→ Lancer le programme “C :\Programme\winpython\Spyder.exe” et revenez lire la suite le temps de patienter...

I Environnement de travail

1.1 Les dossiers réseau

Il y a deux dossiers importants qui vous serviront cette année, ils se situent dans le lecteur “classe sur serveur”, i.e. ne sont pas stockés sur la machine que vous êtes en train d'utiliser

- le dossier “travail” qui contient des documents déposés par les professeurs. Vous pouvez lire les fichiers de ce dossier, mais pas y écrire (ni créer de nouveaux fichiers, ni modifier les fichiers existant).
- le dossier qui sera souvent appelé “dossier personnel” dans les énoncés de TD et qui est nommé suivant votre login. Ce dossier est accessible seulement par vous même (dans le sens login/mot de passe) et les professeurs.

Je vous invite à créer dès maintenant un sous-dossier “python” ou “informatique”.

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 (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. le professeur pendant les TD
3. 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).

Nombre flottants La virgule des nombres est notée avec un point en Python. Comme évoqué en cours, 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. Winpython s'occupe pour nous de “charger” des bibliothèques utiles dans la console (concept qui sera ré-expliqué plus tard). Calculez la racine carrée de 729, sachant que la fonction adéquate est sqrt (pour square root i.e. carrée racine).

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).

Exercice Il s'agit ensuite 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 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 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 à la main puis u_{10} à l'aide de python.

Exercice Créez un nouveau script nommé **mystere.py** dans votre dossier personnel à l'aide de Spyder pour y définir deux fonctions (à recopier exactement) :

```
def f1(x):
    if x > 0:
        print(' mouais ')
    else:
        print(' non ')

def f2(x):
    if x > 0:
        print(' mouais ')
    print(' non ')
```

Prédire le résultat des appels à $f1(10), f1(-3), f2(-2), f2(100)$ puis vérifier.
Quelle est la différence entre ces deux fonctions ? Laquelle comporte sûrement une erreur ?
Ces noms de fonction sont particulièrement mal choisis. Trouvez-en un meilleur pour la fonction correcte.

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.