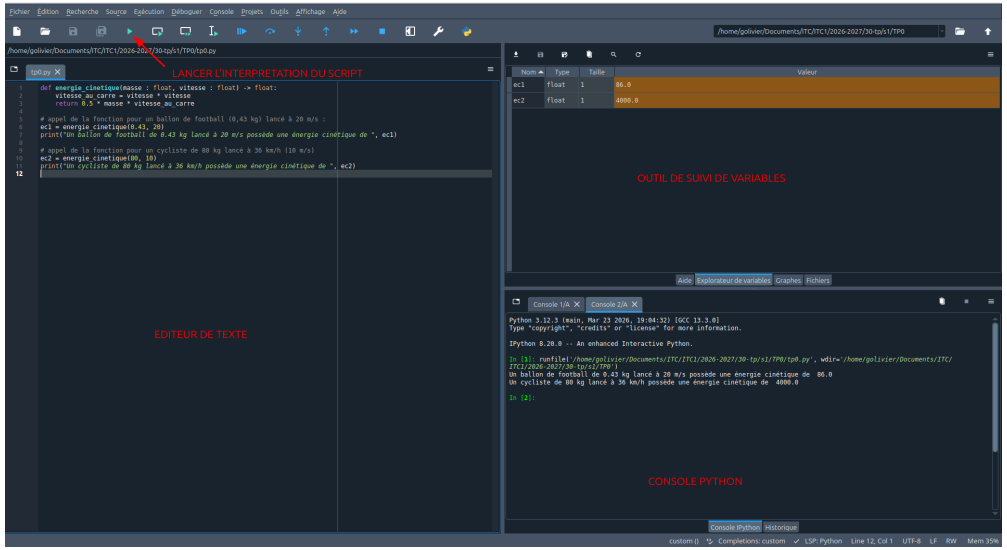

FIGURE 1 – L'environnement de développement *Spyder*

```
2 + 3
5 * 7
210
```

Dans la console, le résultat final de l'exécution est affiché. On dit que l'on procède en *mode interactif* : je demande à l'interpréteur Python d'exécuter une instruction, il me répond directement.

Question 8. L'éditeur permet d'écrire une suite d'instructions sous la forme d'un code source en langage Python. On parle souvent de script Python plutôt que de code source Python. Ce code source est enregistré dans un fichier comme un simple texte au format ASCII, généralement en utilisant l'extension `.py` pour que l'on sache qu'il s'agit d'un script Python. Ouvrir dans l'éditeur le fichier *tp0.py* que vous avez copié dans votre dossier personnel et visualiser les instructions Python écrites dans ce fichier.

Si l'on souhaite que l'interpréteur exécute d'un coup toutes les instructions contenues dans le script Python, on utilise le bouton d'exécution ou la touche *F5*. Toutes les instructions du fichier sont alors exécutées et seules les valeurs explicitement pour lesquelles le programmeur a demandé explicitement un affichage avec la fonction `print` apparaissent dans la console. On dit que l'on procède en *mode non intéractif* : je demande à l'interpréteur Python d'exécuter toute une série d'instructions : il traite

toutes les instructions de manière séquentielle (c'est-à-dire les unes à la suite des autres) et ne me donne pas de *feed-back* à chaque instruction.
Par exemple, si un fichier contient l'instruction suivante :

```
a = 3 + 5
b = 2*a
print("La variable b contient la valeur", b)
```

alors l'exécution du fichier affiche :

```
La variable b contient la valeur 16
```

En revanche, si un fichier contient seulement :

```
a = 3 + 5
b = 2*a
```

les instruction sont bien exécutées, mais aucun affichage n'apparaît dans la console.

Question 9. Exécuter l'intégralité du script *tp0.py*. Comment faire pour observer un résultat.

Remarque – Interprétation intéractive console v.s interprétation non interactive script

Il faut donc bien distinguer deux usages :

en mode intéractif, on teste rapidement une instruction dans la console, sans la sauvegarder, et le résultat de l'exécution est affiché directement ;

en mode non intéractif, on écrit généralement plusieurs instructions dans l'éditeur et on enregistre ce code source dans un fichier. Pour qu'une valeur ou un résultat soit affiché lors de l'exécution, celui-ci doit être demandé explicitement dans le script en utilisant la fonction `print`.

Dans le cadre de ce cours, pour garder une trace de vos tests notamment, vous utiliserez généralement le mode non intéractif. Cela permettra d'écrire vos tests directement dans votre script Python, de sorte que, à partir de votre fichier e code source, vos enseignants puissent exécuter vos tests et vous aider.

Question 10. Il est également possible d'exécuter seulement une partie d'un fichier : sélectionner uniquement la définition de fonction, le premier appel et l'affichage de son résultat dans le script `tp0.py`, puis utiliser la commande permettant d'exécuter uniquement les instructions sélectionnées.

Question 11. Analyser le script `tp0.py`.

- **11.1.** Quelle est la portée de la variable `vitesse_au_carre` ? Quelle est son type ? Comment qualifie-t-on cette variable ?
- **11.2.** Quelle est la portée de la variable `ec1` ?
- **11.3.** Quelle est la signature (on dit aussi le type) de la fonction `energie_cinetique` ?
- **11.4.** Que se passe-t-il lors de l'exécution de l'appel `energie_cinetique(1, 3)` ?

Question 12. Écrire un autre appel de la fonction.

- **12.1.** Écrire dans la console un appel de fonction pour calculer l'énergie cinétique d'un TGV de 400 tonnes lancé à 300 km/h.
- **12.2.** Écrire le même appel dans le script et lancer l'exécution uniquement des nouvelles instructions écrites. Observer le résultat dans la console. Que faut-il rajouter pour observer quelque chose lors de l'exécution ?

Question 13. Apprivoiser les outils de suivi et de débogage.

- **13.1.** Positionner des points d'arrêts à la fin des instructions des lignes 2 et 6 avec `F12`.
- **13.2.** Exécuter le script en mode débogage et observer pas à pas l'affichage des variables.

2. A vous de jouer

Remarque – Sauvegarder très régulièrement !!

Sauvegardez très très régulièrement votre fichier de code source (au minimum toutes les 5 minutes) pour éviter toute déconvenue en cas de panne réseau ou de mauvaise manipulation. Vous pouvez le faire sans lever les mains du clavier en appuyant simultanément sur la touche `Ctrl` et sur la touche `S` (comme *save*).

Ex. 1 – Changement d'unités

Question 1. Écrire une fonction qui permet de convertir une vitesse donnée en km/h en m/s.

Question 2. Utiliser cette fonction pour calculer l'énergie cinétique d'une voiture de $800\,kg$ qui roule à $50\,km/h$. Attention, on ne modifiera pas la fonction `energie_cinetique`.

Ex. 2 – Énergie cinétique d'un objet sphérique plein

Question 1. On souhaite écrire une fonction `calcul_masse_boule` qui calcule la masse d'une boule de rayon r constituée d'un matériau homogène dont la masse volumique est connue. Donner le diagramme entrée-sortie de cette fonction (avec tous les types et les éventuelles préconditions).

Question 2. Écrire cette fonction `calcul_masse_boule` en Python. On rappelle que le volume d'une sphère de rayon r est $\frac{4}{3}\pi r^3$. On peut obtenir une valeur très précise de π dans le *package* (bibliothèque) `math` en rajoutant au tout début du code l'instruction

```
from math import pi
```

qui va rendre visible une variable globale nommée `pi` contenant une valeur précise de π.

Question 3. Écrire plusieurs tests et les exécuter pour valider votre fonction.

Question 4. En utilisant les deux fonctions `calcul_masse_boule` et `energie_cinetique`, écrire une troisième fonction `energie_cinetique_boule` qui calcule l'énergie cinétique d'un objet plein homogène sphérique de rayon r, de densité volumique ρ et de vitesse v. On n'hésitera pas à utiliser des variables intermédiaires bien nommées pour faciliter la lisibilité du code.

Question 5. Écrire plusieurs tests et les exécuter pour valider votre fonction.