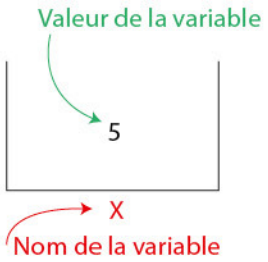
1 Les instructions d'entrée-sortie, l'affectation, les variables

A Notion de variable

Dans un programme, une variable est repérée par son nom et possède une valeur qui évolue au cours de l'exécution du programme.
On peut la schématiser par une boîte qui porte une étiquette et son contenu.



B Type d'une variable

Les langages de programmation distinguent différents types de variables.
Par exemple, une variable peut être de type :

- **nombre entier** ;
- **nombre flottant**, c'est-à-dire nombre à virgule ;
- **chaîne de caractères**, sa valeur est alors une suite ordonnée de caractères ;
- **liste**, c'est-à-dire une suite ordonnée d'objets du langage.

Par exemple : L = [1, 3, 5, 7, 9] ou M = ["a", "e", "i", "o", "u", "y"].

- **booléen**, elle n'a alors que deux valeurs possibles : True (Vrai) et False (Faux).

Par exemple $a < 5$ est un booléen qui a la valeur True si a est strictement inférieur à 5 ou False sinon.
Des opérateurs et des fonctions du langage permettent de travailler avec chaque type de variables.

C L'affectation

L'instruction d'affectation permet d'attribuer une valeur à une variable.

Algorithme	Programme Python
X ← 2	X=2

La variable X est affectée de la valeur 2.

D Les instructions d'entrée-sortie

Les instructions d'entrée-sortie permettent de saisir en entrée et d'afficher en sortie les valeurs des variables.

Algorithme	Programme Python
Saisir X	X=float(input())
Afficher X	print(X)

Ces instructions permettent aussi d'afficher un message, par exemple :
- n=int(input("Nombre d'essais:")) ;
- print("Surface=",S).

INFO

Dans le langage Python, l'instruction d'entrée précise le type de la variable : int (nombre entier), float (nombre à virgule) ou chaîne de caractères lorsque rien n'est précisé.

2 Notion de fonction

A Fonction en programmation

Les langages de programmation donnent la possibilité de définir des fonctions.
Une fonction réalise un traitement et renvoie un résultat, elle peut être appelée à plusieurs reprises par un programme.

Programme Python

```
def f(x,y):
    ...
    return z
```

paramètres : x et y

Une fonction peut admettre aucun, un ou plusieurs paramètres.

B Paramètres

Lors d'un appel à la fonction, des valeurs particulières (arguments) sont données aux paramètres de la fonction.

C Intérêt des fonctions

Programmer des fonctions facilite l'écriture des programmes, permet de les structurer et les rend plus lisibles.

Exemple

Écrire une fonction qui renvoie l'aire d'un triangle

- On connaît la longueur c d'un côté d'un triangle, la hauteur h relative à ce côté et on calcule son aire.
- La fonction **Triangle** a pour paramètres la longueur c et la hauteur h.
Elle renvoie pour résultat l'aire S du triangle.
- Voici le codage de cette fonction en langage Python.

Programme Python

```
1 def Triangle(c,h):
2     S=c*h/2
3     return S
```

- Par exemple, dans la console, on effectue un appel à la fonction **Triangle** lorsque c a pour valeur 7 et b a pour valeur 9.
Ainsi, un triangle dont la longueur d'un côté est 7 et la hauteur relative à ce côté est 9 a une aire égale à 31,5.

```
>>> Triangle(7,9)
31.5
```

Remarque : on peut également écrire un programme qui fait appel à cette fonction.

Programme Python

```
4
5 côté=float(input("Côté ="))
6 hauteur=float(input("Hauteur ="))
7 aire=Triangle(côté,hauteur)
8 print("Aire =",aire)
```

Sur la ligne 7, le programme fait appel à la fonction **Triangle** avec les arguments côté et hauteur.

3 Les instructions conditionnelles

A Condition

Une **condition** est un énoncé qui peut être vrai ou faux.
Par exemple, $a = b$, $x \geqslant 0$, n est pair sont des conditions vraies ou fausses suivant les valeurs des variables a, b, x et n.

B Traitements conditionnels

Dans un programme, on peut tester une condition et, selon que celle-ci est vraie ou fausse, effectuer un traitement ou un autre.
On parle alors de **traitements conditionnels**.

Algorithme	Programme Python

Si condition alors
  Traitement 1
  sinon
  Traitement 2
Fin Si
	if X==2:
	  …
	else:
	  …

Ici, la condition X==2 est vraie lorsque X est égal à 2 et fausse sinon.

C Une situation particulière

Dans un test, on peut ne pas effectuer de traitement dans le cas où la condition est fausse. Dans cette situation, on omet « sinon ».

Algorithme	Programme Python

Si condition alors
  Traitement
Fin Si
	if X==2:
	  …

Exemple

- n désigne un entier naturel.
- Voici un algorithme qui teste si ce nombre est divisible par 11.
- L'algorithme est codé par une fonction **Divise** écrite en langage Python.

Algorithme	Programme Python

Si n est divisible par 11 alors
  C ← "Ce nombre est divisible par 11"
  sinon
  C ← "Ce nombre n'est pas divisible par 11"
Fin Si
	1 def Divise(n):
	2 if n%11==0:
	3 C="Ce nombre est divisible par 11"
	4 else:
	5 C="Ce nombre n'est pas divisible par 11"
	6 return C

- On exécute la fonction **Divise** avec $n = 86\,834$.
On obtient dans la console :

```
>>> Divise(86834)
'Ce nombre est divisible par 11'
```

INFO

La condition n%11==0 peut aussi être considérée comme une variable B de type booléen. B prend la valeur True lorsque n est divisible par 11 et la valeur False sinon.

4 Boucle bornée

A Répéter *n* fois

Une boucle permet de répéter plusieurs fois de suite un même traitement.
Lorsque le nombre *n* d'itérations est connu à l'avance, on parle de **boucle bornée**.

B Compteur

Afin de compter le nombre d'itérations, on utilise un compteur initialisé par exemple à 1 et qui s'incrémente automatiquement de 1 à chaque itération jusqu'à la valeur n.

Algorithme	Programme Python

Pour *i* allant de 1 à *n*
  Traitement
Fin Pour
	for i in range (1, n+1):
	  …

Par exemple, dans la boucle
for i in range (1,6):
…
le compteur i prend successivement les valeurs 1, 2, 3, 4, 5.

Exemple

Étudier l'évolution d'une population

• Deux chercheurs étudient l'évolution d'une population de truites dans un lac de montagne.
Ils constatent une diminution annuelle moyenne de 20 % de cette population.
Afin de sauvegarder l'espèce, ils décident d'introduire chaque année 50 nouvelles truites.
On estime la population initiale du lac à 500 truites.

• Voici :
– un algorithme qui détermine le nombre de truites au bout de n années ($n \geqslant 1$). La variable X est initialisée à 500 ;
– son codage par une fonction **Population** écrite en langage Python.

Algorithme	Programme Python
Pour *i* allant de 1 à *n* X ← 0,8X + 50 Fin Pour	1 def Population(n): 2 X=500 3 for i in range(1,n+1): 4 X=0.8*X+50 5 return X

• On exécute la fonction **Population**.

Avec $n = 12$, on obtient :
```
>>> Population(12)
267.17986918400004
```
Ainsi, après 12 années, le lac compte environ 267 truites.

Avec $n = 25$, on obtient :
```
>>> Population(25)
250.94447329657396
```
Après 25 années, le lac compte environ 251 truites.

Lorsqu'on donne à n des valeurs de plus en plus grandes, on remarque que le nombre de truites de ce lac tend à se stabiliser à 250.

INFO

Dans le langage Python, la fin de l'indentation marque la fin du traitement effectué dans la boucle.

5 Boucle non bornée

A Répéter tant que ...

Dans une boucle, le nombre d'itérations peut dépendre d'une condition.
Le traitement est alors répété tant que la condition est vraie.
Lorsque la condition est fausse, on sort de la boucle et la suite des instructions du programme s'applique.

B Nombre d'itérations

Le nombre d'itérations n'est donc en général pas prévu à l'avance et on parle de **boucle non bornée**.

Algorithme	Programme Python

Tant que condition
  Traitement
Fin Tant que
	while X<2:
	  ...

Ici, tant que la condition X<2 est vraie, on exécute les instructions de la boucle.

Exemple

Simuler une expérience aléatoire

• Au jeu des petits chevaux, pour sortir de l'écurie, on effectue des lancers successifs d'un dé équilibré jusqu'à ce que l'on obtienne 6.

• Voici :

– un algorithme qui effectue une simulation de cette expérience aléatoire et donne le nombre n de lancers effectués. La variable n est initialisée à 1 ;

– son codage par une fonction **L** écrite en langage Python.

Algorithme

$x \leftarrow$ nombre entier aléatoire de 1 à 6
Tant que $x \neq 6$
  $n \leftarrow n+1$
  $x \leftarrow$ nombre entier aléatoire de 1 à 6
Fin Tant que

Programme Python

```
from random import *

def L():
    n=1
    x=randint(1,6)
    while x!=6:
        n=n+1
        x=randint(1,6)
    return n
```

On importe le module random car on utilise la fonction randint

La condition « x différent de 6 » est notée : x!=6

• On exécute la fonction **L**, on obtient dans la console :

```
>>> L()
4
```

Cela signifie que pour cette expérience, on a lancé le dé 4 fois pour sortir le cheval de l'écurie.

INFO

Dans le langage Python, randint(1,6) renvoie un nombre entier aléatoire compris entre 1 et 6.
random() renvoie un nombre aléatoire de l'intervalle $[0 ; 1[$.

Remarque :

Pour écrire ce programme, on peut aussi utiliser la fonction random.
En effet, pour x=random(), la probabilité de l'événement « x<1/6 » est 1/6 et celle de l'événement « x>=1/6 » est 5/6.

```
from random import *

def L():
    n=1
    x=random()
    while x>=1/6:
        n=n+1
        x=random()
    return n
```

INFO

Dans le langage Python, la fin de l'indentation marque la fin des instructions de la boucle.

6 Le type liste

A Éléments d'une liste

La fonction len renvoie le nombre n d'éléments d'une liste L.
Les éléments de L sont notés à l'aide d'un indice : L[0], L[1], …, L[n-1].

Exemple

L=[1,3,5,7,9] est une liste.
L'élément d'indice 3 de L est 7. Ainsi, L[3]=7 , L[0]=1 est l'élément d'indice 0 de L.

B Ajouter un élément à une liste

- Pour ajouter un élément x à la fin d'une liste, on écrit L=L+[x] ou L.append(x).
- On peut aussi insérer un élément à un indice précisé.

Exemple

Ci-contre, on insère l'élément 5 à l'indice 2 dans la liste L.

```
>>> L=[1,3,7,9]
>>> L.insert(2,5)
>>> L
[1, 3, 5, 7, 9]
```

C Supprimer un élément d'une liste

Exemple

Ci-contre, on supprime l'élément d'indice 2 de la liste L.

```
>>> L=[1,3,4,5,7,9]
>>> del L[2]
>>> L
[1, 3, 5, 7, 9]
```

D Parcourir une liste

Voici deux rédactions d'une fonction **Somme**.
Cette fonction a pour paramètre une liste L de nombres et elle renvoie pour résultat la somme des nombres de L.

Programme Python

```
1 def Somme(L):
2     l=len(L)
3     S=0
4     for i in range(0,l):
5         S=S+L[i]
6     return S
```

Programme Python

```
1 def Somme(L):
2     S=0
3     for x in L:
4         S=S+x
5     return S
```

Exemple

```
>>> L=[1,3,5,7,9]
>>> Somme(L)
25
```

E Liste en compréhension

La fonction **Ecarts** a pour paramètres une liste L de nombres et un nombre e. Elle renvoie la liste des écarts abs(x-e) entre les nombres x de L et le nombre e.

Programme Python

```
1 def Ecarts(L,e):
2     E=[abs(x-e) for x in L]
3     return E
```

Exemple

```
>>> L=[1,3,5,7,9]
>>> Ecarts(L,3)
[2, 0, 2, 4, 6]
```

Ici, les éléments de la liste E ne sont pas énumérés, ils sont caractérisés par des propriétés mathématiques et logiques

F Liste en compréhension avec une condition

La fonction **Proches** a pour paramètres une liste L de nombres et deux nombres e et r. Elle renvoie la liste des éléments x de L tels que abs(x-e)⩽r.

Programme Python

```
1 def Proches(L,e,r):
2     E=[x for x in L if abs(x-e)<=r]
3     return E
```

Exemple

```
>>> L=[1,3,5,7,9]
>>> Proches(L,5,3)
[3, 5, 7]
```

Une **proposition** mathématique est un énoncé qui est soit **vrai**, soit **faux**.

Exemples

- « 1 025 est un nombre impair » est une proposition vraie.
- « 275 est divisible par 3 » est une proposition fausse.
- « $f(x)=\begin{cases} e^x+x+1 & \text{si } x\leqslant 0 \\ x+\ln(x)+1 & \text{si } x>0 \end{cases}$ » est une proposition vraie pour certaines valeurs du nombre réel x et fausse pour les autres valeurs

1 Les connecteurs logiques « et », « ou »

A Conjonction

La **conjonction** de deux propositions P et Q est la proposition, notée **P et Q**, qui est vraie uniquement lorsque les propositions P et Q sont toutes les deux vraies.

Exemples

- a désigne un nombre réel.
« $a^2-2a+1=(a-1)^2$ et $|a|\geqslant 0$ » est une proposition vraie.
- « $5<2^3$ et $2^3<7$ » est une proposition fausse. En effet, la proposition « $2^3<7$ » est fausse.

B Disjonction

La **disjonction** de deux propositions P et Q est la proposition, notée **P ou Q**, qui est :
– vraie lorsque l'**une au moins** des propositions P et Q est vraie ;
– fausse lorsque les propositions P et Q sont toutes les deux fausses.

Exemples

- « 65 est divisible par 2 ou 262 est divisible par 2 » est une proposition vraie.
En effet, la proposition « 262 est divisible par 2 » est vraie.
- « 25 est premier ou 49 est premier » est une proposition fausse. En effet, chacune des propositions « 25 est premier », « 49 est premier » est fausse.

2 La négation « non »

La **négation** d'une proposition P est la proposition, notée **non P**, qui est vraie lorsque P est fausse et fausse lorsque P est vraie.

Exemples

- a désigne un nombre réel.
La négation de la proposition « $a<5$ » est la proposition « $a\geqslant 5$ ».
- x désigne un nombre réel.
La négation de la proposition « $x\neq 13$ » est la proposition « $x=13$ ».

3 L'implication « Si ..., alors ... »

A Implication

• Une **implication** est une proposition de la forme « **Si P, alors Q** » où P est une proposition appelée **hypothèse** et Q une proposition appelée **conclusion**.
• On suppose la proposition P vraie, alors :
– si la proposition Q est vraie, l'implication « Si P, alors Q » est vraie ;
– si la proposition Q est fausse, l'implication « Si P, alors Q » est fausse.

Exemple

x désigne un nombre réel.
« Si $x = 3$, alors $x^2 = 9$ » est une implication vraie.
« Si $x = -13$ alors $x^2 < 0$ » est une implication fausse.

B Réciproque

La **réciproque** de l'implication « Si P, alors Q » est l'implication « **Si Q, alors P** ».

Exemple

x désigne un nombre réel.
La réciproque de l'implication « Si $x = 3$, alors $x^2 = 9$ » est l'implication :
« Si $x^2 = 9$, alors $x = 3$ ».
Cette réciproque est fausse pour $x = -3$, en effet $(-3)^2 = 9$ et $-3 \neq 3$.

C Contraposée

• La **contraposée** de l'implication « Si P, alors Q » est l'implication « **Si non Q, alors non P** ».
• Une implication et sa contraposée sont soit toutes les deux vraies, soit toutes les deux fausses.

Exemple

x désigne un nombre réel.
La contraposée de « Si $x = 3$, alors $x^2 = 9$ » est l'implication « Si $x^2 \neq 9$, alors $x \neq 3$ » et cette implication est vraie.
En effet, si on suppose $x^2 \neq 9$ alors x ne peut pas être égal à 3.

4 L'équivalence « ... si, et seulement si ... »

Une **équivalence** est la conjonction de deux implications réciproques :
« Si P, alors Q » et « Si Q, alors P ».
On la note « **P si, et seulement si, Q** » ou « **P équivaut à Q** ».

Exemple

ABCD est un quadrilatère.
« ABCD est un parallélogramme si, et seulement si, ses diagonales se coupent en leur milieu » est une proposition vraie.

5 Condition nécessaire suffisante

A Lorsqu'une implication « Si P, alors Q » est vraie, on dit que :
• Q est une **condition nécessaire à P**. En effet, il faut que la proposition Q soit vraie pour que la proposition P soit vraie.
• P est une **condition suffisante à Q**. En effet, il suffit que la proposition P soit vraie pour que la proposition Q soit vraie.

Exemple

ABCD est un quadrilatère. « Si ABCD est un losange, alors ABCD est un parallélogramme » est une implication vraie.
« ABCD est un parallélogramme » est une condition nécessaire à « ABCD est un losange » et « ABCD est un losange » est une condition suffisante à « ABCD est un parallélogramme ».

B Lorsqu'une équivalence « P si, et seulement si, Q » est vraie, P est une **condition nécessaire et suffisante** à Q.

Exemple

x et y désignent deux nombres réels.
L'équivalence « Le produit xy est nul si, et seulement si, l'un des facteurs x ou y est nul » est vraie. La proposition « Le produit xy est nul » est une condition nécessaire et suffisante à la proposition « L'un des facteurs x ou y est nul ».

6 Les quantificateurs

A Le quantificateur « Pour tout », « Quel que soit »

• L'expression « Pour tout » est appelée **quantificateur universel**.
• On l'emploie pour exprimer que tous les éléments d'un ensemble vérifient une certaine propriété. On dit aussi « Quel que soit ».

Exemple

« Pour tout nombre réel x, $x^2 \geqslant 0$ » est une proposition vraie.

B Le quantificateur existentiel « Il existe »

• L'expression « Il existe » est appelée **quantificateur existentiel**.
• On l'emploie pour exprimer qu'au moins un élément d'un ensemble vérifie une certaine propriété.

Exemple

« Il existe un nombre entier naturel non nul n tel que $\frac{1}{n} - \frac{1}{n+1} < 0{,}01$ » est une proposition vraie.

En effet, pour $n = 10$, $\frac{1}{10} - \frac{1}{11} = \frac{1}{110}$. Or, $\frac{1}{110} \approx 0{,}0091$ donc $\frac{1}{10} - \frac{1}{11} < 0{,}01$.