Quelques notions d'algorithmique

I- Généralités

Un algorithme est une suite finie d'instructions permettant la résolution systématique d'un problème donné.
Un algorithme peut-être décrit en langage « naturel », mais devra être traduit en un langage de programmation pour pouvoir être utilisé.
Un programme est donc la traduction d'un algorithme dans un langage de programmation particulier.

Il existe de très nombreux langages de programmation tels que, parmi bien d'autres, Basic, Fortran, Python, PHP, C, C++, Matlab, assembleur, ainsi que ceux implantés dans les calculatrices (alors dites 'programmables"..)

Les exemples de ce cours impliquant de l'algorithmique seront détaillés de plusieurs façons:

- l'algorithme en langage naturel
- sa traduction en trois langages de programmation
  - -le langage utilisé par **algobox**.
  - -le langage de programmation **Python**.
  - -le langage utilisé par les calculatrices programmables TI-83 (ou similaires)

II-La description en langage « naturel »

De façon générale, on peut considérer trois étapes dans un algorithme

1. L'entrée des données
Dans cette étape figure la lecture des données qui seront traitées au cours de l'algorithme. Dans un programme, ces données peuvent être saisies au clavier ou bien être lues dans un fichier annexe.

2. Le traitement des données
C'est le cœur de l' algorithme. Il est constitué d'une suite d'instructions, parmi lesquelles les différentes opérations sur les données, qui permettent de résoudre le problème.

3. La sortie des résultats
C'est le résultat obtenu qui peut être affiché à l'écran ou enregistré dans un fichier.

Mais il n'y a pas de syntaxe imposée, l'essentiel est que le lecteur comprenne la procédure décrite.

III-Les instructions

1-L'affectation

L'affectation est l'instruction fondamentale: elle se traduit de différentes manière suivant les langages.

Avec algobox	Avec Python	Avec TI
	a=3	3→A

Remarques:

-Avec algobox il faut déclarer la variable avant de l'utiliser.

-Avec Python la déclaration est dynamique: elle se fait dès que l'on écrit l'affectation.

-Sur TI-83 (ou similaire), une seule lettre majuscule peut désigner une variable.

2-Instruction conditionnelle

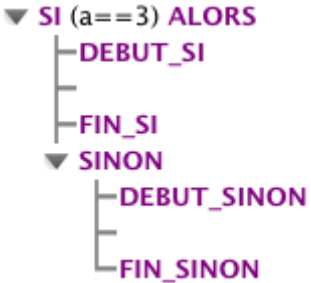
En algorithmique, on est très souvent amené à effectuer des instructions sous certaines conditions. Il s'agit, par exemple, d'effectuer des instructions qui dépendent, la plupart du temps, de la comparaison de deux valeurs affectées à deux variables.

Ces relations de comparaison sont < , > , ≤ , ≥ , = , ≠ (respectivement <, >, <=, >=, ==, != en Python). II est possible d'imbriquer plusieurs tests conditionnels les uns dans les autres.

En langage naturel, une instruction conditionnelle peut se formuler par:

«Si ... alors ... sinon ... »

Avec algobox	Avec Python	Avec TI

	`if condition:`
	`  instruction1`
	`  instruction2`
	`  ....`
	`else:`
	`  instruction1`
	`  instruction2`
	`  ....`
		:IF conditions
		:Then instruction1
		:instruction2
		: . ..
		:Else instruction1
		:instruction2
		: . . .
		:End

Remarques:

-Avec algobox les blocs à exécuter sont délimités par les mots réservés **DEBUT** et **FIN**.

-Avec Python les blocs à exécuter sont repérés par l'indentation (l'indentation consiste en l'ajout de tabulations ou d'espace pour faire ressortir un bloc de code.

-Sur TI-83 (ou similaire), les blocs sont délimités par **Then**, **Else** et **End**

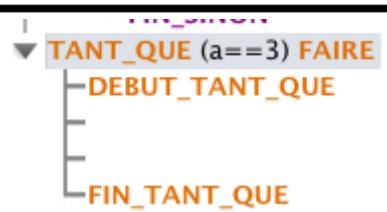
3-Instruction Tant que (ou while en anglais)

En langage naturel, une instruction conditionnelle peut se formuler par:

«Tant que ... »

Le bloc d'instructions qui suit « Tant que ... » sera exécutée tant que la condition sera vraie.

Avec algobox	Avec Python	Avec TI

	`while a==3:`
	`  instruction1`
	`  instruction2`
	`  .....`
		:While conditions
		:instruction1
		:instruction2
		: . . .
		:End

4-Boucle itérative

Lorsque l'on connait d'avance le nombre de boucles qu'il faut répéter, on peut utiliser l'instruction for.

En langage naturel, une instruction conditionnelle peut se formuler par:

«Pour i allant de 5 à 10 ... »

Le bloc d'instructions qui suit « Pour i allant de 5 à 10 » sera exécutée en faisant varier i de la valeur 5 jusqu'à la valeur 10 en incrémentant d'une unité la valeur de i à chaque boucle.

Avec algobox	Avec Python	Avec TI

	`for i in range(5,11):`
	`  instruction`
	`  instruction`
	`  ....`
		:For(I,5,10)
		:instruction1
		:instruction2
		: . . .
		:End

5-Fonctions et procédures

Les fonctions et les procédures sont des sous programmes qui peuvent être utilisés plusieurs fois dans le programme initial: cela facilite la lecture et l'écriture d'un programme complexe.

Malheureusement, sur algobox c'est trop compliqué à mettre en oeuvre (et on peut en utiliser qu'une!). Sur TI-83 (ou similaire) on appelle simplement le programme à l'aide de prgm.

Avec algobox	Avec Python	Avec TI

non	`def MonProg(param1,param2, ... ):`
	`  instruction1`
	`  instruction2`
	`  ......`
	`  return valeur (non obligatoire)`
		:prgm MonProg
		
		(MonProg étant un programme déjà écrit dans la calculatrice)

IV-Des exercices

Exercice 1:

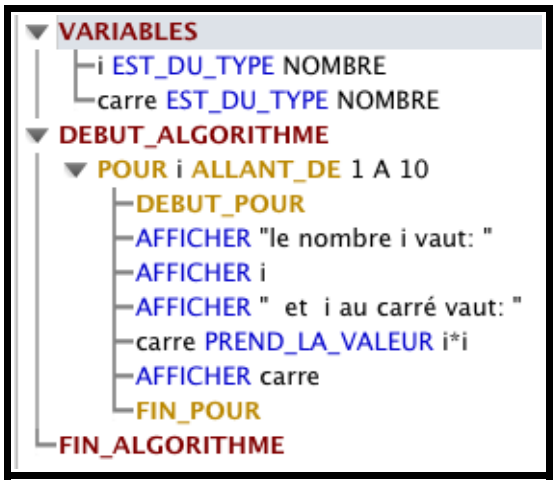
Écrire un algorithme qui affiche la suite des carrés des nombre entiers de 1 à 10.

Correction:

Algorithme en langage naturel:

```
Pour i de 1 à 10
        Afficher i*i
Fin Pour
```

Programme avec algobox



Programme avec Python

```
for i in range(1,11):
    print('i=',i,' et son carré vaut:',i*i,)
```

Avec TI-83

```
:For (I,1,10)
:Disp I*I
:End
```

Exercice 2:

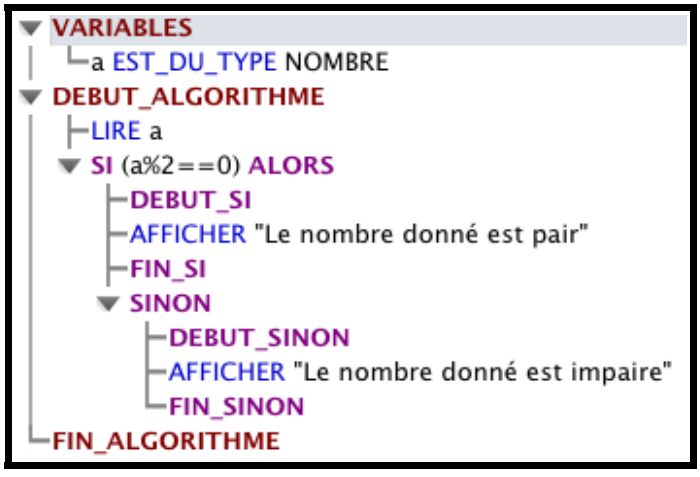
Écrire un algorithme qui demande un nombre entier et qui nous répond si ce nombre est pair ou impair.

Correction:

Algorithme en langage naturel:

```
Lire n
si n est paire
        afficher "n est pair"
sinon
        afficher "n est impair"
fin si
```

Programme avec algobox:



Programme avec Python

```
a=input("a=?")
a=int(a)          #La valeur entrée est une chaine de caractère
                  #Il faut la convertir en entier
if a%2==0:
    print(a," est pair")
else:
    print(a," est impair")
```

Programme avec TI

```
:Prompt A
:If fpart(A/2)=0
:Then
:Disp "A est pair"
:Else
:Disp "A est impair"
:End
```

Exercice 3:

Écrire un algorithme qui demande à l'utilisateur un nombre entier N et qui calcule la somme S=1+2+3+...+N.

Correction:

Algorithme en langage naturel:

```
Lire N
0→S
Pour i de 1 à N
        S+I→S
Fin Pour
Afficher S
```

Programme avec Python:

```
N=input("N=")
N=int(N)
S=0
for i in range(1,N+1):
    S=S+i
print("1+2+...+",N,"=",S)
```

Programme avec algobox:

```
VARIABLES
    N EST_DU_TYPE NOMBRE
    S EST_DU_TYPE NOMBRE
    i EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    LIRE N
    S PREND_LA_VALEUR 0
    POUR i ALLANT_DE 1 A N
        DEBUT_POUR
        S PREND_LA_VALEUR S+i
        FIN_POUR
    AFFICHER "La somme 1+2+...+N="
    AFFICHER S
FIN_ALGORITHME
```

Programme avec TI

```
:Prompt N
:0 →S
:For (I,1,N)
:S+I →S
:End
:Disp S
```