Introduction à l'algorithmique et la programmation avec Python

1 Qu'est-ce qu'un algorithme ?

Un algorithme est une suite d'instructions (aussi appelées commandes), qui, une fois exécutée, conduit à un résultat donné.

Pour fonctionner, un algorithme doit donc contenir uniquement des instructions compréhensibles par celui qui devra l'exécuter.

Le terme d'algorithme vient du nom du mathématicien arabe du IX$^{\text{ième}}$ siècle **Al Khwarizmi (788-850)** qui a mis en place des méthodes systématiques pour la résolution de problèmes.

2 Qu'est-ce qu'un langage de programmation ?

Un langage de programmation est un ensemble d'instructions et de règles syntaxiques compréhensibles par l'ordinateur et permettant de créer des algorithmes.
Il existe des milliers de langages de programmation, ayant chacun leurs spécificités. Désormais et à partir du bac 2021, Python est le programme et le langage officiel pour le lycée et la prépa EC.
Nous utiliserons cette année une interface en ligne, **Capytale**, qui nécessite donc d'être connecté à internet.
On peut aussi installer sur ordinateur un logiciel qui permet d'utiliser Python : **EduPython** en est un que l'on peut télécharger gratuitement à l'adresse suivante : http ://edupython.tuxfamily.org
Vous pouvez également installer **Pyzo** comme sur les machines du lycée.

3 Premiers obstacles : le langage

On poursuit en se connectant sur Capytale (si besoin code de l'activité : 023a-3917537).
Pour communiquer avec Python, il faut utiliser un langage ne souffrant aucune ambiguïté.
▷ Par exemple lorsqu'on tape dans la console `x=3`, on définit une variable numérique contenant la valeur 3.
Si on tape désormais `x`, Python nous renvoie le contenu de la variable `x=3`, mais si on tape alors `2x`, Python nous parle d'erreur de syntaxe.
En effet, il s'agit alors d'une multiplication, ce qu'il faut expliciter et donc taper `2*x`. De même que Python n'interprètera pas `23` comme une multiplication, il ne le fait pas pour `2x`.

Autre illustration, comme nous l'avons vu avec `ans`, Python ne parle qu'en anglais. Et il utilise également des conventions britanniques, donc on écrira 3.14 et non 3, 14.
Pour le coup, le programme ne renvoie pas forcément d'erreur si on utilise la virgule, ce qui peut nous compliquer la tâche.
▷ Par exemple, si on tape `4*3,5`, Python renvoie `12` puis `5` (le résultat de 4×3 « virgule » le nombre 5) au lieu que 14 qui est obtenu en tapant `4*3.5`.

4 L'interface d'un logiciel Python

Dans la plupart des interfaces qui exploitent Python (dont Pyzo, Edupython... mais pas Capytale), on peut entrer des informations à deux endroits :
- dans la console, plutôt pour des opérations ou des commandes « simples » (qui tiennent en une ligne) ;
- dans l'éditeur, plutôt pour l'exécution de programmes (succession de commandes).

5 Programmer avec Python, les premiers pas et outils

1. Python est avant tout une calculatrice : testez !
   $2 \times 3, 10/4, 4^2, 2 \times (3+2), 2 \times 3 + 2, 4^2 + 1, 4^{2+1} \ldots$
2. Afficher quelque chose : c'est la fonction `print`

```
print("Salut !")
```
Affichera Salut !

```
print(11+3)
```
Affichera 14

3. Tester une égalité, une inégalité... : $1+1=2$?$1=2$? $1<4$? $1-1 \leqslant 0$?
4. Donner une condition : c'est la fonction `if`

```
import numpy.random as rd # permet d'utiliser la fonction randint ci-dessou

a=rd.randint(1,6)                    # simule le lancer d'un dé

if a==6 :
    print("gagné") # affichera gagné si le chiffre est 6
else :
    print("perdu") # affichera perdu si le chiffre est 1, 2, 3, 4 ou 5
```

> Remarques générales de syntaxe : ne pas oublier le « : » après la condition du if. Avec Python, on retrouvera souvent ce « : »avant une instruction. De même l'indentation (le décalage) à la ligne suivant la condition est indispensable.

Autres remarques de syntaxe :

- le # permet d'écrire du texte qui ne sera pas pris en compte par le programme. Typiquement des explications sur le programme.
- le == est différent du =, il représente une condition d'égalité.

5. Définir une fonction : ici encore ne pas oublier le « : » et l'indentation.

```
def double(a): # définit la fonction qui renvoie le double d'un nombre
    return 2*a

print(double(5))
```

6. Les boucles pour utiliser un grand nombre de valeurs, répéter une commande...
   (a) La boucle for

```
def double(a):  # définit la fonction double
    return 2*a


for i in range(1,100):  # pour tous les nombres de 1 à 99
    print("Le double de", i, "est", double(i))
```

   Remarque : la boucle for s'arrête au nombre précédant celui qui est affiché comme fin d'intervalle (ici 99 et non 100).

(b) La boucle while pour utiliser des nombres tant qu'une condition est vérifiée.

```
def double(a):   # définit la fonction double
    return 2*a

a=1              # fixe un point de départ

while (double(a)<100):  # on effectue la commande tant que
                        # le double est plus petit que 100
    a=a+1               # augmente la valeur de a
                        # à chaque passage dans la boucle
    print("Le double de", a, "est", double(a))
```

7. Réaliser des graphiques peut se faire avec différents outils. Ci-dessous un exemple qui affiche une ligne brisée entre trois points à l'aide de la librairie Matplot.

```
import matplotlib.pyplot as plt # importe la librairie de représentation

x = [0, 1, 2]                   # définit 3 abscisses
y = [1, 0, 2]                   # et 3 ordonnées correspondantes
plt.plot(x, y)                  # représente les points et les lignes
plt.show()
```