★★ *Chapitre 5 : Algorithmique* ★★

1 Les bases

1.1 Le principe

Toute programmation informatique a commencé par de l'algorithmique .

Mais qu'est-ce que l'algorithmique ?

Simplement une succession d'instructions que l'on donne avec une structure un peu formelle

On doit d'abord indiquer les variables , c'est à dire les lettres qui vont être utilisées dans notre algorithme .

On indique également si ce sont des entiers , des réels , des listes ...

Ensuite , le traitement commence . On donne alors les valeurs initiales de certaines variables , on demande à l'utilisateur de saisir des valeurs , on indique les calculs à faire ...

Et enfin , en sortie , on demande classiquement un affichage .

A retenir

La structure d'un algorithme comprend les Variables , le Traitement et la Sortie

1.2 Le langage naturel

Commençons par le vocabulaire utilisé pour les instructions simples .

Si l'utilisateur doit donner la valeur d'une variable , on utilise : Saisir

Pour donner une valeur à une variable , on utilise : Affecter et on le note $\leftarrow$

Pour demander un affichage , on utilise : Afficher .

Exemple

Variables
x, y : réels
Début de l'algorithme
Saisir x
$y \leftarrow x + 8$
Sorties :
Afficher y

Dans cet algorithme , l'utilisateur doit donner une valeur pour x , l'algorithme calcule x+8 et affiche le résultat .

Astuce

Pour ne pas oublier de variables , on les liste après avoir écrit le corps de l'algorithme .

1.3 En python

A retenir

Saisir X réel se traduit par X =float(input("X ="))
Affecter se traduit par =
Afficher se traduit par print

Avec l'exemple précédent :

```
1 X=float(input("X="))
2 X=X+8
3 print(X)
```

2 Instruction conditionnelle

2.1 Le principe

En mathématiques , le calcul dépend parfois d'une condition . Par exemple , des forfaits de téléphonie dégressifs selon la durée choisie . Il faut donc qu'un algorithme puisse tenir compte de ces conditions .

On utilise alors une boucle conditionnelle .

A retenir

Quand des conditions différentes conduisent à des calculs différents , on utilise la boucle conditionnelle

2.2 Le langage naturel

La structure est simple : Si .... Alors ...Sinon

Le "sinon" n'apparait pas systématiquement .

Exemple

Si un peintre achète moins de 10 litres de peinture , chaque litre lui est facturé 15 euros . S'il achète strictement plus de 10 litres de peinture , il paie chaque litre 12 euros . On obtient alors l'algorithme suivant :

Variables
x , y : réels
Début de l'algorithme
Saisir x
Si $x \leq 10$ **Alors**
| $y \leftarrow 15x$
Sinon
| $y \leftarrow 12x$
Finsi
Sorties :
Afficher y

2.3 En python

A retenir

Si A alors B sinon C se traduit par
if A :
    B
else :
    C

Avec l'exemple précédent :

```
1 X=float(input("X="))
2 if  X<=10:
3       Y=15*X
4 else:
5       Y=12*X
6 print (Y)
```

3 Boucle bornée

3.1 Le principe

Nous devons souvent répéter le même calcul en mathématiques . Par exemple , si on veut déterminer les intérêts obtenus lors d'un placement . En algorithmique , pour éviter d'écrire

plusieurs fois la même instruction , on utilise la boucle POUR .

Dans celle-ci , un compteur indique le nombre de fois où le bloc d'instruction doit être répété .

A retenir

Quand un bloc d'instructions doit être répété plusieurs fois , on utilise la boucle POUR

3.2 Le langage naturel

La structure est simple : Pour i allant de ... à ...

Exemple

On met dans une tirelire 30 euros et on ajoute 2 euros tous les mois. On veut savoir la somme contenue dans la tirelire pour un nombre de mois donnés . On peut utiliser l'algorithme suivant :

Variables
x : réel
n : entier
Début de l'algorithme
Saisir n
$x \leftarrow 30$
Pour *i allant de 1 à n* **Faire**
| $x \leftarrow x + 2$
FinPour
Sorties :
Afficher x

Attention

Le compteur , ici i , n'intervient pas dans le calcul. Il permet seulement de connaitre le nombre de répétitions voulues .

3.3 En Python

A retenir

Pour i allant de 1 à n se traduit par for i in range (1,n+1) :
Python n'utilise pas la dernière borne .

Avec l'exemple précédent :

```
1 N=int(input("N="))
2 X=30
3 for i  in range (1,N+1):
4     X=X+2
5 print(X)
```

4 Boucle non bornée

4.1 Le principe

Comme pour la boucle pour , on utilise la boucle tant que lorsque nous devons répéter un bloc d'instructions . La différence est simple : ici , nous ne connaissons pas le nombre de répétitions à effectuer . Nous avons seulement une condition qui va arrêter le calcul .

A retenir

Quand un bloc d'instructions doit être répété jusqu'à obtention d'une condition d'arrêt , on utilise la boucle Tant que

4.2 Le langage naturel

La structure est simple : Tant que ...

Exemple

On place 150 euros sur un compte à un taux de 1,25 % par an . Au bout de combien d'années la somme dépassera t-elle 300 euros ? On peut utiliser l'algorithme suivant :

Variables
x : réel
n : entier
Début de l'algorithme
$n \leftarrow 0$
$x \leftarrow 150$
Tant que $x \leq 300$ **Faire**
  $x \leftarrow 1,0125x$
  $n \leftarrow n+1$
FinTantque
Sorties :
Afficher n

Attention

Ne pas oublier d'incrémenter n , c'est à dire de le transformer en $n+1$

4.3 En Python

A retenir

Tant que se traduit par while ... :

Avec l'exemple précédent :

```
1 N=0
2 X=150
3 while X<=300:
4     X=1.025*X
5     N=N+1
6 print(N)
```

5 Les fonctions en python

5.1 Le principe

Lorsqu'on doit résoudre un problème complexe en mathématiques , on le divise en plusieurs questions qu'on traite séparément , puis la solution du problème général découle des conclusions intermédiaires .

En python , on a la possibilité de créer des sous-programmes , qu'on appelle fonctions , qui ont ou non des paramètres et qui renvoient ou non un résultat .

On allège ainsi la rédaction d'un programme en faisant appel à ces fonctions .

5.2 Le langage python

A retenir

```
def nomfonction(paramètres) :
        instructions
        return(résultat)
```

Exemple

On cherche à calculer le périmètre d'un triangle rectangle dont on connait la longueur des deux côtés de l'angle droit , a et b . On va commencer par écrire une fonction qui va calculer

la longueur de l'hypoténuse , on va l'appeler hyp , puis appeler cette fonction pour calculer un périmètre .

```
1 from math import*
2 def hyp(a,b):
3         h= sqrt(a**2+b**2)
4         return (h)
5 def perimetre (a , b) :
6       p=a+b+ hyp(a,b)
7       return ( p)
```