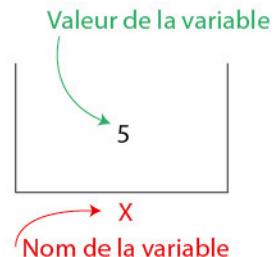


1 Les instructions d'entrée-sortie, l'affectation, les variables

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
<code>X ← 2</code>	<code>X=2</code>

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	<code>X=float(input())</code>
Afficher X	<code>print(X)</code>

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.

Algorithmique et programmation

Exemple 1

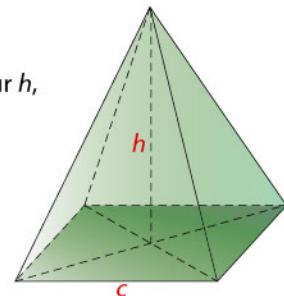
Calculer le volume d'une pyramide à base carrée

- Une pyramide a pour base un carré de côté c , en cm, et pour hauteur h , en cm.

On note S l'aire, en cm^2 , de sa base et V son volume, en cm^3 .

- Voici :

- un algorithme qui permet de saisir en entrée les valeurs de c et h puis qui calcule S , V et affiche le volume V ;
- son codage en langage Python.



Algorithme	Programme Python
Saisir c , h $S \leftarrow c^2$ $V \leftarrow \frac{1}{3} Sh$ Afficher V	<pre>1 c=float(input("c =")) 2 h=float(input("h =")) 3 S=c**2 4 V=1/3*S*h 5 print("V =",V)</pre>

- On exécute le programme avec $c = 3$ et $h = 4,5$.

On obtient dans la console **V = 13,5**.

- Ainsi, le volume de la pyramide est $13,5 \text{ cm}^3$.

Exemple 2

Écrire une séquence d'instructions

- Un opticien propose à ses clients une réduction sur le prix des montures d'un pourcentage égal à leur âge.

Sophie a 16 ans, elle aura donc droit à une réduction de 16 % sur le prix de sa monture.

- Voici :

- un algorithme qui permet de saisir le prix initial, en euro, de la monture, l'âge du client puis qui calcule et affiche le prix réduit de la monture ;
- son codage en langage Python.

Algorithme	Programme Python
Saisir prix, age $reduction \leftarrow \text{prix} \times \text{age}/100$ $\text{prix_reduit} \leftarrow \text{prix} - reduction$ Afficher prix_reduit	<pre>1 prix=float(input("Prix =")) 2 age=int(input("Age =")) 3 reduction=prix*age/100 4 prix_reduit=prix-reduction 5 print("Prix réduit =",prix_reduit)</pre>

Les variables et leurs valeurs sont :

- prix : le prix initial, en euro ;
- age : l'âge du client, en année ;
- reduction : la réduction appliquée, en euro ;
- prix_reduit : le prix après réduction, en euro.

Les valeurs des variables prix et age sont saisies en entrée. On calcule ensuite et on affecte aux variables reduction et prix_reduit les résultats des calculs.

- On exécute le programme avec $\text{prix} = 65$ et $\text{age} = 25$.

On obtient dans la console **Prix réduit = 48,75**.

- Ainsi, après réduction, le client paie sa monture 48,75 €.

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 3

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, $a = b$, $x \geq 0$, 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 4

- L'algorithme suivant permet de saisir un nombre entier naturel n puis détermine et affiche si ce nombre est divisible par 11.
- L'algorithme est codé en langage Python.

$n \% 11$ renvoie pour résultat le reste de la division de n par 11

Algorithme	Programme Python
Saisir n Si n est divisible par 11 alors $C \leftarrow "n$ est divisible par 11" sinon $C \leftarrow "n$ n'est pas divisible par 11" Fin Si Afficher C	1 $n = \text{int}(\text{input}("n ="))$ 2 if $n \% 11 == 0$: 3 $C = "est divisible par 11"$ 4 else: 5 $C = "n'est pas divisible par 11"$ 6 print(n, C)

- On exécute le programme

avec $n = 86\,834$.

On obtient dans la console :

86834 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.

Algorithmique et programmation

Exemple 5

Programmer une fonction définie par intervalles

- f est la fonction qui, à tout nombre réel x , associe le nombre réel $f(x)$ défini par :

$$\begin{cases} f(x) = 1 + \sqrt{x} & \text{si } x \geq 0 \\ f(x) = (x - 1)^2 & \text{si } x < 0 \end{cases}$$

- Voici un algorithme qui détermine le nombre réel y image de x par la fonction f .
- L'algorithme est codé par une fonction F écrite en langage Python.

Algorithme
Si $x \geq 0$ alors $y \leftarrow 1 + \sqrt{x}$ sinon $y \leftarrow (x - 1)^2$ Fin Si

Programme Python
<pre>1 from math import * 2 3 def F(x): 4 if x>=0: 5 y=1+sqrt(x) 6 else: 7 y=(1-x)**2 8 return y</pre>

On importe le module math car on utilise la fonction sqrt (racine carrée).

- On exécute la fonction F .

Avec $x = 2$,
on obtient :
Ainsi $f(2) \approx 2,4$.

Avec $x = -3$,
on obtient :
Ainsi $f(-3) = 16$.

Exemple 6

Programmer des instructions conditionnelles

- Un magasin de reprographie applique le tarif suivant :

Nombre de photocopies	Tarif à l'unité
De 1 à 50	0,15 €
À partir de 51	0,10 €

- Voici un algorithme qui détermine le prix dû pour n photocopies réalisées.
- L'algorithme est codé par une fonction $Prix$ écrite en langage Python.

Algorithme
Si $n \leq 50$ alors $x \leftarrow 0,15x$ sinon $n \leftarrow 7,5 + 0,1(n - 50)$ Fin Si

Programme Python
<pre>1 def Prix(n): 2 if n<=50: 3 x=0.15*n 4 else: 5 x=7.5+0.1*(n-50) 6 return x</pre>

- On exécute la fonction $Prix$.

Avec $n = 45$,
on obtient :
Ainsi le prix dû pour 45 photocopies
est 6,75 €.

Avec $n = 72$,
on obtient :
Ainsi le prix dû pour 72 photocopies
est de 9,70 €.

INFO

Dans le langage Python, l'indentation, c'est-à-dire le décalage vers la droite, permet de repérer les instructions qui font partie d'un traitement conditionnel.

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 7

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 permet de saisir un nombre d'années n , $n \geq 1$, puis qui calcule et affiche le nombre de truites au bout de n années ;
 - son codage en langage Python.

Algorithme	Programme Python
Saisir n $X \leftarrow 500$ Pour i allant de 1 à n $X \leftarrow 0,8X + 50$ Fin Pour Afficher X	1 n=int(input("n =")) 2 X=500 3 for i in range(1,n+1): 4 X=0.8*X+50 5 print("X =",X)

- On exécute le programme.

Avec $n = 12$,

on obtient : $X = 267.17986918400004$

Ainsi, après 12 années,

le lac compte environ 267 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.

Avec $n = 25$,

on obtient : $X = 250.94447329657396$

Après 25 années,

le lac compte environ 251 truites.

INFO

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

Algorithmique et programmation

Exemple 8

Calculer une somme à l'aide d'une boucle

- L'algorithme suivant calcule la somme S des carrés des nombres entiers de 1 à n, avec n nombre entier naturel, $n \geq 1$.

Pour tout nombre entier naturel $n \geq 1$, on pose $S = 1^2 + 2^2 + \dots + n^2$.

- L'algorithme est codé par une fonction Somme écrite en langage Python.

Algorithme	Programme Python
<pre>S ← 0 Pour i allant de 1 à n S ← S + i² Fin Pour</pre>	<pre>1 def Somme(n): 2 S=0 3 for i in range(1,n+1): 4 S=S+i**2 5 return S</pre>

- On exécute le programme avec $n = 5$, voici le tableau de suivi des variables :

i	X	1	2	3	4	5
S	0	1	5	14	30	55

On obtient dans la console : >>> Somme(5)

55

Ainsi $1^2 + 2^2 + 3^2 + 4^2 + 5^2 = 55$.

Exemple 9

Compter le nombre d'occurrences d'un caractère

- Il s'agit de compter dans une phrase le nombre de fois où l'on rencontre un caractère donné.

Voici :

- un algorithme qui détermine le nombre d'occurrences du caractère dans la phrase ;
- son codage par une fonction Nbr écrite en langage Python.

Algorithme	Programme Python
<pre>l ← longueur(phrase) n ← 0 Pour k allant de 0 à l - 1 Si phrase[k] = car alors n ← n + 1 Fin Si Fin Pour</pre>	<pre>1 def Nbr(phrase,car): 2 l=len(phrase) 3 n=0 4 for k in range(0,l): 5 if phrase[k]==car: 6 n=n+1 7 return n</pre>

La variable n compte le nombre d'apparitions du caractère car dans phrase.

- On exécute la fonction avec phrase="Je programme en Python" et car="e".

Voici le tableau de suivi des variables :

k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
phrase[k]	J	e		p	r	o	g	r	a	m	m	e		e	n		P	y	t	h	o	n
n	0	0	1	1	1	1	1	1	1	1	1	1	2	2	3	3	3	3	3	3	3	3

On obtient dans la console : >>> Nbr("Je programme en Python", "e")
3

Ainsi, la lettre e est présente 3 fois dans la phrase saisie.

INFO

La variable phrase est du type chaîne de caractères.

La fonction len renvoie le nombre de caractères / de phrase.

Les caractères sont repérés par phrase[0], phrase[1], ..., phrase [l-1].

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 10

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 ;
 - son codage par une fonction `L` écrite en langage Python.

Algorithme
$n \leftarrow 1$
$x \leftarrow$ 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
1 from random import *
2
3 def L():
4 n=1
5 x=randint(1,6)
6 while x!=6:
7 n=n+1
8 x=randint(1,6)
9 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()`

Cela signifie que pour cette expé-

INFO

rience, on a lancé le dé 4 fois pour sortir le cheval de l'écurie.

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 « $x \geq 1/6$ » est $5/6$.

INFO

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

```
1 from random import *
2
3 def L():
4     n=1
5     x=random()
6     while x>=1/6:
7         n=n+1
8         x=random()
9     return n
```

Algorithmique et programmation

Exemple 11

Déterminer un nombre d'années

- Louis a placé une somme de 1500 € sur son livret d'épargne à un taux de 1,5 % par an.
- L'algorithme suivant détermine le nombre d'années N au bout duquel Louis disposera d'une somme supérieure à 1 600 €.
- L'algorithme est codé par une fonction A écrite en langage Python.

Algorithme	Programme Python
S ← 1500 N ← 0 Tant que S ≤ 1600 S ← 1,015 × S N ← N + 1 Fin Tant que	1 def A(): 2 S=1500 3 N=0 4 while S<=1600: 5 S=1.015*S 6 N=N+1 7 return N

Chaque année, le montant S du livret de Louis augmente de 1,5 %, il est donc multiplié par 1,015.

- On exécute la fonction A, on obtient dans la console : >>> A()

5

Ainsi, au bout de 5 ans, Louis disposera d'une somme supérieure à 1 600 €.

Exemple 12

Tracer des points aléatoires dans un repère

- Une puce effectue des sauts aléatoires. Au départ, elle se situe à l'origine d'un repère du plan.
- Voici :
 - un algorithme qui construit les points de coordonnées ($x ; y$) qui correspondent aux positions successives de la puce.
 - Les sauts ont lieu tant que $x \leq 10$ et $y \leq 10$.
 - son codage en langage Python.

Algorithme	Programme Python
x ← 0 y ← 0 Tant que x ≤ 10 et y ≤ 10 Tracer le point de coordonnées ($x ; y$) x ← x + nombre aléatoire de [0 ; 1[y ← y + nombre aléatoire de [0 ; 1[Fin Tant que	1 from pylab import * 2 x=0 3 y=0 4 while x<=10 and y<=10: 5 plot(x,y,'r.') 6 x=x+random() 7 y=y+random() 8 show()

La condition x<=10 and y<=10 est vraie lorsque x<=10 et y<=10 sont vraies toutes les deux.

INFO

x<=10, y<=10 sont des variables booléennes.

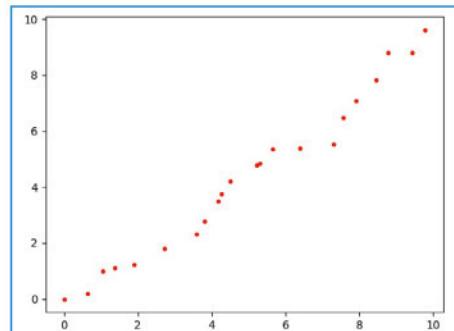
Les opérateurs logiques and (et), or (ou) et not (non) s'appliquent aux variables de ce type.

- On exécute le programme, voici ci-contre un affichage des positions successives de la puce.

INFO

Dans le langage Python, on importe le module pylab pour réaliser des constructions graphiques. L'instruction `plot(x,y,'r')` trace le point de coordonnées ($x ; y$) en rouge.

L'instruction `show()` affiche la figure à l'écran.



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.

Exemple

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

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
```

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 $\text{abs}(x-e)$ entre les nombres x de L et le nombre e.

Exemple

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

Programme Python

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

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 $\text{abs}(x-e) \leq r$.

Exemple

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

Programme Python

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

Algorithmique et programmation

Exemple 13

Travailler avec des listes et leurs indices

- Une roue de loterie est divisée en dix secteurs identiques numérotés de 1 à 10.
- Une expérience aléatoire consiste à faire tourner la roue et à noter le numéro obtenu.
- Voici :
 - un algorithme qui réalise n simulations de cette expérience aléatoire et donne pour résultat la liste des n numéros obtenus ;
 - son codage par la fonction **Loterie** écrite en langage Python.

Algorithme	Programme Python
<pre>L ← [] Pour i allant de 0 à n - 1 a ← nombre entier aléatoire de 1 à 10 L ← L + [a] Fin Pour</pre>	<pre>1 from random import * 2 3 def Loterie(n): 4 L=[] 5 for i in range(0,n): 6 a=randint(1,10) 7 L=L+[a] 8 return L</pre>

L'instruction `L=[]` initialise la liste `L` à la liste vide.

- Par exemple dans la console, on obtient :

```
>>> Loterie(5)
[4, 8, 2, 10, 9]
```

- Ainsi on a simulé 5 fois l'expérience aléatoire et obtenu les numéros 4, 8, 2, 10 et 9.
- On souhaite également obtenir le plus grand et le plus petit des n numéros obtenus.
- Pour cela, on écrit une fonction **Max** et une fonction **Min** qui renvoient respectivement le plus grand et le plus petit des nombres d'une liste de n nombres ($n \geq 2$).
- Voici leurs algorithmes et le codage en langage Python à la suite du précédent.

Algorithme de la fonction Max	Programme Python
<pre>n ← Longueur (L) M ← L[0] Pour i allant de 1 à n - 1 Si M < L[i] alors M ← L[i] Fin Si Fin Pour</pre>	<pre>1 from random import * 2 3 def Loterie(n): 4 L=[] 5 for i in range(0,n): 6 a=randint(1,10) 7 L=L+[a] 8 return L 9 10 def Max(L): 11 n=len(L) 12 M=L[0] 13 for i in range(1,n): 14 if M<L[i]: 15 M=L[i] 16 return M 17 18 def Min(L): 19 n=len(L) 20 m=L[0] 21 for i in range(1,n): 22 if L[i]<m: 23 m=L[i] 24 return m</pre>
Algorithme de la fonction Min	

- Par exemple, dans la console, on obtient :

```
>>> Max(Loterie(5))
8
>>> Min(Loterie(5))
2
```

- Ainsi pour une première simulation de 5 expériences, on a obtenu 8 pour plus grand nombre et pour une seconde simulation de 5 expériences, on a obtenu 2 pour plus petit nombre.

Algorithmique et programmation

Exemple 14

Déterminer une liste d'images

- f est la fonction cube.
- La fonction **F** écrite en langage Python a pour paramètre un nombre x et renvoie pour résultat l'image de x par la fonction f .
- La fonction **Images** a pour paramètre une liste L de nombres.
- Elle renvoie pour résultat la liste des images des nombres x de L par la fonction f .
- Par exemple, dans la console on obtient la liste des cubes des nombres entiers de 1 à 9.

Programme Python

```
1 def F(x):
2     y=x**3
3     return y
4
5 def Images(L):
6     I=[F(x) for x in L]
7     return I
```

```
>>> L=[1,2,3,4,5,6,7,8,9]
>>> Images(L)
[1, 8, 27, 64, 125, 216, 343, 512, 729]
```

Remarque :

Pour une autre fonction f , on adapte l'écriture de la fonction Python **F** et on obtient de même la liste des images par f des nombres d'une liste donnée.

Exemple 15

Travailler avec des listes en compréhension

- La fonction **Moyenne** écrite en langage Python a pour paramètre une liste L de nombres.
- Elle renvoie la moyenne des nombres de cette liste.
- Par exemple, dans la console, on obtient :

```
>>> S=[1,2,5,7,7,8,9,11,15,20]
>>> Moyenne(S)
8.5
```

Programme Python

```
1 def Moyenne(L):
2     n=len(L)
3     S=0
4     for x in L:
5         S=S+x
6     m=S/n
7     return m
```

- La fonction **Diff** saisie à la suite a également pour paramètre une liste L de nombres.
- Elle renvoie la liste E des différences $x-m$ où x désigne un nombre L et m est la moyenne de la liste L .
- Par exemple, dans la console, on obtient :

```
>>> S
[1, 2, 5, 7, 7, 8, 9, 11, 15, 20]
>>> Diff(S)
[-7.5, -6.5, -3.5, -1.5, -1.5, -0.5, 0.5, 2.5, 6.5, 11.5]
```

Programme Python

```
9 def Diff(L):
10     m=Moyenne(L)
11     E=[x-m for x in L]
12     return E
```

- La fonction **Inter** ci-contre a pour paramètres une liste L et un nombre a ($a > 0$).
- Elle renvoie pour résultat une nouvelle liste I formée des nombres x de la liste L tels que $m - a \leq x \leq m + a$, c'est-à-dire qui se trouvent dans l'intervalle $[m - a ; m + a]$.
- Ici, m désigne toujours la moyenne des nombres de L .
- Par exemple, dans la console, **Inter(S,7)** renvoie la liste formée des nombres x de S tels que $8,5 - 7 \leq x \leq 8,5 + 7$, soit $1,5 \leq x \leq 15,5$.

Programme Python

```
14 def Inter(L,a):
15     m=Moyenne(L)
16     I=[x for x in L if m-a<=x and x<=m+a]
17     return I
```

```
>>> S
[1, 2, 5, 7, 7, 8, 9, 11, 15, 20]
>>> Inter(S,7)
[2, 5, 7, 7, 8, 9, 11, 15]
```

Logique et raisonnement

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.
- « $x^2 \leq 16$ » 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.
- « $a^2 \geq 0$ et $|a| \geq 0$ » 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 « $a \geq 5$ ».
- 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 faut 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 , $x^2 \geq 0$ » 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

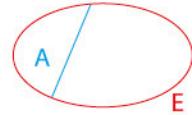
- « Il existe un nombre entier naturel non nul n tel que $\frac{1}{n} - \frac{1}{n+1} < 0,01$ » est une proposition vraie.
- En effet, pour $n = 10$, $\frac{1}{10} - \frac{1}{11} = \frac{1}{110}$. Or, $\frac{1}{110} \approx 0,0091$ donc $\frac{1}{10} - \frac{1}{11} < 0,01$.

7

Raisonnements

A Démonstration par disjonction des cas

Pour démontrer qu'une proposition est vraie pour tout x d'un ensemble E , on peut démontrer qu'elle est vraie pour tout x de A (partie de E) puis qu'elle est vraie pour tout x de E n'appartenant pas à A .



Énoncé : Démontrer que pour tout nombre n de \mathbb{Z} , $n(n + 3)$ est pair.

Solution

1^{er} cas : n est pair, c'est-à-dire $n = 2k$ avec $k \in \mathbb{Z}$.

Alors $n(n + 3) = 2k(2k + 3) = 2K$ avec $K = k(2k + 3)$. Or, $K \in \mathbb{Z}$ donc $n(n + 3)$ est pair.

2^e cas : n est impair, c'est-à-dire $n = 2k + 1$ avec $k \in \mathbb{Z}$.

Alors $n(n + 3) = (2k + 1)(2k + 4) = 2(2k + 1)(k + 2) = 2K'$

avec $K' = (2k + 1)(k + 2)$. Or, $K' \in \mathbb{Z}$ donc $n(n + 3)$ est pair.

Finalement, pour tout nombre n de \mathbb{Z} , $n(n + 3)$ est pair.

B Démonstration par contraposée

Une implication et sa contraposée sont soit toutes les deux vraies, soit toutes les deux fausses. Pour démontrer « Si P , alors Q », on peut démontrer « Si non Q , alors non P ». Ainsi, on suppose non Q vraie et on prouve non P vraie.

Énoncé : Démontrer que si le carré d'un nombre n de \mathbb{Z} est pair, alors n est pair.

Solution

On démontre la contraposée « Si n est impair, alors n^2 est impair ».

On suppose n impair, c'est-à-dire $n = 2k + 1$ avec $k \in \mathbb{Z}$.

Alors $n^2 = (2k + 1)^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1 = 2K + 1$

avec $K = 2k^2 + 2k$. Or, $K \in \mathbb{Z}$ donc n^2 est impair ».

La contraposée est vraie donc la proposition de l'énoncé est vraie.

C Démonstration par l'absurde

Pour démontrer qu'une proposition P est vraie, on suppose que la proposition non P est vraie et on en déduit une contradiction.

Énoncé : Démontrer que $\sqrt{2} \neq 1,414$.

Solution

On suppose que $\sqrt{2} = 1,414$. Deux nombres égaux ont le même carré donc

$2 = 1,414^2 = 1,999396$, on obtient ainsi une contradiction.

Donc la proposition $\sqrt{2} = 1,414$ est fausse.

D Démonstration par contre-exemple

$P(x)$ est une proposition définie pour les éléments x d'un ensemble E .

Pour démontrer que la proposition « Pour tout x de E , $P(x)$ » est fausse, il suffit de trouver un contre-exemple, c'est-à-dire un élément x de E pour lequel $P(x)$ est fausse.

Énoncé : La proposition « Pour tout nombre réel $x > 0$, $\frac{1}{x} \leqslant x$ » est-elle vraie ou fausse ?

Solution

Pour $x = 0,5$, $\frac{1}{x} = 2$ donc $\frac{1}{x} > x$ et la proposition énoncée est fausse.