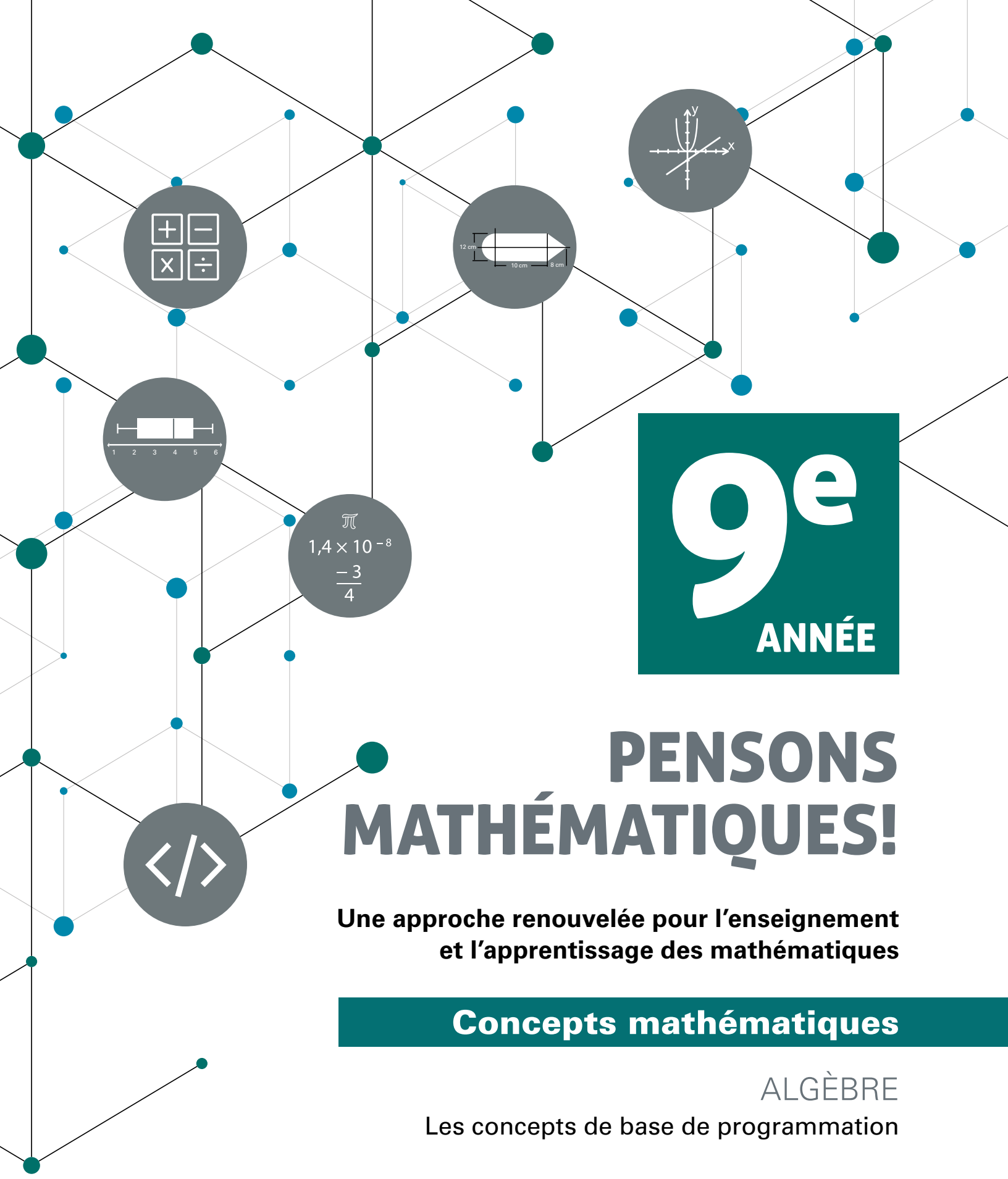


9e ANNÉE

PENSONS MATHÉMATIQUES!

Une approche renouvelée pour l'enseignement et l'apprentissage des mathématiques

Concepts mathématiques

ALGÈBRE

Les concepts de base de programmation

Terminologie liée au concept mathématique

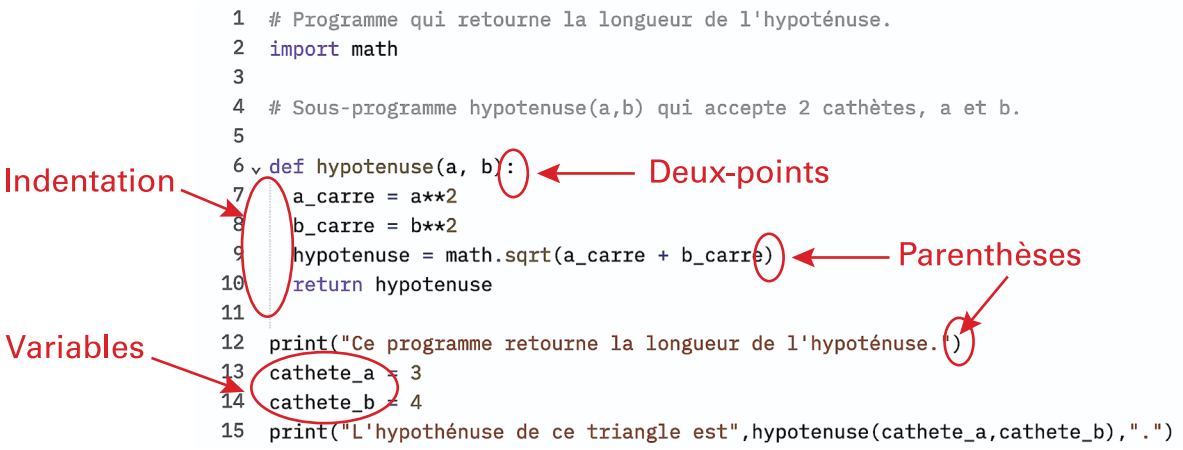
Programmation. La programmation en informatique est l'écriture de codes ou de programmes servant à créer un logiciel. Un programme contient l'ensemble des informations nécessaires pour que le logiciel remplisse une tâche. Le code source est ensuite exécuté par un ordinateur ou même par différents appareils électroniques automatisés.

Langage de programmation. Une notation conventionnelle destinée à construire des algorithmes et produire des programmes informatiques.

> **Note :** Les langages de programmation, d'une manière semblable à un langage courant naturel, sont composés d'un alphabet, d'un vocabulaire, de règles de syntaxe, de significations dans des environnements spécifiques. Ces langages sont par la suite passés dans un environnement de traduction qui permet de rendre la syntaxe compréhensible par l'ordinateur.
>
> **Exemple :** Des exemples de langages de programmation sont Python, C#, JavaScript, C++, Java.

Syntaxe. Un élément essentiel d'un langage de programmation liée à la forme du programme. C'est la syntaxe qui définit les différentes manières dont les éléments du langage peuvent être combinés pour obtenir des programmes.

> **Exemple :** En langage Python :




```
1  # Programme qui retourne la longueur de l'hypoténuse.
2  import math
3
4  # Sous-programme hypotenuse(a,b) qui accepte 2 cathètes, a et b.
5
6  def hypotenuse(a, b):
7      a_carre = a**2
8      b_carre = b**2
9      hypotenuse = math.sqrt(a_carre + b_carre)
10     return hypotenuse
11
12 print("Ce programme retourne la longueur de l'hypoténuse.")
13 cathete_a = 3
14 cathete_b = 4
15 print("L'hypothénuse de ce triangle est",hypotenuse(cathete_a,cathete_b),".")
```

- Les parenthèses sont des symboles utilisés afin de définir une fonction.
- L'indentation indique que l'ajout de tabulations ou d'espaces dans un fichier définit le code appartenant au même bloc.

- La variable ne peut contenir ni espaces ni symboles spéciaux et contient une sensibilité à la casse. C'est-à-dire que la variable abc n'est pas la même que ABC.

Variable. Dans le codage, une variable est un emplacement de stockage temporaire pour des données telles qu'une valeur numérique ou une série de caractères, et les valeurs stockées à cet emplacement varient en fonction des commandes données par le programme. La valeur que représente une variable peut changer tout au long du programme.

Note :
Dans le programme ci-dessous, les variables x et X sont numériques, mais représentent des nombres différents. Certains langages de programmation tels que Python reconnaissent la casse des lettres dans une variable. Par exemple, la variable xyz, n'est pas la même que xYz.

Certains mots sont réservés et ne peuvent pas être utilisés comme nom de variable. Les caractères spéciaux ($, #, @, etc.) et les espaces ne peuvent pas être utilisés.

Note. Malgré leur utilisation semblable, il ne faut pas confondre une **variable en mathématique** et une **variable en programmation**.

Une **variable en mathématique** est un symbole, une lettre ou des mots représentant une valeur qui peut varier en fonction du contexte.

Une **variable en programmation** est un emplacement de stockage temporaire pour des données telles qu'une valeur numérique ou une série de caractères, et les valeurs stockées à cet emplacement varient en fonction des commandes données par le programme.

Exemple :

```
1  # Programme qui utilise des variables.
2
3  Ma_variable = 8
4  x = 12
5  Prenom = "Ton Prénom"
6  X = 15
7
8  print(Ma_variable + x)
9  print(Ma_variable + X)
10
11 print("Bonjour,", Prenom)
```

Affectation

Variables numériques

Variable texte

Sortie du programme

```
20
23
Bonjour, Ton Prénom
```

Les variables numériques et textes sont créés et assignés au moment de l'affectation. Leurs variables sont placées en stockage temporaire. Pour les récupérer, il suffit de les utiliser dans le programme à l'endroit approprié.

Commentaire. Portions du code source ignorées lors de la sortie du code. Cette portion du code est destinée à l'usager qui en fait la lecture afin de comprendre et recevoir des directives.

Note : Différents langages de programmation utilisent différents symboles pour indiquer un commentaire dans le code source. Dans le langage Python ce symbole est le croisillon (#).

Exemple :

```
1  # Ceci est un commentaire en Python et il est ignoré par le programme.
2  # Il sert à titre explicatif seulement.
3
4  Ma_variable = 8 # Ceci est un commentaire à la fin d'une instruction.
5
```

Opérateurs. Des signes ou des symboles qui nous permettent d'effectuer des opérations arithmétiques.

Note : En programmation, les expressions numériques incluant plus qu'une opération arithmétique seront évaluées en respectant la priorité des opérations.

Exemple :

Opération arithmétique	Opérateur dans Python	Exemple	Valeur dans Python
Addition	+	8 + 12	20
Soustraction	-	15 - 2	12
Division	/	18 / 3	6,0
Multiplication	*	11 * 5	55
Exposant	**	2 ** 5	32
Notation scientifique ($x10^n$)	E ou e	1E3 (1e3)	1 000,0
Modulo (le restant d'une division)	%	14 % 3	2
Division entière (la partie entière du quotient d'une division)	//	14 // 3	4

Affectation. Principe par lequel une valeur numérique ou un texte est assigné à une variable en utilisant l'opérateur d'affection =. Le signe = ne sert pas à dire que la valeur est égale au nom de la variable. Il indique qu'on affecte ou qu'on stocke une certaine valeur dans un espace de mémoire.

Notes : En programmation :

- Il est possible d'affecter la valeur d'une variable à une autre.
- Il est possible d'affecter le résultat d'une opération arithmétique à une variable.
- Une variable peut se référencer afin d'effectuer une opération sur elle-même. Par exemple, x = x + 2 augmente la valeur de x par 2.

Exemple :

```
1  x = 3 # J'affecte la valeur de 3 à la variable x en mémoire.
2  a = x # J'affecte la valeur de x à la variable a.
3
4  print(x, "(Valeur de x)") # J'affiche la valeur de x.
5  print(a, "(Valeur de a)") # J'affiche la valeur de a.
6  print("\n") # Je saute une ligne dans la sortie.
7
8  x = x + 2 # J'augmente la valeur de x par 2.
9
10 print(x, "(Valeur de x)") # J'affiche la nouvelle valeur de x.
11 print(a, "(Valeur de a)") # J'affiche la nouvelle valeur de a. (Aucun changement)
12 print("\n") # Je saute une ligne dans la sortie.
13
14 x = x ** 4 # Je mets la valeur de x à l'exposant 4.
15
16 print(x, "(Valeur de x)") # J'affiche la nouvelle valeur de x.
17 print(a, "(Valeur de a)") # J'affiche la nouvelle valeur de a.  (Aucun changement)
18
```

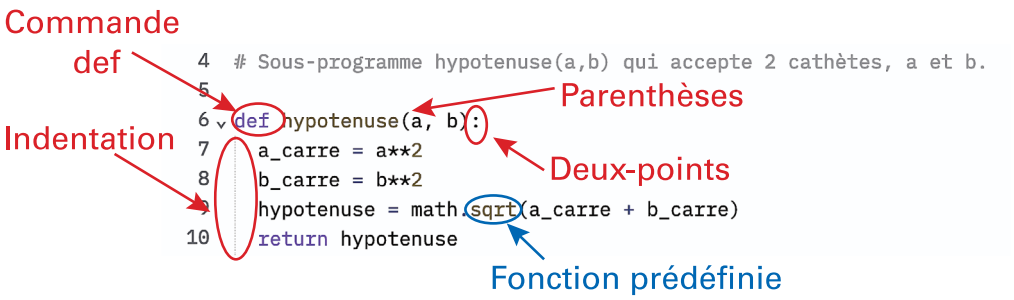
```
3 (Valeur de x)
3 (Valeur de a)


5 (Valeur de x)
3 (Valeur de a)


625 (Valeur de x)
3 (Valeur de a)
>
```

Fonction. Un sous-programme à l'intérieur d'un autre programme qui est nommé et qui regroupe un ensemble d'instructions pour effectuer une tâche précise.

Notes :

- Certaines fonctions sont prédéfinies dans le langage Python tel que `print()`. Il est également possible d'importer des fonctions de modules avec la commande import. Dans l'exemple ci-dessous, le module « `math` » est importé pour donner accès à la fonction de racine carrée `sqrt()`.
- Une fonction définie par un utilisateur doit commencer avec la commande `def` suivie du nom de la fonction et des parenthèses et deux points (`:`).
- Les instructions regroupées par la fonction sont indentées.

Exemple :

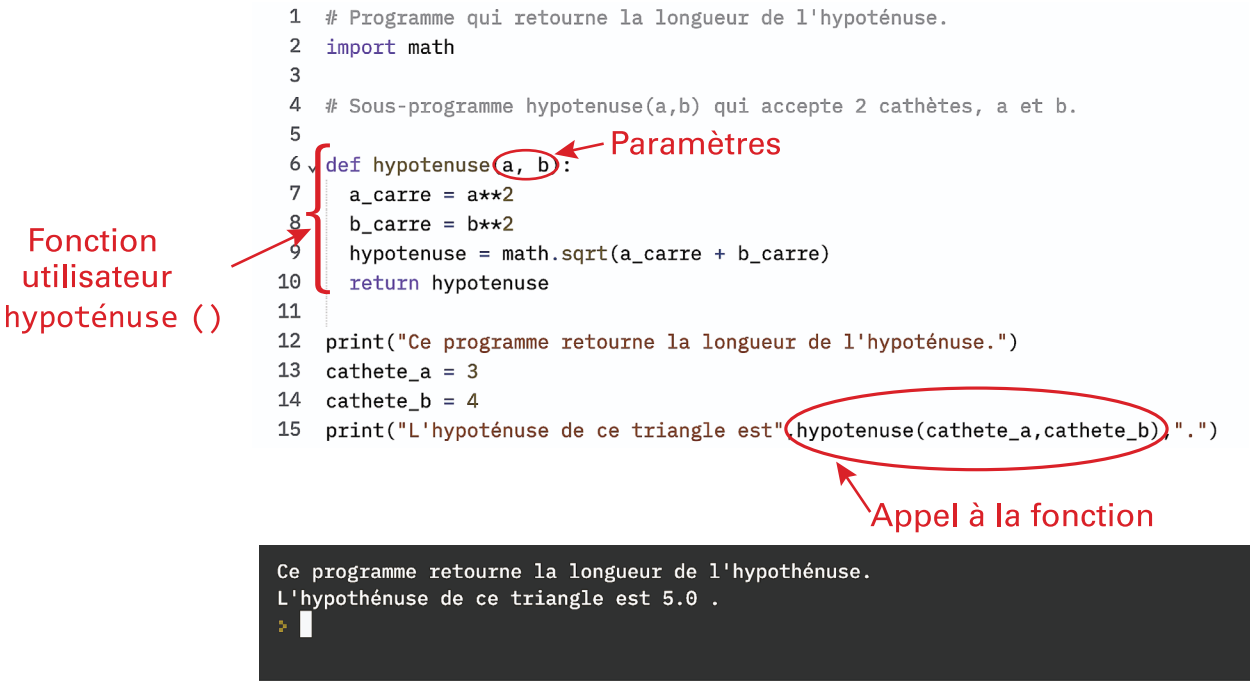


Paramètres. Types de variable dans la définition d'un sous-programme utilisés lors de l'exécution du sous-programme.

Notes :

- Il peut y avoir plusieurs paramètres pour une fonction. Les paramètres peuvent être des nombres ou des caractères (entre guillemets).
- L'ordre des paramètres doit être pareil pour la définition de la fonction et pour l'appel à la fonction.

Exemple : Pour trouver la longueur de l'hypoténuse le sous-programme demande la longueur des deux cathètes a et b comme paramètres.
Le sous-programme retourne la longueur de l'hypoténuse.




```
# Programme qui retourne la longueur de l'hypoténuse.
import math

# Sous-programme hypotenuse(a,b) qui accepte 2 cathètes, a et b.

def hypotenuse(a, b):
    a_carre = a**2
    b_carre = b**2
    hypotenuse = math.sqrt(a_carre + b_carre)
    return hypotenuse

print("Ce programme retourne la longueur de l'hypoténuse.")
cathete_a = 3
cathete_b = 4
print("L'hypoténuse de ce triangle est" ,hypotenuse(cathete_a,cathete_b),".")
```

```
Ce programme retourne la longueur de l'hypothénuse.
L'hypothénuse de ce triangle est 5.0 .
```

Condition. Une opération qui utilise des opérateurs de comparaison pour vérifier si la première expression est égale à (==), plus grand que (>), plus grand ou égale à (>=), plus petit que (<) ou plus petit ou égale à (<=) la deuxième expression. La condition retourne une valeur de « Vrai » (`True`) ou « Faux » (`False`) selon la comparaison.

Opérateur de comparaison. Utilisé dans une condition afin de vérifier si elle est vraie ou fausse.

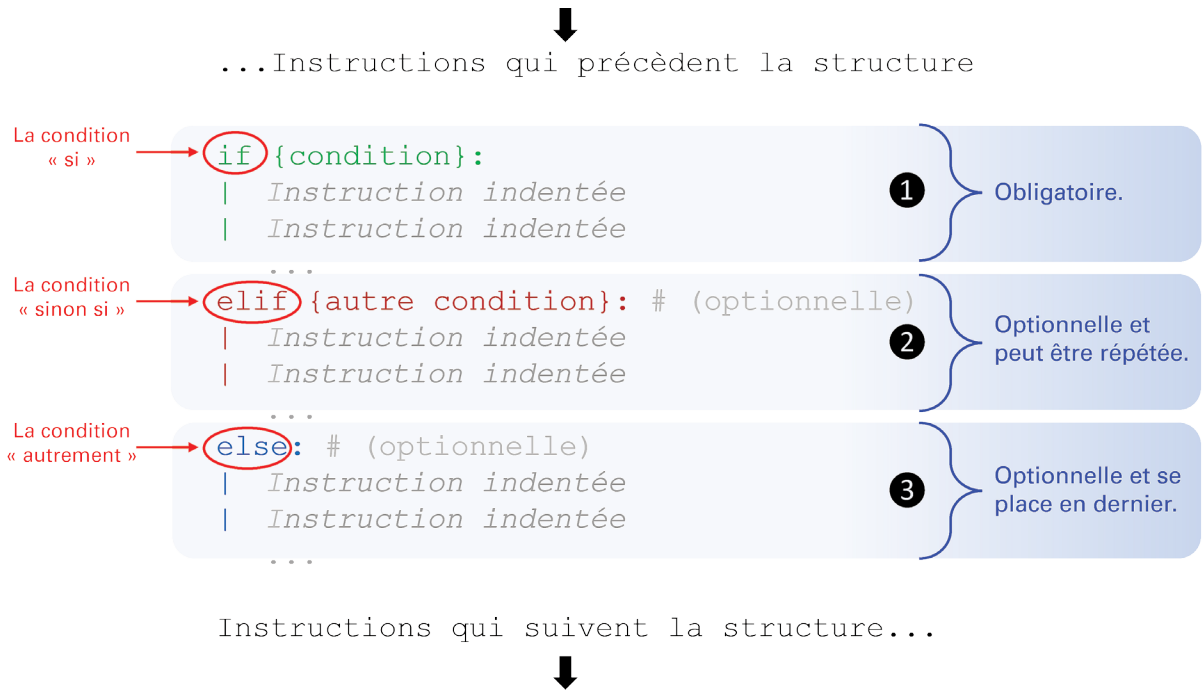
Note : Il y a 5 opérateurs de comparaison de base, soit :

Opérateur	Définition	Exemple	Résultat
== (égal à)	Vérifie si une valeur est égale à une autre.	3+4 == 5+2	True (Vrai)
< (plus petit que)	Vérifie si une valeur est strictement plus petite qu'une autre	3**2 < 2**3	False (Faux)
> (plus grand que)	Vérifie si une valeur est strictement plus grande qu'une autre	20/4 > 18/6	True (Vrai)
<= (plus petit ou égal à)	Vérifie si une valeur est inférieure ou égale à une autre	1E4 <= 10000	True (Vrai)
>= (plus grand ou égal à)	Vérifie si une valeur est supérieure ou égale à une autre.	8%5 > 2	False (Faux)
!= (pas égale à)	Vérifie si une valeur n'est pas égale à une autre.	1/4 != 2/8	False (Faux)

Structure conditionnelle. En programmation, les structures conditionnelles permettent d'exécuter différentes parties du code selon des conditions spécifiques. Les opérateurs de comparaison sont utilisés dans les conditions.

Note : Dans le langage Python :

- La structure conditionnelle contient 3 parties. La première est obligatoire, et deux autres parties sont optionnelles.
- Les instructions regroupées dans une condition doivent être indentées.



Exemple :

a) À l'aide d'une condition « si », démontre que la variable *x* dont la valeur est égale à 6 est supérieure ou égale à 5.

```
1  x = 6
2  # J'utilise la condition if.
3  if x >= 5 :
4      print (x, "est plus grand ou égale à 5.")
5  if x < 5 :
6      print (x, "est plus petit que 5.")
7
```

```
6 est plus grand ou égale à 5.
```

b) À l'aide d'une condition « si…sinon », démontre que la variable *x* dont la valeur est égale à 3 est plus petite que 5.

```
x = 3
# J'utilise la condition if…else.
if x >= 5 :
    print (x, "est plus grand ou égal à 5.")
else :
    print (x, "est plus petit que 5.")

```

```
3 est plus petit que 5.
```

c) À l'aide d'une condition « si… sinon si… sinon », démontre que la variable *x* dont la valeur est égale à 2 est plus petite que 5.

```
x = 2
# J'utilise la condition if…elif…else
if x > 5 :
    print (x, "est plus grand que 5.")
elif x < 5 :
    print (x, "est plus petit que 5.")
else :
    print (x, "est égal à 5.")

```

```
2 est plus petit que 5.
```

Mise en contexte du concept mathématique

EXEMPLE 1

Vérifier des équations

Vérifie les équations suivantes en utilisant Python dans l'environnement de ton choix.

a) $6 \times (3 + 4) \div 2 + 5 = 26$

b) $6 \times 3 + 4 \div 2 + 5 = 25$

c) $6 \times 3 + 4 \div (2 + 5) = 18{,}57$

d) $6 \times (3 + 4) \div (2 + 5) = 6$

J'utilise la fonctionnalité de la console et les opérateurs du langage Python pour évaluer chaque expression et ainsi, vérifier les équations données.

a) 6*(3 + 4) / 2 + 5

b) 6* 3 + 4 / 2 + 5

c) 6* 3 + 4 / (2 + 5)

d) 6* (3 + 4) / (2 + 5)

```
1  # Programme qui vérifie le résultat d'équations arithmétiques.
2  a = 6*(3+4)/2+5
3  print("6x(3+4)÷2+5 =",a)
4
5  b = 6*3+4/2+5
6  print("6x3+4÷2+5 =",b)
7
8  c = 6*3+4/(2+5)
9  print("6*3+4÷(2+5) =",c)
10
11 d = 6*(3+4)/(2+5)
12 print("6*(3+4)÷(2+5) =",d)
13
```

```
6x(3+4)÷2+5 = 26.0
6x3+4÷2+5 = 25.0
6*3+4÷(2+5) = 18.571428571428573
6*(3+4)÷(2+5) = 6.0
```

Ce programme peut être consulté en ligne.

Avec la console et les opérateurs du langage Python, j'ai vérifié chaque équation donnée.

a) $6 \times (3 + 4) \div 2 + 5 = 26$

b) $6 \times 3 + 4 \div 2 + 5 = 25$

c) $6 \times 3 + 4 \div (2 + 5) \approx 18{,}57$

d) $6 \times (3 + 4) \div (2 + 5) = 6$

EXEMPLE 2

Calculer des valeurs

Calcule la valeur de chaque expression numérique en utilisant le langage Python dans l'environnement de ton choix.

a) $2{,}34 \times 10^3 + 1{,}55 \times 10^4$

b) 4^4

c) 2^{-2}

d) $\dfrac{3^4}{3^2}$

Je retranscris les opérations dans un programme en utilisant les bons opérateurs mathématiques.

```
1  # Programme qui vérifie des expressions numériques.
2  a = 2.34E3+1.55E4
3  print ("2.34x10^3 + 1.55x10^4 = ",a)
4
5  b = 4**4
6  print("4^4 =",b)
7
8  c = 2**-2
9  print("2^(-2) =",c)
10
11 d = 3**4/3**2
12 print("(3^4)÷(3^2) =",d)
13
```

```
2.34x10^3 + 1.55x10^4 =  17840.0
4^4 = 256
2^(-2) = 0.25
(3^4)÷(3^2) = 9.0
```

Ce programme peut être consulté en ligne.

J'ai déterminé la valeur de chaque expression numérique.

a) $2{,}34 \times 10^3 + 1{,}55 \times 10^4 = 17\,840$

b) $4^4 = 256$

c) $2^{-2} = 0{,}25$

d) $\dfrac{3^4}{3^2} = 9$

EXEMPLE 3

Créer un programme

Crée un programme qui évalue les expressions suivantes pour une valeur de x égale à 4.

a) x

b) $x + 2$

c) $2x - 3$

d) x^2

e) $4x^5 + 6x^3$

Afin de rendre mon programme compréhensible et exécutable, je vais :

- utiliser des commentaires afin d'expliquer les procédures du code;
- affecter le nombre 4 à la variable x;
- utiliser la fonction `print()` afin d'afficher mes résultats;
- exécuter le programme afin de faire ressortir l'information qui est contenue dans celui-ci.

```
1  # Programme qui évalue des expressions quand x = 4.
2
3  # J'affecte la valeur de 4 à la variable x.
4  x = 4
5  print ("Quand x = 4, x =",x) # Cette fonction affiche la valeur de x
6  print ("Quand x = 4, x + 2 =",x+2) # Cette fonction affiche la valeur de x + 2
7  print ("Quand x = 4, 2^x - 3 =",2*x-3) # Cette fonction affiche la valeur de 2*x - 3
8  print ("Quand x = 4, x^2 =",x**2) # Cette fonction affiche la valeur de x**2
9  print ("Quand x = 4, 4x^5 + 6x^3 =",4*x**5+6*x**3) # Cette fonction affiche la valeur de 4*x**5+6*x**3
```

```
Quand x = 4, x = 4
Quand x = 4, x + 2 = 6
Quand x = 4, 2^x - 3 = 5
Quand x = 4, x^2 = 16
Quand x = 4, 4x^5 + 6x^3 = 4480
```

Ce programme peut être consulter en ligne.

Je détermine à l'aide du langage de programmation Python que :

a) $x = 4$

b) $x + 2 = 6$

c) $2x - 3 = 5$

d) $x^2 = 16$

e) $4x^5 + 6x^3 = 4480$

EXEMPLE 4

Comparer des nombres

Détermine à l'aide d'un logiciel de programmation si :

a) $(-2)^{10}$ est égal à 1/2;

b) $(-2)^{10}$ est plus petit que 1 000;

c) 10 est plus grand que $100^{1/2}$;

d) 35 // 4 est inférieur ou égal à 9.

En utilisant un programme Python, je peux faire l'affectation à des variables pour faciliter la comparaison.

```
1  # 1. Une structure conditionnelle avec if seulement
2  a = -2**20
3  b = 1 / 2
4  if a == b:
5      print(a, "est égal à", b, ".")
6
7  # 2. Une structure conditionnelle avec if...elif
8  a = -2**20
9  c = 1000
10 if a < c:
11     print("a est plus petit que c.")
12 elif a >= c:
13     print("a est plus grand ou égal à c.")
14
15 # 3. Une structure conditionnelle avec if...elif...else
16 d = 10
17 e = 100**(1 / 2)
18 if d > e:
19     print(d, "est plus grand que", e, ".")
20 elif d < e:
21     print(d, "est plus petit que", e, ".")
22 else:
23     print(d, "est égale à", e, ".")
24
25 # 4. Une structure avec seulement if...else
26 f = 35 // 4
27 g = 9
28 if f >= g:
29     print(f, "est plus grand que", g, ".")
30 else:
31     print(f, "est moins que", g, ".")
```

```
a est plus petit que c.
10 est égale à 10.0 .
8 est moins que 9 .
```

Ce programme peut être consulté en ligne.