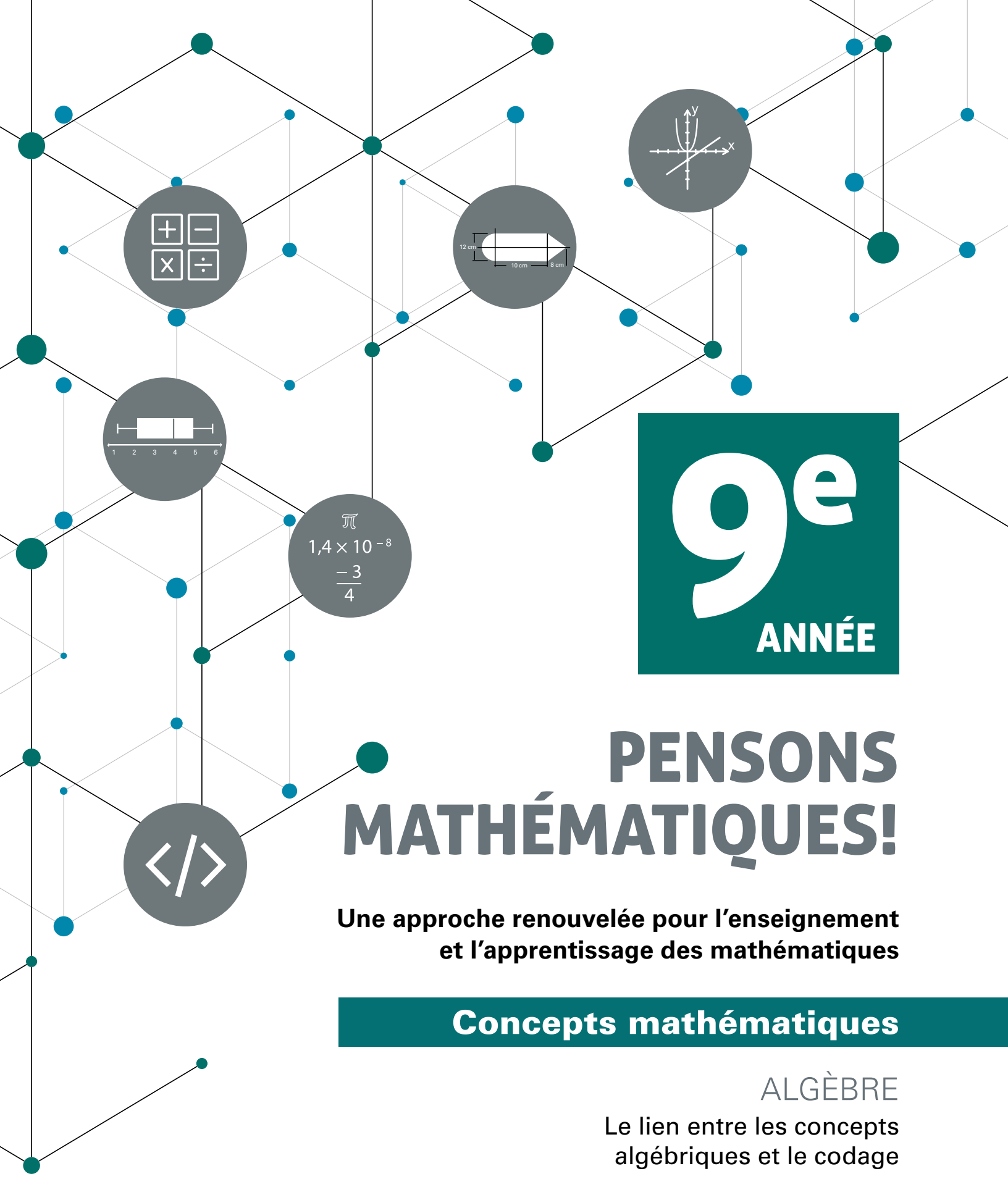


9e ANNÉE

PENSONS MATHÉMATIQUES!

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

Concepts mathématiques

ALGÈBRE

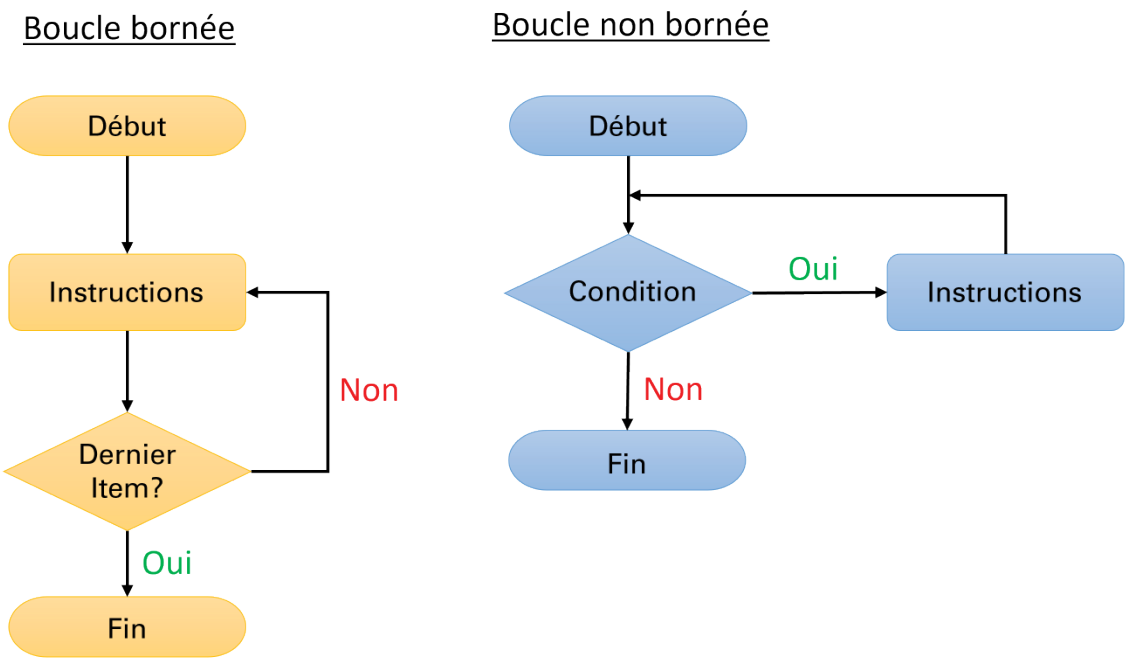
Le lien entre les concepts algébriques et le codage

Terminologie liée au concept mathématique

Boucle. Une structure itérative qui permet d'exécuter plusieurs fois un bloc de code. Cette fonction permet d'exécuter un code tant que la condition donnée est vérifiée.

Note : Il y a deux sortes de boucles. Celle qui doit être exécutée une quantité prédéterminée de fois qu'on appelle **« bornée »** et celle qui est exécutée selon une condition qu'on nomme « non bornée ».

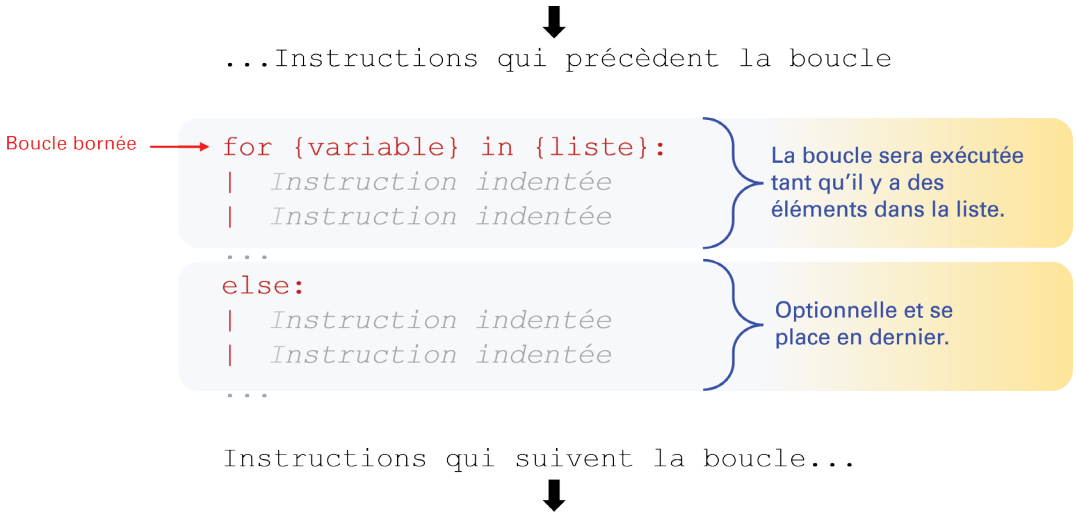
Exemple : Voici la séquence d'exécution pour les deux sortes de boucles dans le langage Python.



Boucle bornée. Une boucle est bornée lorsqu'il est possible de prédire exactement le nombre de fois que la boucle va se répéter.

Note : Dans le langage Python, il existe plusieurs façons de borner une boucle.

- `for x in [0,1,2,3]` : *x* est calculé pour l'ensemble des nombres naturels compris entre 0 et 3 inclusivement.
- `for y in range(0,4)` : *y* est calculé pour l'ensemble des nombres naturels de 0 à 3, incluant le 0 mais excluant le 4. L'utilisation de `range()` est efficace lorsque l'ensemble des nombres est très grand, mais il faut faire attention, car le deuxième chiffre de paramètre n'est pas inclus dans la liste produit.
- Une liste n'a pas à être ordonnée ni contenir des valeurs numériques



- Le bloc « `else` » est exécuté une fois la boucle terminée, si elle est incluse.

Exemple :

```
1  # Un programme qui démontre les deux types de boucles bornées.
2  for x in [0,1,2,3] : # Créer une liste de 0 à 3.
3    print("x =", x ) # Afficher les valeurs de x.
4
5  print("") # Sauter une ligne
6
7  for j in ["Pomme", "Banane", "Cerise", "Raisin", "Pamplemousse"]:
8    print("j =",j) # Afficher les valeurs de j.
9  else:
10   print("Il ne reste plus de fruits.") # Afficher un message à la fin
11
12 print("") # Sauter une ligne
13
14 for y in range (0,4) : # Créer une liste de 0 à 3.
15   print("y =", y) # Afficher les valeurs de y.
```

```
x = 0
x = 1
x = 2
x = 3

j = Pomme
j = Banane
j = Cerise
j = Raisin
j = Pamplemousse
Il ne reste plus de fruits.

y = 0
y = 1
y = 2
y = 3
>
```

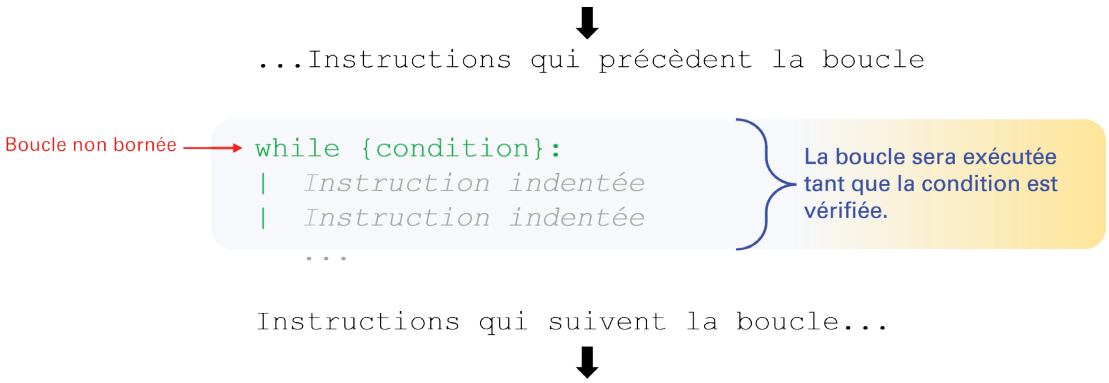
Boucle non bornée. Une boucle est non bornée lorsqu'elle dépend d'une condition pour cesser son exécution. Il n'est pas possible de prédire le nombre de fois qu'elle va s'exécuter et de plus il est possible qu'elle s'exécute à l'infini.

Rappel. Les **opérateurs de comparaison** sont utilisés dans les boucles non bornées dans le langage Python afin de vérifier si une valeur est supérieure, égale, inférieure à une autre valeur ou différente.

Par exemple : i < 6

Notes :
Dans le langage Python :

- La boucle non bornée `while` permet d'exécuter un certain bloc de code tant qu'une condition donnée est vérifiée.
- La boucle utilise les opérateurs de comparaison pour la condition et est exécutée pendant que celle-ci est vraie.



Exemple : Une boucle non bornée dans le langage Python qui compte les nombres entiers de 1 à n.

```
1  # Un programme qui calcule la somme des nombres entiers de 1 à n.
2  n = 10
3  i = 1
4  Somme = 0
5
6  while i <= n  : # Vérifier si "i" est plus petit ou égale à "n"
7    Somme = Somme + i # Ajouter la valeur de i à la Somme courante.
8    i = i + 1  # augmenter i de 1
9    # retourner au début de la boucle
10
11 print("La somme des nombres entiers de 1 à",n,"est",Somme,".")
```

```
La somme des nombres entiers de 1 à 10 est 55 .
```

Mise en contexte du concept mathématique

EXEMPLE 1

Créer des tables de valeurs

À l'aide d'un outil de programmation, crée une table des valeurs pour l'intervalle $1 \leq x \leq 4$ pour chacune des équations suivantes :

a) $y = 2x + 4$

b) $y = 2x^2 + 4$

J'utilise le langage Python pour programmer ce problème. Mon programme doit :

1. Utiliser des commentaires afin d'expliquer les procédures du code.
2. Utiliser la boucle bornée pour l'intervalle de x entre 1 et 4 inclusivement.
3. Utiliser la fonction `print()` afin d'afficher mes résultats.
4. Exécuter le programme afin de faire ressortir l'information qui est contenue dans celui-ci.

STRATÉGIE 1

En utilisant un intervalle de données pour $y = 2x + 4$

```
1  # Stratégie 1 : En utilisant un intervalle de données pour y = 2x + 4
2
3  print("x , y") # Afficher le nom des variables
4  print("-----")
5
6  for x in [1,2,3,4] : # Associer x à un intervalle de données entre 1 et 4 inclusivement.
7      print(x,",", 2*x+4) # Afficher la valeur de x et la valeur correspondante de y = 2x + 4
```

```
x , y
-----
1 , 6
2 , 8
3 , 10
4 , 12
```

Ce programme peut être consulter en ligne.

En utilisant un intervalle de données pour $y = 2x^2 + 4$

```
#Stratégie 1 : En utilisant un intervalle de données pour y = 2x^2 + 4

print("x , y") # Afficher le nom des variables
print("-----")

for x in [1,2,3,4] : # Associer x à un intervalle de données entre 1 et 4 inclusivement.
    print(x,",", 2*x**2+4) # Afficher la valeur de x et la valeur correspondante de y = 2x^2 + 4
```

```
x , y
-----
1 , 6
2 , 12
3 , 22
4 , 36
```

Ce programme peut être consulter en ligne.

STRATÉGIE 2

En utilisant la fonction `range()` pour créer l'intervalle pour $y = 2x + 4$

```
#Stratégie 2 : En utilisant la fonction range() pour créer l'intervalle pour y = 2x + 4

print("x , y") # Afficher le nom des variables
print("-----")

for x in range(1,5) : # Associer x à un intervalle de données entre 1 et 4 inclusivement.
    print(x,",", 2*x+4) # Afficher la valeur de x et la valeur correspondante de y = 2x + 4
```

```
x , y
-----
1 , 6
2 , 8
3 , 10
4 , 12
```

Ce programme peut être consulter en ligne.

En utilisant la fonction `range()` pour créer l'intervalle pour $y = 2x^2 + 4$

```
#Stratégie 2 : En utilisant la fonction range() pour créer l'intervalle pour y = 2x^2 + 4

print("x , y") # Afficher le nom des variables
print("-----")

for x in range(1,5) : # Associer x à un intervalle de données entre 1 et 4 inclusivement.
    print(x,",", 2*x**2+4) # Afficher la valeur de x et la valeur correspondante de y = 2x^2 + 4
```

```
x , y
-----
1 , 6
2 , 12
3 , 22
4 , 36
```

Ce programme peut être consulter en ligne.

EXEMPLE 2

Comparer des nombres

À l'aide d'un outil de programmation, détermine si une valeur de *x* est inférieure, supérieure ou égale à une valeur de *y*.

J'utilise le langage Python pour programmer ce problème. Mon programme doit :

1. Utiliser des commentaires afin d'expliquer les procédures du code.
2. Donner une valeur aux variables *x* et *y*.
3. Utiliser les conditions et les opérateurs de comparaison afin de déterminer la relation entre *x* et *y*.
4. Utiliser la fonction `print()` afin d'afficher mes résultats.
5. Exécuter le programme afin de faire ressortir l'information qui est contenue dans celui-ci.

STRATÉGIE 1

En utilisant seulement la condition `if` :

```
1  #Stratégie 1 : En utilisant seulement la condition if
2  x = 4
3  y = 7
4
5  if x >= y :
6      print ("Si x =",x, "et y =", y, ", x est plus grand ou égal à y. ")
7
8  if x < y :
9      print ("Si x =",x, "et y =", y, ", x est plus petit que y. ")
```

```
Si x = 4 et y = 7 , x est plus petit que y.
>
```

Ce programme peut être consulter en ligne.

STRATÉGIE 2

La condition `if : …elif : …else` :

```
1  #Stratégie 2 : La condition if…elif…else
2  x = 2.5
3  y = 9/2
4
5  if x > y :
6      print ("Si x =",x, "et y =", y, ", x est plus grand que y. ")
7  elif x < y :
8      print ("Si x =",x, "et y =", y, ", x est plus petit que y. ")
9  else :
10     print ("Si x =",x, "et y =", y, ", x est égal à y. ")
```

```
Si x = 2.5 et y = 4.5 , x est plus petit que y.
>
```

Ce programme peut être consulter en ligne.

EXEMPLE 3

Comparer des relations

a) À l'aide d'un outil de programmation, détermine si la relation entre *x* et *y* est linéaire ou non.

x	*y*
2	451
6	759
8	913
11	1 144

J'utilise le langage Python pour programmer ce problème. Mon programme doit :

1. Utiliser des commentaires afin d'expliquer les procédures du code.
2. Associer une variable à chacune des valeurs de *y* de la table de valeurs.
3. Écrire une équation qui permet de déterminer les taux de variation.
4. Utiliser la fonction `print()` afin d'afficher mes résultats pour les taux de variation.
5. Exécuter le programme afin de faire ressortir l'information qui est contenue dans celui-ci.

Je vérifie les sauts dans les deux colonnes *x* et *y* et je compare les taux pour chacun. Je détermine les variables nécessaires pour la résolution du problème.

- Chaque coordonnée (x, y) est dénotée par *xa*, *ya*; *xb*, *yb*; respectivement.
- Les variations entre chaque valeur de *x* sont dénotées par *tvxa*, *tvxb* et *tvxc*.
- Les variations entre chaque valeur de *y* sont dénotées par *tvya*, *tvyb* et *tvyc*.

	x	*y*	
tvxa	*xa* = 1	*ya* = 374	*tvya*
tvxb	*xb* = 2	*yb* = 451	*tvyb*
tvxc	*xc* = 3	*yc* = 528	*tvyc*
	xd = 4	*yd* = 605	

```python
# Programme qui détermine si une relation entre x et y est linéaire ou non
# Chaque coordonnée (x, y) est dénotée par xa, ya; xb, yb; respectivement.
xa = 2
xb = 6
xc = 8
xd = 11
ya = 451
yb = 759
yc = 913
yd = 1144

# Déterminer les équations pour trouver mes variations de x.
# Les variations entre chaque valeur de x sont dénotées par tvxa, tvxb et tvxc.
tvxa = xb - xa
tvxb = xc - xb
tvxc = xd - xc

# Déterminer les équations pour trouver mes variations de y.
# Les variations entre chaque valeur de y sont dénotées par tvya, tvyb et tvyc.
tvya = yb - ya
tvyb = yc - yb
tvyc = yd - yc

# Déterminer mes taux de variations à l'aide des variations en x et y.
TVA = tvya/tvxa
TVB = tvyb/tvxb
TVC = tvyc/tvxc

# Afficher les variations en x.
print ("tvxa =", tvxa)
print ("tvxb =", tvxb)
print ("tvxc =", tvxc)
# Afficher les variations en y.
print ("tvya =", tvya)
print ("tvyb =", tvyb)
print ("tvyc =", tvyc)
# Afficher les taux de variations.
print ("TVA =", TVA)
print ("TVB =", TVB)
print ("TVC =", TVC)

# Vérifier si TVA = TVB = TVC pour la linéarité
if (TVA == TVB) and (TVB == TVC): # L'opérateur "and" permet de combiner des conditions
    print("Puisque les taux de variation sont constants, la relation est linéaire.")
else:
    print("Puisque les taux de variation ne sont pas constants, la relation n'est pas linéaire.")
```

```
tvxa = 4
tvxb = 2
tvxc = 3
tvya = 308
tvyb = 154
tvyc = 231
TVA = 77.0
TVB = 77.0
TVC = 77.0
Puisque les taux de variation sont constants, la relation est linéaire.
```

Ce programme peut être consulter en ligne.

À l'aide de l'outil de programmation, je détermine que le taux de variation est constant et donc la relation est linéaire, et le taux de variation de la relation est 77.

b) À l'aide d'un outil de programmation, détermine si la relation entre *x* et *y* est linéaire ou non.

x	*y*
0	35
2	75
6	150
9	475

J'utilise le langage Python pour programmer ce problème. Mon programme doit :

1. Utiliser des commentaires afin d'expliquer les procédures du code.
2. Associer une variable à chacune des valeurs de *x* et *y* de la table de valeurs.
3. Écrire une équation qui permet de déterminer les taux de variation.
4. Utiliser la fonction `print()` afin d'afficher mes résultats pour les taux de variation.
5. Exécuter le programme afin de faire ressortir l'information qui est contenue dans celui-ci.

Puisque les sauts dans la colonne de *x* sont de et *y* ne sont pas de 1, je dois vérifier les deux colonnes. Je détermine les variables nécessaires pour la résolution du problème.

- Chaque coordonnée (x, y) est dénotée par *xa*, *ya*; *xb*, *yb*; respectivement.
- Les variations entre chaque valeur de x sont dénotées par *tvxa*, *tvxb* et *tvxc*.
- Les variations entre chaque valeur de y sont dénotées par *tvya*, *tvyb* et *tvyc*.

	x	*y*	
tvxa	*xa* = 0	*ya* = 35	*tvya*
tvxb	*xb* = 2	*yb* = 75	*tvyb*
tvxc	*xc* = 6	*yc* = 150	*tvyc*
	xd = 9	*yd* = 475	

```python
# Programme qui détermine si une relation entre x et y est linéaire ou non.

# Afin de rendre le code plus efficace, les calculs sont fait directement
# au moment de l'affectation.
# Chaque coordonnée (x, y) est dénotée par xa, ya; xb, yb; respectivement.
xa = 0
xb = 2
xc = 6
xd = 9
ya = 35
yb = 75
yc = 150
yd = 475

# Déterminer mes taux de variations à l'aide des variations en x et en y.
# yb-ya et xb-xa représentent les variations entre les coordonnées de x et de y respectivement
TVA = (yb-ya)/(xb-xa)
TVB = (yc-yb)/(xc-xb)
TVC = (yd-yc)/(xd-xc)

# J'affiche les variations en y pour l'utilisateur.
print ("TVA =", TVA)
print ("TVB =", TVB)
print ("TVC =", TVC)

# Vérifier si TVA = TVB = TVC pour la linéarité.
if (TVA == TVB) and (TVB == TVC):
    print("Puisque les taux de variation sont constants, la relation est linéaire.")
else:
    print("Puisque les taux de variation ne sont pas constants, la relation n'est pas linéaire.")
```

```
TVA = 20.0
TVB = 18.75
TVC = 108.33333333333333
Puisque les taux de variation ne sont pas constants, la relation n'est pas linéaire.
```

Ce programme peut être consulter en ligne.

À l'aide de l'outil de programmation et du langage Python, je détermine que les taux de variations TVA, TVB et TVC sont respectivement 20, 18,75 et 108,33 et qu'ils ne sont pas constants. Donc la relation n'est pas linéaire.

EXEMPLE 4

Comparer des relations

À l'aide d'un outil de programmation et d'un graphique, vérifie si la relation est linéaire ou non linéaire.

x	*y*
100	75
145	125
48	210
220	42
65	110
290	270
150	25

J'utilise le langage Python pour programmer ce problème. Mon programme doit :

1. Utiliser des commentaires afin d'expliquer les procédures du code.
2. Insérer un outil me permettant de créer un graphique.
3. Insérer mes coordonnées.
4. Afficher mon graphique.
5. Exécuter le programme afin de faire ressortir l'information qui est contenue dans celui-ci.

Matplotlib est une bibliothèque du langage de programmation Python destinée à la création de graphiques.

`plt.scatter` dessinera un nuage de points.

`plt.plot` dessinera un diagramme à lignes brisées.

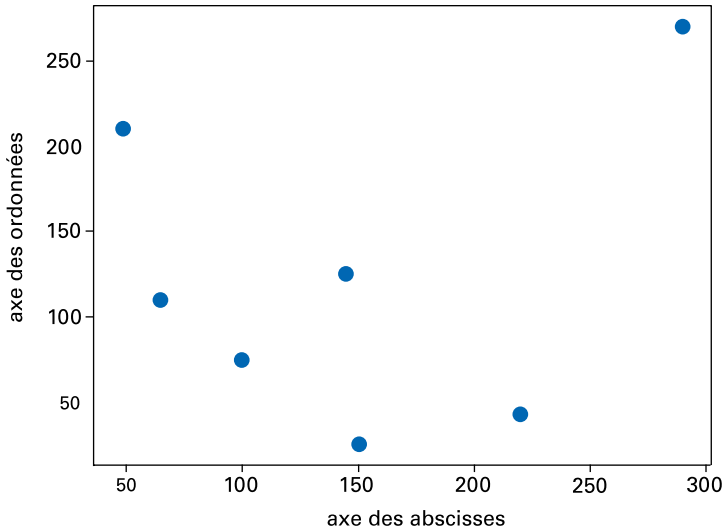
```
# Un programme qui utilise la bibliothèque de mathématique de Python à l'aide de matplotlib
import matplotlib.pyplot as plt

x = [100, 145, 48, 220, 65, 290, 150] # La liste de points "x" pour le graphique
y = [75, 125, 210, 42, 110, 270, 25] # La liste de points "y" pour le graphique

plt.xlabel("axe des abscisses")
plt.ylabel("axe des ordonnées")

# J'indique à Python d'imprimer les données provenant des coordonnées.
plt.scatter(x, y) # Les listes de x et y sont utilisées pour faire le graphique
plt.show() # Afficher le graphique
```

Ce programme peut être consulter en ligne.



À l'aide d'un outil de programmation et du graphique, je constate qu'il n'y a pas de régularité linéaire observable et donc je peux déterminer que la relation est non linéaire.

EXEMPLE 5

Déterminer la quantité de nombres premiers

À l'aide d'un outil de programmation et le sous-programme ci-dessous, détermine la quantité de nombres premiers compris entre 2 et *n* inclusivement, où *n* est une valeur entrée par l'utilisateur lors de l'exécution du code.

Sous-programme. Afin de simplifier et alléger le code, des sous-programmes sont souvent utilisés en programmation. Les sous-programmes sont d'habitude placés au début du programme principal. Dans le langage Python, la définition d'un sous-programme commence avec la commande `def`.

```
1  # Programme qui détermine la quantité de nombres premiers entre 1 et n inclusivement.
2
3  # Sous-programme qui retourne Vrai (True) si "nombre" est un nombre premier.
4  def EstPremier(nombre):
5      Facteurs = False
6      i = 2
7      while i <= (nombre // 2) and not Facteurs:
8          if (nombre % i) == 0:
9              Facteurs = True
10         i += 1
11     if Facteurs:
12         Premier = False
13     else:
14         Premier = True
15     return Premier
```

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

1. Utiliser des commentaires afin d'expliquer les procédures du code.
2. Saisir une valeur `n` du clavier qui servira a vérifié.
3. Utiliser une boucle non bornée afin d'établir mon intervalle de données.
4. Utiliser le sous-programme `EstPremier()` pour vérifier chaque nombre entre 2 et `n` inclusivement.
5. Exécuter quelques fois, le programme afin de vérifier son bon fonctionnement.

```
16
17  # Obtenir le nombre qui représente l'intervalle
18  n = int(input("Entre un nombre pour déterminer la quantité de \nnombres premiers à partir
    de 2 et jusqu'au nombre que tu as choisi : "))
19
20  # Établir les valeurs initiales
21  compteur = 2
22  NombrePremiers = 0
23
24  # vérifier chaque nombre entre 2 et n inclusivement
25  while compteur <= n: # Établir la boucle non bornée
26    if EstPremier(compteur):
27      NombrePremiers = NombrePremiers + 1
28    compteur = compteur + 1
29
30  print("Il y a", NombrePremiers,"nombres premiers entre 2 et",n,"inclusivement.")
```

Entrée de données pour n

Appel au sous-programme

Boucle non bornée qui vérifie les nombres entre 2 et n inclusivement.

Je vérifie que le programme donne le bon montant de nombres premiers entre 2 et n.

```python
# Programme qui détermine la quantité de nombres premiers entre 1 et n inclusivement.

# Sous-programme qui retourne Vrai (True) si "nombre" est un nombre premier.
def EstPremier(nombre):
    Facteurs = False
    i = 2
    while i <= (nombre // 2) and not Facteurs:
        if (nombre % i) == 0:
            Facteurs = True
        i += 1
    if Facteurs:
        Premier = False
    else:
        Premier = True
    return Premier

# Obtenir le nombre qui représente l'intervalle
n = int(input("Entre un nombre pour déterminer la quantité de \nnombres premiers à partir de 2 et jusqu'au nombre que tu as choisi : "))

# Établir les valeurs initiales
compteur = 2
NombrePremiers = 0

# vérifier chaque nombre entre 2 et n inclusivement
while compteur <= n: # Établir la boucle non bornée
    if EstPremier(compteur):
        NombrePremiers = NombrePremiers + 1
    compteur = compteur + 1

print("Il y a", NombrePremiers,"nombres premiers entre 2 et",n,"inclusivement.")
```

```
Entre un nombre pour déterminer la quantité de
nombres premiers à partir de 2 et jusqu'au nombre que tu as choisi : 5
Il y a 3 nombres premiers entre 2 et 5 inclusivement.
>
```

```
Entre un nombre pour déterminer la quantité de
nombres premiers à partir de 2 et jusqu'au nombre que tu as choisi : 10
Il y a 4 nombres premiers entre 2 et 10 inclusivement.
>
```

```
Entre un nombre pour déterminer la quantité de
nombres premiers à partir de 2 et jusqu'au nombre que tu as choisi : 20
Il y a 8 nombres premiers entre 2 et 20 inclusivement.
>
```

Ce programme peut être consulter en ligne.