TP n°1 : Introduction à Python

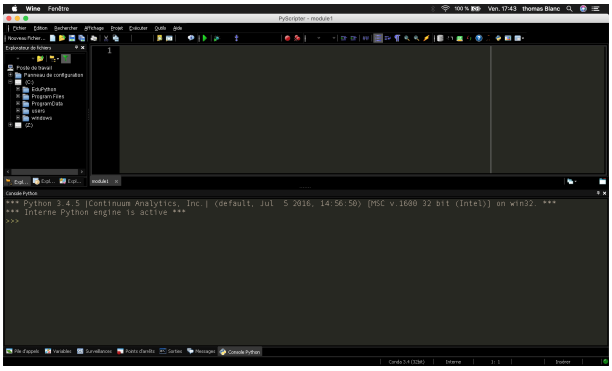
1 Avant de débuter

- Une fois votre session ouverte, commencer par créer un dossier **Python** **dans votre espace de travail**.
- Créer dans le dossier **Python** un sous-dossier **TP1**.
- Chaque fichier programme (appelé *script*) relatif au **TP1** sera enregistré dans ce sous-dossier avec le suffixe .py.
- Vous ferez de même avec les TP suivants en créant successivement les sous-dossiers TP2, TP3,... dans le dossier **Python**.

2 Ouverture de EduPython et fenêtre graphique

Dans le dossier logiciel_maths, ouvrir EduPython.
A l'ouverture, une fenêtre composée de 3 éléments s'affiche.

- Un explorateur de fichier
- La *fenêtre de script*. C'est le fichier .py dans lequel on écrit les instructions. Ce fichier peut être enregistré pour une utilisation ultérieure.
  - Pour exécuter un programme on clique sur le bouton vert ▷
- La *console* où on peut trouver une invite de commande >>> qui indique une attente d'instruction.
  - C'est là que va s'exécuter le programme et vont se signaler les erreurs éventuelles.
  - On peut également directement mettre des instructions simples et s'en servir comme d'une calculatrice. Ces instructions sont exécutées immédiatement, et leur résultat est affiché dans la console.



3 Première commande dans la console et importation d'une bibliothèque

3.1 Utilisation de la console comme d'une calculatrice

Exercice n°1 Cet exercice est à réaliser dans la **console**.

1. Réaliser les calculs suivants : $2 \times 3 - 9$, $2(4 - 13 + 12)$, $\dfrac{(10+15)(9-3)}{30}$.
2. La puissance se note `**` en Python et pas `^`.
   (a) Calculer 2^3, 3^{2000}. **Rem. :** La taille d'un nombre **entier** n'est limité en Python que par la mémoire machine.
   (b) Combien vaut $3^{1000} - 3 \times 3^{999}$? Vérifier à l'aide de Python. **Rem. :** Les calculs avec les nombres entiers sont exacts.
3. La virgule est remplacée par un point en Python
   Combien vaut $0,2+0,1$? Et $0,2 \times 0,1$? Vérifier à l'aide de Python. **Rem. :** Les nombres à virgule, appelés **flottants**, possèdent un nombre limité de chiffres après la virgule et sont toujours des approximations.

3.2 Importer une bibliothèque

- Les constantes usuelles : par exemple `e`, `pi`
- Les fonctions usuelles : par exemple `sqrt` (racine carrée), `exp` (exponentielle), `log` (logarithme népérien),

ne sont pas "directement" définies à l'ouverture de python. Par exemple taper `pi` ou `sqrt(2)` dans la console et lire le message d'erreur. Il est nécessaire d'importer une *bibliothèque* (ensemble de fonctions) les contenant. La bibliothèque contenant les fonctions mathématiques de base s'appelle `math`. Pour importer une bibliothèque, on peut procéder de plusieurs façons :

1) Importer toute la bibliothèque math :

```
> > > import math
> > > math.sqrt(100)
```

On peut alors utiliser `sqrt` en ajoutant le préfixe `math.`

2) Importer toute la bibliothèque math en utilisant un alias plus simple :

```
> > > import math as m
> > > m.sqrt(100)
```

On peut alors utiliser `sqrt` en ajoutant le préfixe `math.`

3) Importer toute la bibliothèque math en enlevant le préfixe

```
> > > from math import*
> > > sqrt(100)
```

On peut alors utiliser `sqrt` sans ajouter de préfixe à condition qu'il n'y ait pas de conflit de bibliothèque.

Exercice n°2 Dans la console, en utilisant la seconde méthode d'importation de la bibliothèque `math`, faire afficher une valeur approchée de chacune des quantités : $\frac{\sqrt{5}+1}{2}$, $\exp(1)$, $\ln(2)$, $\ln(e^2)$, $\ln(0)$, e, $\lfloor 12,3 \rfloor$, $\lfloor -12,3 \rfloor$.

4 Variables et affectations

Les opérations de ce paragraphe sont à réaliser dans la console.

- **Création d'une variable** Une variable est un emplacement mémoire permettant de conserver des données. Ces données pourront alors être de nouveau manipulées par la suite.

  Par exemple, si on souhaite stocker la valeur du calcul $2 \times 3 + 5$ dans une variable, il faut :

  – donner un nom à cette variable (mettons `x`), on dit qu'on **déclare la variable** `x` ;

  – préciser qu'elle contient la donnée $2 \times 3 + 5$, on dit qu'on **affecte** à `x` la valeur $2 \times 3 + 5$.

  Ces deux opérations se font en tapant `>>> x=2*3+5`. Si vous souhaitez par la suite connaître la valeur contenue dans la variable `x`, il suffit d'écrire `>>> x`. Python affiche alors 11.

- **Nom d'une variable** Un nom de variable est un mot composé de lettres et de chiffres mais qui commence par une lettre (noter que python est sensible à la "casse" , il fait la différence entre les minuscules et les majuscules). En mathématiques, une variable est presque toujours désignée par une seule lettre. En informatique, on préfèrera utiliser des **noms de variables plus explicites** : par exemple, si l'on calcule une moyenne, au lieu de l'appeler `x` ou `A`, on l'appellera `moyenne`. À la relecture, le programme informatique n'en sera que plus clair.

- **Modification d'une variable** Une variable peut ensuite être utilisée dans tout calcul à l'aide de son nom et **elle peut être modifiée lors d'une nouvelle affectation** : dans ce cas, la valeur initiale de la variable est écrasée par la nouvelle valeur et est perdue. Tester les commandes suivantes :

```
> > > a=1              (On déclare la variable a et on lui affecte la valeur 1)
> > > b=8              (On déclare la variable b et on lui affecte la valeur 8)
> > > a = b            (On écrase la variable a pour lui donner la valeur contenue dans b)
> > > a                (On affiche le contenu de a)
> > > b                (On affiche le contenu de b)
> > > maVariable=2
> > > maVariable
> > > mavariable       (Renvoie une erreur car la casse est différente)
```

Exercice n°3

1. Essayer de prévoir l'affichage de python de ces 2 groupes d'instructions. Taper ensuite les instructions dans la console python pour vérifier.

   Attention : `a=b` ne signifie pas que `a` est égal à `b` mais qu'on affecte à la variable `a` la valeur de la variable `b`, l'ancienne valeurs contenue dans `a` est alors écrasée.

   (1)

```
> > > x=1
> > > y=4
> > > x=x+y
> > > y=x-y
> > > x=x-y
> > > x
> > > y
```

   (2)

```
> > > x=2
> > > y=x*3
> > > x=x+1
> > > z=x+y
> > > x=y+z
> > > x
```

2. On souhaite construire une suite d'instructions qui échange les valeurs de deux variables `x` (mettons 6) et `y` (mettons 2). Voici la suite d'instructions proposées :

```
> > > x=6
> > > y=2
> > > x=y
> > > y=x
> > > x
> > > y
```

   Les instructions précédentes permettent-elles d'échanger les variables `x` et `y` ? Essayer de répondre à cette question sans exécuter ces lignes dans la console. Dans le cas contraire, proposer une suite d'instructions permettant de faire cet échange.

Exercice n°4 Variables booléennes et chaînes de caractères

1. **Variables booléennes et tests :** Les variables booléennes prennent uniquement deux valeurs : *True* ou *False*. Elles servent à tester si une relation est vraie ou fausse. Par exemple :

```
> > > a1 = 2
> > > a2 = 3
> > > a1 < a2
> > > a1 > a2
> > > x = 3
> > > y = 6
> > > z = 2*x
> > > y == z
> > > y != z
```

2. **Les chaînes de caractères :** On peut définir en Python des variables contenant des mots ou même des phrases. Pour cela on utilise les apostrophe, par exemple `>>> a='coucou'`. On utilise des doubles apostrophes si une apostrophe est présente dans la chaîne, par exemple `>>> b="c'est ma première séance sur Python"`.

   On peut également réaliser des opérations sur les chaînes de caractères. Taper les instructions suivantes et comprendre les résultats :

```
> > > x=3
> > > y=4
> > > x+y
> > > z='EC'
> > > t='G1'
> > > z+t
> > > x+z
> > > print('y prend la valeur : ',y)
```

5 Programmation et fichier script .py

Dès que le programme est un peu long ou que l'on désire exécuter plusieurs fois le même programme, la console de python n'est plus pratique. Il devient préférable de taper toutes les instructions dans un fichier à part, que l'on nommera **script**, qui sera sauvegardé sur le disque dur (donc réutilisable lors d'une autre session python et récupérable sur clef usb) et qui sera exécuté sur la console.

Les programmes sont tapés directement dans la fenêtre de script. Il faut commencer par faire une sauvegarde sous l'extension `.py`. Par exemple, pour le TP1, sauvegarder dans le répertoire TP1 le fichier script sous le nom TP1.py. Par la suite penser à sauvegarder régulièrement votre travail.

Attention, les lignes du script ne s'exécutent pas lorsqu'on les écrit.

Pour exécuter le programme : pour exécuter le programme écrit dans le script, cliquer sur l'icône vert ▷.

Pour ouvrir un fichier script déjà créé lors d'une session précédente, depuis la fenêtre de EduPython, faîtes une recherche dans l'explorateur de fichiers ou cliquer sur fichier, puis ouvrir.

Pour créer un nouveau fichier script, depuis la fenêtre EduPython, cliquer sur fichier, puis nouveau module python.

Commentaire : On peut placer des commentaires dans notre programme python pour le rendre plus lisible et expliquer ce que l'on code. Les commentaires sont précédés du signe #. Dès qu'il voit ce signe, python passe directement à la ligne suivante sans s'occuper de ce qui est écrit après.

Exemple : Taper **dans la fenêtre de script** le script suivant :

```
#Ceci est un commentaire
import math as m #on commence par importer la bibliothèque math avec l'alias m
a=1
b=a-5
c=a+m.exp(5)
d=min(a,b,c)
```

Que remarquez-vous dans la console après exécution ?

- **Affichage :** Les résultats ne s'affichent qu'à condition de le demander. Pour forcer l'affichage, on utilise la commande `print`. Dans l'exemple précédent, ajouter `print(a,b,c,d)`
- On peut aussi s'en servir pour afficher toutes sortes de variables, et donc présenter ses résultats. Par exemple, rajouter en fin de script :

  ```
  print('La valeur de a est ',a,' mais celle de d est ',d)
  ```

6 Fonctions

A partir de maintenant, nous aurons besoin d'exécuter plusieurs lignes à la fois. Nous les écrirons donc d'abord dans la partie script avant de les exécuter.

- Une fonction est une suite finie d'instructions qui reçoit un argument et renvoie un résultat. La syntaxe est la suivante :

```
def maFonction(argument):
    suite d'instructions
    suite d'instructions
    ...
    return résultat
```

- La fonction est composée
  - d'une ligne d'entête commencée par le mot clef `def` et terminé par deux-points,
  - d'un bloc d'instructions indenté (il y a une tabulation, un espace, en début de ligne) par rapport à la ligne d'entête,
  - d'une instruction `return` qui renvoie la résultat (aussi indenté).
- **Remarques :** 1) L'indentation est automatiquement réalisée par EduPython après avoir mis les deux-points. Si les deux-points sont oubliés, EduPython souligne en rouge la ligne où il y a eu l'oubli.
  2) Après `return`, la fonction ne lit plus rien dans le blocs. Toutes les instructions indentées qui suivent seront ignorées.

Exemple :

Taper dans **la fenêtre de script** le code suivant, puis l'exécuter

```
def ajouter1(x):
    x  += 1   # ajoute 1 à la variable x
    return x
```

Puis taper dans **la console**

```
> > > ajouter1(4)
> > > ajouter1(-7)
> > > ajouter1(3.75)
> > > b=9
> > > ajouter1(b)
> > > print(b)
```

Exercice n°5

1. Programmer la fonction f définie, pour tout $x \in]1;+\infty[$, par : $f(x) = \sqrt{\dfrac{\ln(x)}{x^2+1}}$.
2. Programmer une fonction qui prend en argument deux nombres réels a, b et qui affiche la somme de leur carrés.
   Tester ce programme dans la console.
3. On suppose qua ABC désigne un triangle rectangle en B.
   Programmer une fonction `hypotenuse` qui prend en argument les valeurs des longueurs AB et de BC et qui renvoie la longueur de AC.

Exercice n°6

Soit l'équation du second degré d'inconnue $x \in \mathbb{R}$, $ax^2+bx+c=0$. Ecrire une fonction qui prend en argument les valeurs des paramètres a, b et c et qui renvoie le couple des deux racines de l'équation (on supposera le discriminant positif). Vérifier le programme sur des cas simples.
Faire un programme qui traite le cas général, i.e. sans présupposer la positivité du discriminant.