Python 1 : Généralités et rappels du Lycée

Un programme informatique est une succession d'instructions (des lignes de code) qui s'exécutent à la suite une fois qu'on décide de lancer (c'est à dire d'exécuter le programme, en anglais *run*).
A chaque fois qu'on exécute le programme, le programme va effectuer les instructions de la même manière depuis la ligne 1 jusqu'à la dernière ligne, dans l'ordre.

Voici quelques liens vers des consoles en ligne où essayer vos programmes :
https://www.online-python.com/online_python_compiler
https://console.basthon.fr/

Vous pouvez également télécharger le logiciel Spyder pour pouvoir compiler hors ligne :
https://www.spyder-ide.org/

1) Variables et affectation

Une variable est un emplacement mémoire, désigné par une chaîne de caractères (une suite de lettre et de chiffres)

Il existe différents types de variables :

- int : les entiers
- float : les nombres réels
- str : les chaines de caractères
- list : les listes (nous verrons ce type plus tard)
- array : les tableaux ou matrices (nous verrons ce type plus tard)
- bool : les booléens True et False (nous verrons ce type plus tard)

Pour affecter une valeur à une variable, on utilise =, qui signifie que la variable à gauche du signe = va prendre la valeur de ce qu'il y a à droite du signe =

Exemple : pomme=2 affecte à la variable pomme l'entier 2 (son type est donc int)

Exemple : z=1.4 affecte à la variable z le réel 1.4 (son type est donc float)
Attention, **Python est en anglais**, donc la virgule décimale est un point, on écrit 1.5 et non pas 1,5.

Exemple : y='bonjour' affecte à la variable y la valeur bonjour (son type est donc str)

Exemple : En écrivant x=x+1, on affecte à x la valeur x+1 (donc si x valait 1 par exemple, x vaut 2 après cette ligne de code).
Si on veut aller plus vite, on peut affecter à deux variables en même temps des valeurs, en écrivant par exemple : x,y=1.5,0 (ici on affecte à x la valeur 1,5 et à y la valeur 0).

2) Les opérations sur les nombres (int et float)

a+b : addition
a*b :multiplication (attention si vous écrivez 2x ça ne fonctionne pas, il faut écrire 2*x)
a-b : soustraction
a/b : division (n'oubliez pas les parenthèses, par exemple $\frac{2}{4+1}$ s'écrit 2/(4+1)
a**b : élévation de a à la puissance b (a^b)

Par exemple, si l'on veut écrire $\frac{2^{3x}+1}{x+1}$, on peut taper (2**(3*x)+1)/(x+1)

3) L'instruction input

L'instruction a=type(input(a)) une fois le programme lancé, va faire que l'ordinateur demande à l'utilisateur de rentrer une valeur qui va ensuite être utilisée par la variable a dans la suite du programme. On remplacera "type" par le type de variable que l'on veut : int, float etc.
Pour saisir un entier, on utilise int(input('…')) , et si on souhaite saisir un réel on utilise float(input('…')).

Par exemple le programme :
a=int(input('rentrer la valeur du nombre entier a'))

Va faire que la console vous demandera de rentrer vous-même la valeur de a (qui est un entier) à chaque exécution du programme.

4) L'instruction print

L'instruction print('…') affiche le texte écrit entre apostrophe (on met ce que l'on veut).
L'instruction print(a) affiche la valeur de la variable a, si elle a été définie préalablement.
Soit n=5 et u=10. L'instruction print('u', n, 'vaut', u) affiche : u5 vaut 10

Exemple : le programme :
x=1
print(x+4)

Va afficher la valeur 5

Le programme :
x=float(input('rentrer la valeur du réel x'))
*print(x**2)*

Va demander à l'utilisateur un réel x et va renvoyer la valeur de x^2.

5) Utiliser les fonctions usuelles

Pour utiliser les fonctions usuelles (exponentielle, racine carrée etc.) on doit taper au tout début du programme la ligne de code :

from math import *
Cela va débloquer l'usage des fonctions usuelles que l'on pourra utiliser librement dans le programme.

exp(x) : e^x sqrt(x) : $\sqrt{x}$ floor(x) : $\lfloor x \rfloor$ (partie entire de x)
log(x) : ln(x) abs(x) : $|x|$ (valeur absolue de x)

Notamment si on écrire $e^{2x+1}+\sqrt{ln(6)}$ on doit écrire : exp(2*x+1)+sqrt(log(6))

On peut égaler débloquer les fonctions usuelles à l'aide de la bibliothèque numpy qui se débloque en écrivant :

import numpy as np

Le nom des fonctions est le même mais avec np. devant (c'est plus lourd mais il y a plus de commandes dans la bibliothèque numpy, nous en verrons dans de prochains chapitres).

6) Commentaires

On peut écrire des commentaires à la suite de ligne de code, pour expliquer ce que font les lignes, par exemple :

```
x=x+1      #  j'affecte à x la valeur x+1
```

Ce qui se situe après le # n'est pas exécuté par l'ordinateur, donc on peut écrire ce que l'on veut, l'intérêt d'un commentaire est de faire comprendre son code lorsque ce dernier est long et complexe.

➔ A utiliser sur vos copies lorsque le code n'est pas trivial pour expliquer au correcteur ce que vous faites.

7) Gestion des erreurs

Lorsqu'on lance un programme, souvent le programme ne se lance pas à cause d'erreurs. Il y en a de plusieurs sortes :

a) Les erreurs de syntaxe : par exemple si vous orthographiez mal le nom d'une fonction le programme ne va pas se lancer : expp(1) au lieu de exp(1) ; ou encore si vous oubliez de fermer les parenthèses, par exemple sqrt(exp(1)

b) Les oublis de déclaration d'une variable, par exemple si vous écrivez la ligne x=x+1 et que vous n'avez pas affecter de valeur à x dans les lignes au-dessus, le programme ne marchera pas (car il ne sait pas que vaut x !)
c) Les opérations "interdites", par exemple si vous écrivez x=0 et ensuite y=1/x
d) L'oubli d'importation des fonctions : si vous utilisez les fonctions exp, ln mais que vous avez oublié de mettre la ligne from math import * au début (le programme ne connait alors pas ces fonctions)
e) Il ne se passe rien ? Par exemple vous écrivez :
```
x=int(input('rentrer la valeur de x')
x=x**2
```
Et une fois que vous lancez le programme et rentrer la valeur de x, rien ne se passe. Ici c'est normal car vous n'avez pas demandé de faire plus ! Si vous voulez que l'ordinateur affiche x il faut lui dire, en ajoutant print(x) par exemple.

La console va également vous indiquer à quelle ligne se situe votre erreur, pensez-y ça délimite le champ de recherche des erreurs.

Parfois, la console bug, dans ce cas il est sage de tout fermer et de relancer (en n'oubliant pas de garder en copier-coller votre code !)