ALGORITHMIQUE 2de

I Qu'est-ce qu'un algorithme ?

Tout d'abord, le mot algorithme vient du mathématicien perse Al-Khwarizmi (~788-850) un des inventeurs de l'algèbre et du système décimal.

Un algorithme est un enchainement d'instructions permettant, en un temps fini, de résoudre un problème donné.
Voici des exemples d'algorithme :

- chercher un mot dans un dictionnaire ;
- classer le meilleur parcours possible sur une carte ;
- classer des nombres par ordre de grandeur ;
- lister des nombres premiers à un nombre donné ;
- faire la division euclidienne de deux nombres …

Programmer consister à transmettre à un ordinateur, à l'aide des instructions, l'algorithme qu'il doit appliquer pour parvenir au résultat qu'on lui demande.
Il existe des dizaines de langage de programmation (Python, Scratch, Algobox, C++, XCAS, basic TI, Scilab…) qui ont un vocabulaire et des règles syntaxiques propres.
Au lycée, est utilisé le langage **Python** ou un « pseudo-langage » appelé plus communément **langage naturel** dont les instructions sont en français afin que l'on puisse facilement transcrire en langage de programmation (la structure logique de l'algorithme y est mise en avant).

II Généralités

Il y a trois étapes dans un algorithme :

- l'initialisation qui permet de donner une valeur initiale aux variables ou demander à l'utilisateur de le faire ;
- le traitement du problème ;
- une étape de sortie des résultats.

A l'initialisation, on utilise de manière générale la syntaxe suivante :
X <– valeur qui correspond à affecter une valeur à la variable *X* ou aussi celle-ci : *X <– ?* qui demande à l'utilisateur de saisir une valeur pour *X*.
A la sortie, pour afficher un résultat, on écrit l'instruction : *Afficher X*. Cela va afficher la valeur contenue dans la variable *X*. L'instruction *Afficher « … »* va afficher un texte entre guillemets.

Syntaxe des instructions utiles

Langage naturel	Python
Affecter à X la valeur 3	N=3
Afficher X	print(X)
A au carré	A**2
Racine carrée de A	sqrt(A) *Importer* au préalable *les fonctions mathématiques* en saisissant : **from math import***
Quotient de la division euclidienne de A par B	A//B
Reste de la division euclidienne de A par B	A%B

III Les principales structures algorithmiques

Voici une liste des structures principales utilisées sur des exemples.

3.1 L'instruction conditionnelle *if*

Langage naturel	Python
Si *condition* alors *instruction(s)*	if *condition* : *instruction(s)*

Initialisation U <– 3
Traitement Si U<4
Alors U <– 2U
Sortie Afficher *U*

```
U=3
If U<4 :
    U=2U
print(U)
```

Explication de l'algorithme : U=3 est plus petit que 4, c'est vrai. Alors U devient $2U = 2 \times 3 = 6 \Rightarrow U = 6$. Et donc 6 s'affiche.

Remarque
Le mot clé « alors » n'existe en langage Python. C'est l'indentation, c'est-à-dire le décalage automatique du retour à la ligne vers la droite, qui le remplace.

Langage naturel	Python
Si *condition* alors *instruction(s)1* Sinon *instruction(s)2*	if *condition* : *instruction(s)1* else : *instruction(s)2*

Initialisation X <– 5
Traitement Si X<4
alors X <– 2X
Sinon X <– X+7
Sortie Afficher X

```
X=3
If X<4 :
    X=2*X
else :
    X=X+7
print(X)
```

Explication de l'algorithme : X=5 est plus grand que 4. On rentre donc dans le « sinon ». X=5 devient 5+7=12 $\Rightarrow$ X=12. Et donc 12 s'affiche.

Langage naturel	Python
Si *condition1* alors *instruction(s)1* Sinon *condition2* alors *instruction(s)2* Sinon *instruction(s)3*	if *condition* : *instruction(s)1* elif *condition2* else : *instruction(s)3*

Initialisation n <– 6
Traitement Si n<4
m <– 2n
sinon si 2<n<5
alors m <– -2×n+5
Sinon m=n^2
Sortie Afficher m

```
n=6
If n<=2 :
    m=2*n
elif 2<n<5
    m=2*n+5
else :
    m=n**2
print(m)
```

Explication de l'algorithme : n=6 est plus grand que 2 et n'est pas compris entre 2 et 5. On rentre donc dans le dernier « sinon ». m devient 6^2. Et donc 36 s'affiche.

3.2 La boucle *for*

Langage naturel	Python
Pour *variable* allant de … à … *instruction(s)*	for *variable* in range() : *instructions(s)*

Remarque

En python, la fonction range() permet d'énumérer le nombre de passages dans la boucle (bornée). Elle peut être appelée de plusieurs façons :

- range(n), où n est un entier, fait prendre à la variable des valeurs entières de 0 à $n-1$, donc n valeurs ;
- range(n,m), où n et m sont des entiers, fait prendre à la variable des valeurs entières de n à $m-1$;
- range(n,m,k), où n,m et k sont des entiers, fait prendre à la variable des valeurs entières de n à $m-1$, avec un pas de k.

Initialisation U <– 2
Traitement Pour i allant de 1 à 3
U <– 3U
Sortie Afficher U

```
U=2
for i in range(1,4) :
    U=3*U
print(U)
```

Explication de l'algorithme : la 1ère fois (i=1), U devient $3U = 3\times 2 = 6 \Rightarrow U = 6$. .. et la 3ème fois (i=3). Et donc 54 s'affiche.

3.3 La boucle *while*

Langage naturel	Python
Tant que *condition* *instruction(s)*	while *condition :* *instruction(s)*

Initialisation U <– 3
Traitement Tant que U>1
U <– U/2
Sortie Afficher U

```
U=3
while U>1 :
    U=U/2
print(U)
```

Explication de l'algorithme : U=3>1, c'est vrai. U devient $U \div 2 = 3 \div 2 = 1{,}5 \Rightarrow U = 1{,}5$. U=1,5>1, c'est vrai ; au final on a U=0,75. Et donc 0,75 s'affiche.

IV Programmer des fonctions avec Python

On peut simplifier l'écriture des programmes en utilisant des fonctions, sur le modèle des fonction numériques étudiées en mathématiques.

Une fonction est un bloc d'instructions qui a reçu un nom et dont le fonctionnement dépend d'un certain nombre de paramètres. La fonction renvoie un résultat (au moyen de la commande return) ; le programme s'arrête après return.

Voici ci-dessous la structure algorithmique en langage naturel d'une fonction :
def nom_fonction(*paramètre1,paramètre2,…*) :
instructions
return résultat

Dans la console, on appellera : nom_fonction(…)

Exemples

```
def f(x) :
    return(2*x+3)
```

```
def aire_triangle(b,h) :
    return(b*h/2)
```