1 Introduction

Le langage que nous utiliseront pour programmer est le le **Python**, (version 3).
Nous devrons donc nous familiariser avec les bases de ce langage, afin de programmer des algorithmes liés au programme de mathématiques.

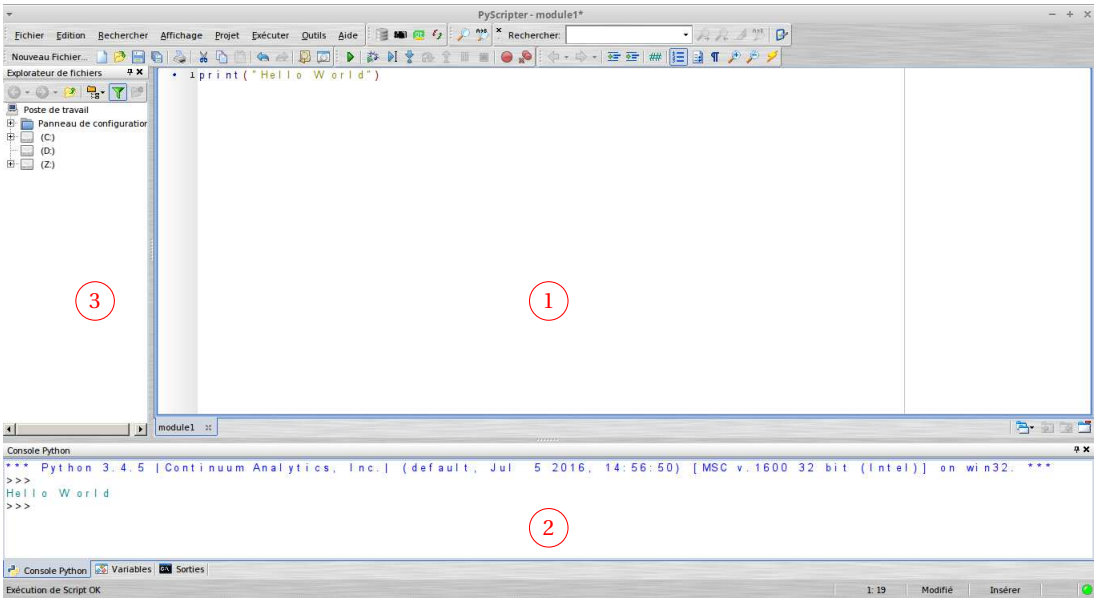
2 Hello World

Comme dans tout cours de programmation, le premier programme sera l'affichage du texte "Hello World".

Nous utiliserons, pour une première approche, la console "EDUPYTHON", mais les principales fonctionnalités décrites dans ce cours restent valables sur d'autres consoles Python.

Ouvrir le logiciel Edupython, vous devez avoir une console comme ci-dessous :

Pour afficher "Hello World", l'instruction à saisir dans la zone de saisie du programme est print("Hello World"), ensuite, exécuter le programme en cliquant sur la flèche verte ▶.



① : Zone de saisie du programme ② : Zone d'exécution du programme ③ : Explorateur de fichiers

Commande 1.

print

print(5) affiche la valeur numérique (ici 5).

print(12,7) affiche à la suite, sans passer à la ligne les éléments 12 et 7, avec un espace entre les deux.

Pour afficher du texte, ...

Exercice 1.

Faire afficher :
Ma date de naissance est le 5 décembre 2000.

3 Effectuer des calculs

3+2	donne 5 (de même pour la soustraction)
4*3	donne 12 (* correspond à la multiplication)
2/3	donne 0.6666666666666666 (valeur approchée de la division de 2 par 3)
2 ** 3	donne 8 (= 2^3)
14 // 3	donne 4 (quotient de la division euclidienne de 14 par 3)
14%3	donne 2 (reste de la division euclidienne de 14 par 3)

C'est une valeur approchée et non arrondie.

D'autres calculs nécessitent l'import du module "math", on l'indique au début du programme par l'instruction **from math import***.

On peut alors obtenir une valeur approchée de la **racine carrée** de 2, en tapant simplement sqrt(2).

Nous verront, quand cela sera nécessaire, d'autres instructions de calculs provenant de ce module.

Pour afficher le calcul qui résulte de l'addition de 2 et de 3, entrer l'instruction *print(2+3)*.

Exercice 2.

Demander l'affichage des calculs suivants :

1. $2+3\times 5$.
2. $\sqrt{2}$.
3. $3\sqrt{2}$.
4. $\sqrt{2+3\sqrt{5+4\sqrt{3}}}$.

Solution.

On remarque que l'écriture naturelle .....n'est pas reconnue ici, et que le symbole de .....est absolument nécessaire.

1.

2.

3.

4.

Notes :

Exercice 3.

On considère, dans un repère **orthonormé** $(O, \vec{i}, \vec{j})$ les points $A(5;3)$ et $B(-1;15)$.
Faire afficher :
Les coordonnées du milieu du segment $[AB]$ sont (...,...).
La distance AB est environ égale à ....

Solution.

Attention à la précision des calculs!

Nous devront souvent prendre garde à la précision des calculs...

Exercice 4.

1. Calculer $\left(\sqrt{2}\right)^2$. Quel est le résultat affiché?
2. Calculer $\frac{1}{10}+\frac{1}{10}+\frac{1}{10}$. Quel est le résultat affiché?
3. Calculer $3\times\left(\frac{1}{10}\right)$ et $3\times\frac{1}{10}$. Que constate-t-on?

4 Première utilisation d'une variable

Définition 1.

Les **variables** permettent de stocker des données. En langage Python, leur type (nature du contenu) est défini par leur première affectation.

Exemple 1.

Notez que le symbole "=" n'est pas utilisé ici comme en mathématiques, il s'agit ici d'une affectation et non d'une égalité. Ce symbole, en programmation n'est pas symétrique, il faudra bien écrire $a=2$, car $2=a$ ne sera pas reconnu.

l'instruction $a = 2$ permet à la fois de créer une variable dont le nom est a, et de lui affecter la valeur 2. Le type de cette variable sera donc "nombre entier".

l'instruction $a = 2.1$ affectera à la variable a le type "nombre à virgule flottante", couramment appelé nombre décimal.

On peut définir plusieurs variables et leur affecter une valeur en une seule instruction, ainsi, l'instruction $a, b = 2,3$ définira les variables a et b et affectera à a la valeur 2 et à b la valeur 3.

Exercice 5.

Qu'affichera le programme suivant ?
$a, b = 1, 5$
$x, y = a + b, a - b$
print(x,y)

Exercice 6.

Qu'affichera le programme suivant ?
$x, y = 1, 5$
$x = x + y$
$y = x - y$
print(x,y)
Obtient-on le même résultat que dans l'exercice précédent ?
Comment pourrait-on modifier ce programme pour qu'il affiche les mêmes résultats que dans l'exercice précédent ?

Solution.

Exercice 7.

Écrire un programme qui affecte deux valeurs aux variables x et y, puis qui échange les valeurs de ces deux variables.

Une troisième variable de « transition » est utilisable, même si Python le peut le faire directement (en utilisant néanmoins le même type de procédé).

Solution.

Exercice 8.

Évolution de l'exercice 3.
On souhaite écrire un programme permettant d'affecter à des variables les coordonnées de deux points A et B, et de faire afficher les cordonnées de leur milieu I, puis la distance AB.
Proposer une solution.

Solution.

Commande 2.

La commande **input**.
Celle-ci permet, lors de l'exécution du programme, de faire intervenir l'utilisateur, afin que celui entre des caractères à l'aide du clavier, afin de les stocker dans une variable.

Exercice 9.

Tapez le programme suivant : prenom = input("Entrez votre prénom : ")
print("Bonjour,", prenom)
Qu'est-il affiché ?

Exercice 10.

1. Tapez le programme suivant :
   x=input("x")
   print(x)
   Qu'est-il affiché ?
2. Tapez le programme suivant :
   x=input("x")
   y=x+1
   print(y)
   Qu'est-il affiché ?

x est ici considéré comme
.....

Commande 3.

Si l'on veut que notre entrée (numérique) soit effectivement de type nombre, on utilise la commande **x=eval(input("x"))** .

Exercice 11.

Réécrire le programme de l'exercice 8 afin d'entrer les coordonnées de A et de B lors de l'exécution du programme.

Solution.

Exercice 12.

On considère le programme de calcul suivant :

- Choisir un nombre
- Lui retirer 10
- Multiplier le résultat par le nombre de départ
- Ajouter 25

1. Transcrire ce programme en langage Python.
2. Quelle conjecture peut-on faire sur le résultat obtenu? (On pourra faire faire quelques tests avec des entiers, puis étudier le cas général.)

Solution.

Penser aux identités remarquables...

Exercice 13.

Créer un programme qui demande à l'utilisateur ses trois dernières notes de SVT ainsi que les trois coefficients associés et qui affiche sa moyenne.

Solution.

Exercice 14.

Écrire un programme qui demande à l'utilisateur d'entrer les coordonnées de deux points A et B et qui affiche les coordonnées du symétrique de A par rapport à B.

Solution.

Exercice 15.

Écrire un programme qui demande à l'utilisateur d'entrer les coordonnées de trois points A , B et C et qui affiche les coordonnées du point D tel que $ABCD$ soit un parallélogramme.

Solution.

5 Les fonctions

Vous pouvez choisir n'importe quel nom pour la fonction que vous créez, à l'exception des mots réservés du langage, et à la condition de n'utiliser aucun caractère spécial ou accentué (le caractère souligné "_" est permis). Comme c'est le cas pour les noms de variables, il vous est conseillé d'utiliser surtout des lettres minuscules, notamment au début du nom.

Commande 4.

Une fonction Python est définie par le spécificateur "def" suivi du nom de la fonction et de ses paramètres :

```
def nomDeLaFonction(liste de paramètres):
    ...
    bloc d'instructions
    ...
    return résultat
```

Attention aux deux points en bout de ligne et à l'indentation qui suit.

Si l'on utilise des variables dans une fonction, celles-ci ne seront que locales, et ne seront pas reconnues à l'extérieur de la fonction.

La notion de fonction étant très riche, nous verrons au fur et à mesure ses nombreuses possibilités.

Exemple 2.

1. la fonction

   ```
   def addition(x, y):
           return x + y
   ```

   retournera la somme de x et y, ainsi, si l'on tape dans la console : addition(5,7), cela nous retournera 12.
2. La fonction

   ```
   def moyenne2nombres(a,b):
           return (a+b)/2
   ```

   retournera la moyenne de a et de b , ainsi, si l'on tape dans la console : moyenne2nombres(4,12), cela nous retournera 8.

Exercice 16.

Créer deux fonctions, milieu et distance, dont les paramètres sont les coordonnées de deux points A et B et qui retournent respectivement les coordonnées du milieu du segment $[AB]$ et la distance AB.

Solution.

Dorénavant, quand nous écrirons un script python qui est susceptible d'être réutiliser, nous créerons une fonction que nous placerons dans un fichier "fonctionsperso.py".

6 Tests

Commande 5.

On a souvent besoin de tester si une condition est vraie ou fausse, et selon le cas, d'agir de différentes façons. La commande **if** :

```
if condition vraie:
    action
```

ou bien

```
if condition vraie:
    action
else:
    action
```

En Python, le Alors se traduit par deux points (**:**) et un décalage des instructions. SINON se traduit par **else :**

Exemple 3.

```
x=eval(input("x"))
if x>=0:
    print("nombre positif ou nul")
else:
    print("nombre strictement négatif")
```

Ce programme demande à l'utilisateur d'entrer un nombre, et teste son signe, puis affiche un des deux messages.
Tester ce programme.

Exercice 17.

Écrire un programme qui demande à l'utilisateur d'entre un entier, et indique la parité de celui-ci .

Solution.

Exercice 18.

Écrire un programme qui demande à l'utilisateur d'entrer un entier, et indique la parité de celui-ci, en émettant un message d'alerte si le nombre entré n'est pas de type entier.

Solution.

La commande pour connaître le type d'une variable est .....

Exercice 19.

Écrire un algorithme qui permet de résoudre l'équation $ax+b=0$ avec $a \in \mathbb{R}$ et $b \in \mathbb{R}$.

Solution.

Exercice 20.

Écrire un programme qui demande à l'utilisateur d'entrer deux réels a et b, et qui résout l'équation $ax+b=0$.

Solution.

Exercice 21.

Écrire un programme qui demande à l'utilisateur d'entrer les coordonnées de quatre points A , B , C et D et qui affiche évalue si $ABCD$ est un parallélogramme.

Solution.

Exercice 22.

Écrire un programme qui demande à l'utilisateur d'entrer les coordonnées de deux vecteurs et qui teste si ils sont colinéaires.

Solution.

Exercice 23.

Écrire un programme qui demande à l'utilisateur d'entrer les coordonnées de trois points et qui teste si ils sont alignés.

Solution.

Exercice 24.

Écrire un programme qui demande à l'utilisateur d'entrer les coordonnées de quatre points A, B, C et D et qui teste si les droites (AB) et (CD) sont parallèles.

Solution.

7 Les boucles

Définition 2.

Les boucles, en programmation, servent à répéter un certain nombre d'instructions, soit un nombre prédéfini de fois, soit tant qu'une condition est réalisée.

7.1 La boucle « for »

Commande 6.

Bien qu'il soit possible d'utiliser ce type de boucle de très nombreuses façons, nous n'en verrons que deux dans un premier temps.

```
for i in range(0,4):
```

Ici, **i** prend les valeurs entières de l'intervalle $[0;4[$, c'est à dire 0, 1, 2 et 3.
Attention : vous remarquerez que la dernière valeur entière n'est pas prise par **i**.

```
for i in range(1,15,2):
```

Ici, **i** prend les valeurs entières de l'intervalle $[1;15[$ avec un pas de **2**.
i prend donc les valeurs...
Comme il y a deux points (" :") en fin de ligne, il y a une indentation vers la droite du bloc d'instructions qui suit.

Exemple 4.

1. Le programme suivant suivant demande à l'utilisateur un entier, et affiche les carrés parfaits inférieurs ou égaux à celui-ci (avec une alerte si le nombre entré n'est pas entier) :

```
n=eval(input("n"))
if type(n)!=int:
    print("ce n'est pas un entier")
else:
    for i in range(n+1):
        if i**2<=n:
            print(i**2)
```

2. Le programme suivant demande à l'utilisateur un entier, et affiche les carrés des entiers pairs inférieurs ou égaux à celui-ci (avec une alerte si le nombre entré n'est pas entier) :

```
n=eval(input("entrer un entier n"))
if type(n)!=int:
    print("ce n'est pas un entier")
else:
    for i in range(0,n+1,2):
        print(i**2)
```

Exercice 25.

1. Écrire un programme qui fait afficher vingt fois « J'aime la programmation en Python ».
2. Que se passe-t-il si l'on supprime l'indentation avant le « print » ?
3. Écrire un programme qui demande à l'utilisateur un entier naturel n, et qui affiche tous les entiers pairs inférieurs ou égaux à n. Proposer deux solutions en utilisant les deux commandes « in range ».
4. Écrire un programme qui demande à l'utilisateur un entier naturel n, et qui affiche tous les diviseurs de n.

Exercice 26.

Écrire le programme de dichotomie vue en classe, et le faire évoluer pour changer le nombre de répétitions de la boucle.
A-t-on la main sur la précision obtenue ?

Le coin des curieux.

Chercher la définition d'un nombre parfait.

Adapter le **3.** de l'exercice précédent afin que le programme affiche si un entier naturel est parfait ou pas.

Et pour les plus curieux, adapter ce programme pour qu'il donne les nombres parfaits inférieurs ou égaux à un entier naturel n entré par l'utilisateur lors de l'exécution du programme.

Combien de nombres parfaits inférieurs ou égaux à 1000 existe-t-il?

7.2 La boucle « while »

Commande 7.

Cette boucle se présente de la façon suivante :

```
while condition:
    bloc d'instructions
```

Tant que la condition est vraie, le bloc d'instructions se répète.

Attention : il faut s'assurer que ce type de boucle ne tourne pas indéfiniment...

Exemple 5.

Que fait le programme suivant?

```
deb=eval(input("deb"))
fin=eval(input("fin"))
pas=eval(input("pas"))
x=deb
while x<=fin:
    print(x**2)
    x=x+pas
```

Exemple 6.

Analyser ce programme :

```
x=eval(input("x"))
while x**2>=0:
    print(x)
    x=x+1
```

Exercice 27.

Reprendre le programme de dichotomie précédent afin de demander à l'utilisateur la précision voulue pour l'encadrement de la solution cherchée.
Code du programme :