→ Lancer le programme “Spyder.exe” et revenez lire la suite le temps de patienter...

I Environnement de travail

1.1 Les dossiers réseau

Vous disposez d’un espace de travail sur le réseau du lycée. Il s’agit , dans l’explorateur de fichier, d’un disque portant le même nom que votre login. Vous devrez enregistrer les fichiers nécessaires au travail en TD d’informatique dans ce dossier, et pas sur la machine locale (ie. pas sur le bureau ou dans tout autre sous-dossier de C:).

Clé USB

Il peut être utile d’apporter une clé USB en TD d’informatique, pour stocker vos documents et pouvoir les consulter chez vous par la suite.
Vous pouvez même y installer votre propre version de WinPython.

1.2 Bonnes pratiques

Dès le début d’un TD, créez un dossier dans votre répertoire personnel qui contiendra l’énoncé et éventuellement d’autres documents. “TD1” paraît être un nom convenable pour le TD de ce jour.

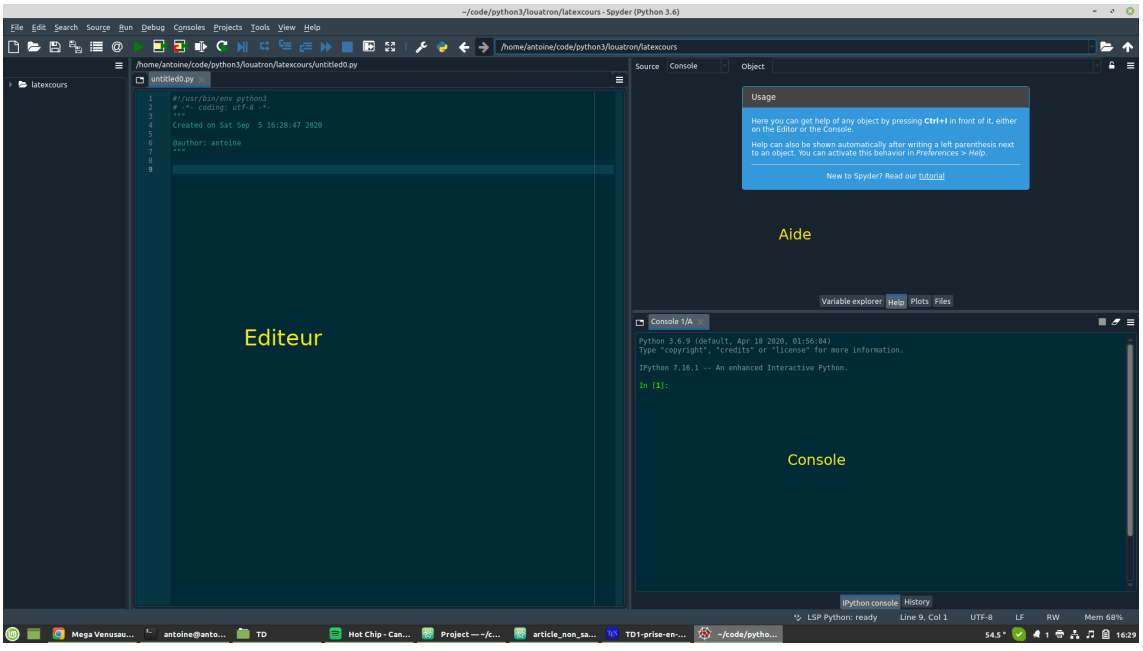
Vous aurez régulièrement des *scripts* python à créer (des fichiers texte ayant l’extension .py). Pensez à sauvegarder très régulièrement, éventuellement en utilisant le classique **CTRL-S** (ou le menu fichier).

1.3 L’IDE

Python, comme tout langage informatique, s’écrit dans un fichier texte. Il faut donc utiliser un éditeur de texte pour créer ou modifier les scripts.
Spyder sera notre éditeur par défaut. Mais ce programme est bien plus qu’un simple éditeur de texte. Vous pouvez voir au moins trois cadres

1. l’éditeur en lui-même, qui nous permettra d’écrire du texte dans un fichier
2. la console qui est le programme python.exe en lui même. Il s’agit d’un “interpréteur” (ou console) : vous tapez une ou plusieurs commandes (en langage Python évidemment) et l’interpréteur calcule le résultat de ces commandes.
3. l’inspecteur d’objet qui nous permettra souvent d’avoir accès à l’aide.

La gestion de ces cadres s’effectue via le menu “Affichage”.



Coin Culture Un éditeur de texte qui possède en plus des fonctionnalités pour programmer (comme une console) est appelé un IDE pour Integrated Development Environment.

1.4 Aide de python

Il y a plusieurs sources d'aide pour programmer en Python. Dans l'ordre :

1. la documentation de python (menu Aide ou https://docs.python.org/3/)
2. l'aide intégrée à la console :

```
help(abs)
```

   Durant toute l'année, le contenu des cadres pourra être exécuté directement dans la console, pour observer l'effet. Méfiez-vous des copier-coller en provenance d'un PDF, ils réservent souvent de mauvaises surprises.
3. le professeur pendant les TD
4. Google...

II Python

2.1 En tant que calculatrice

Il est très facile d'utiliser la console (aussi appelé l'interpréteur) pour effectuer des opérations arithmétiques simples : +, ×, -, /

Exercice Deviner le rôle des opérateurs **, //, % (entre deux entiers à chaque fois).

Nombre flottants La virgule des nombres est notée avec un point en Python. Cependant, les calculs sur les nombres non entiers peuvent donner des résultats faux. Testez les commandes suivantes que l'on expliquera en cours ultérieurement.

```
1 - 0.2 - 0.2 - 0.2 - 0.2 - 0.2
1 - (0.2 + 0.2 + 0.2 + 0.2 + 0.2)
```

Fonctions Évidemment, les fonctions mathématiques classiques peuvent être aussi utilisées. Cependant, il est nécessaire d'exécuter la commande suivante (dans la console)

```
import math
```

Calculez la racine carrée de 729, sachant que la fonction adéquate est `math.sqrt` (pour square root i.e. carrée racine).
D'une manière plus générale, les fonctions de la *bibliothèque* `math` peuvent être utilisées en préfixant (écrivant avant) leurs noms par `math.`

```
math.sin(math.pi)
```

2.2 Exécuter un script

Ouvrez dans l'éditeur de Spyder le script *fonctions.py* (via le menu Fichier ou simplement en faisant glisser le fichier correspondant depuis l'explorateur de dossier).
La différence entre une feuille de script et la console est que l'on écrit seulement les lignes que l'on souhaite sauvegarder dans le script (ie dans l'éditeur), et qu'on utilise la console pour tester notre code ou faire des calculs.

Exercice Il s'agit maintenant de trouver comment exécuter ce script (en entier, et pas ligne par ligne dans la console). Trouvez comment faire dans Spyder, et notez bien le raccourci/bouton à utiliser.
De quelle fonction avons-nous une partie de la courbe représentative ? Tracer $x \mapsto \sin(x), x \mapsto \cos(\sin(x))$. Pour cela créez un **nouveau** fichier de script dans lequel vous copierez le code qui fonctionne.
Ne JAMAIS effacer un morceau de code qui fonctionne...

Exercice Exécutez le script *vonkoch.py* sans chercher à comprendre son effet pour l'instant.
...
Cette fois aucun effet n'est visible. Ce script a pour unique but de créer une nouvelle fonction **creAnimation** dans l'interpréteur. Pour utiliser cette nouvelle fonction, il suffit de lui passer un argument (ou un paramètre) entier. Essayez avec 5. On remarque une fois l'animation finie (cliquez pour fermer la fenêtre) que **creAnimation** n'a rendu aucun résultat. Tous ses effets étaient graphiques.

2.3 Premier pas en programmation

Variables Effectuer un unique calcul est rarement suffisant pour atteindre nos objectifs. il faut donc pouvoir stocker (dans la mémoire vive, pour schématiser) les données (ou résultats) intermédiaires.
Pour donner la valeur *val* à la variable nommée a :

```
a = val
```

Cette commande crée la variable a si elle n'existait pas, et change sa valeur sinon. Il faut lire "a reçoit val" et non "a égal val".
Une variable peut avoir un nom d'une longueur quelconque. Voici quelques règles pour les noms de variables :

- ne jamais mettre d'espace
- ne pas commencer par un chiffre
- éviter les accents et autres caractères spéciaux

Exercice Ouvrez une nouvelle console (pour "oublier" le travail précédent). Devinez la valeur de la variable a après exécution de ces instructions.

```
a = 10
a = a + a
```

```
a = 5
a = a + b
```

```
a = 3
b = 2
a = b
b = a
```

Exercice Créer une suite d'instruction (dans l'éditeur ou dans la console) qui donne une valeurs à a et b puis échange les valeurs contenues dans a et b (ce script doit échanger les valeurs des variables quelles que soient les valeurs que vous avez choisi au départ).

Exercice On pose $u_0 = 1$ et $\forall n \in \mathbb{N}\ u_{n+1} = \frac{u_n}{2} + \frac{1}{u_n}$. Calculer u_1, u_2, u_3 à la main (comprendre, sans l'aide de l'ordinateur, on attend une fraction comme résultat) puis u_{10} à l'aide de python.
Astuce : pour répéter l'instruction précédente, dans la console, appuyer sur la flèche du haut.

Exercice Dans une nouvelle feuille de script (trouver dans l'éditeur comment faire ceci, et la sauvegarder au bon endroit), recopier le code suivant

```
def f(x):
    return x**2
```

puis exécuter cette feuille en entier. En cas d'erreur (ou de copier-coller), vérifier que la deuxième ligne commence bien par 4 espaces (ou au moins 1, mais le code est illisible dans ce cas).
Que devrait afficher la console après l'exécution de f(12) ? Tester ! Nous avons, un peu à la manière de la section 2.2, défini une nouvelle fonction dans la console python.

Exercice Ouvrez à l'aide de l'éditeur le fichier *degre2.py*. Le but est de compléter ce fichier de telle manière qu'une fois remplie les valeurs pour a, b, c, l'exécution de ce script affiche à l'écran les deux solutions de l'équation $ax^2 + bx + c = 0$ d'inconnue x.
Vous devez pour cela calculer les valeurs de ces solutions (et les stocker !), puis les afficher grâce aux deux `print` laissés vide.
Testez votre script avec différentes valeurs de a, b, c.
Pour éviter d'obtenir une erreur dans le cas $a = b = c = 1$ (par exemple), on doit utiliser un test :

```
if condition:
    instruction1
    instruction2
    ...
else:
    instructions
    ...
```

Les instructions du "if" seront exécutées si la condition est vraie. Le "else" est optionnel et donne des instructions à exécuter si la condition est fausse.

Il faut bien remarquer les deux points après la condition et l'indentation (le nombre d'espace qui commence la ligne). La règle en python est : pour les blocs d'instruction qui suivent un ":", il faut une indentation strictement supérieure à celle de l'instruction englobante. Généralement, on utilisera 4 espaces ou une tabulation pour chaque niveau d'indentation (Spyder est réglé pour insérer automatiquement 4 espaces).

2.4 Plus dur

Exercice Dans une nouvelle feuille Python créez :

- une fonction qui détermine si un entier est pair ou non.
- une fonction **maxi** qui prend deux arguments numérique et retourne (utiliser *return* au lieu de *print*) le plus grand des deux.
- une fonction qui étant donnés trois nombres a, b, c affiche les solutions de l'équation $ax^2 + bx + c = 0$.
- Sachant que le nombre complexe $a + ib$ avec $a, b \in \mathbb{R}$ se note $a + bj$ en python, compléter la fonction précédente pour le cas où le discriminant est négatif, et en utilisant les nombres complexes de python.
- Créer une fonction **corde** qui accepte deux paramètres a, x et qui dessine la courbe représentative de sin ainsi que la corde reliant les points de la courbe de sin d'abscisses a et x.