Chapitre 1 - Introduction au langage Python

I - Programme et variable

1) Programme

Définition 1

Un **programme** est une suite d'instructions élémentaires que l'on exécute séquentiellement (les unes après les autres). Un programme doit toujours commencer par un début et se terminer par une fin.

Pour être exécuté par une machine, un programme doit être écrit dans un langage de programmation. Il existe de nombreux langages.
En cours de SNT, nous utiliserons le langage **Python**.

Exemple

Programme écrit en pseudo-code

```
Variables A, B
Début
    A ← - 5
    B ← 7
    A ← A + B
    B ← A × B
    Afficher A et B
Fin
```

Le même programme écrit en langage Python

```
A = -5
B = 7
A = A + B
B = A * B
print(A)
print(B)
```

2) Langage compilé versus interprété

Pour écrire des programmes informatique, il faut utiliser un **langage de programmation**.
Il existe 2 types de langage :

- **Langage interprété**
  Un programme informatique doit être traduit dans un langage compréhensible par une machine (ordinateur, tablette, smartphone...). La traduction d'un programme en une une suite d'instruction se fait en temps réel, lors de l'exécution. Dans ce cas, le langage est dit "interprété" et l'exécution du programme nécessite la présence d'un interpréteur. L'un des avantages est qu'un même programme peut être exécuté sur plusieurs plateformes différentes, en revanche la traduction (interprétation) du code à chaque exécution a un impact sur les performances.
  Python est un langage interprété.
- **Langage compilé**
  La traduction se fait "une fois pour toutes". Dans ce cas, on parle de langage "compilé". Le code source du programme est traduit par un compilateur qui génère un fichier exécutable. Cela impose donc de compiler le programme pour chaque plateforme de destination.
  Le langage C/C++ est un langage compilé.

3) Variable

Dans un programme, des valeurs doivent régulièrement être stockées pour pourvoir être utilisées ultérieurement. Pour cela, nous devons utiliser des **variables**.

Définition 2

Une variable est une zone dans la mémoire d'une machine permettant de stocker une **valeur**. Une variable doit obligatoirement avoir un **nom**.
Plus simplement, une variable peut être considérée comme "boite" avec un nom dans laquelle on stocke une valeur.

Déclaration de variable

Pour pourvoir utiliser une variable, on dit qu'il faut *déclarer* cette variable.
Dans le programme de la section 1, la première ligne permet de déclarer deux variables dont les noms sont A et B.

Affectation de variable

L'affectation de variable est une instruction élémentaire qui permet de **mettre une valeur dans la boite de la variable**.
Dans un programme écrit en pseudo-code, une affectation est représentée par une flèche allant de droite à gauche ($\longleftarrow$) :

- à gauche de la flèche, se trouve le nom de la variable,
- à droite de la flèche la valeur que l'on souhaite mettre dans la boite variable.

Dans le programme de la section 1 :

- La 3ème ligne permet de mettre la valeur -5 dans la boite de la variable A.
- La 4ème ligne permet de mettre la valeur 7 dans la boite variable B.
- La 5ème ligne permet de mettre la somme des valeurs contenues dans les boites variables A et B dans la boite variable A. La somme est égale à $-5 + 7 = 2$, on met donc la valeur 2 dans la boite variable A.
- La 6ème ligne permet de mettre le produit des valeurs contenues dans les boites variables A et B dans la boite variable B. Le produit est égal à $2 \times 7 = 14$, on met donc la valeur 14 dans la boite variable B.

Voici un tableau permettant de visualiser le contenu des boites variables A et B à la fin de chaque instruction élémentaire représentée par un numéro de ligne.

Numéro de ligne	Variable A	Variable B
1	rien	rien
3	-5	rien
4	-5	7
5	2	7
6	2	14

A la fin de son exécution, le programme affiche les valeurs 2 et 14.

4) Utilisation des variables en Python

Déclaration et initialisation des variables

En Python, la déclaration et l'initialisation d'une variable se font en même temps. Pour déclarer une variable, vous devez lui donner un nom et lui attribuer une valeur initiale.

L'attribution d'une valeur, également appelée **affectation**, se fait à l'aide de l'opérateur =.

Voici quelques exemples :

1. Déclaration et initialisation d'une variable de type 'nombre entier'. Nom de la variable : age, valeur : 25.

```
age = 25
```

2. Déclaration et initialisation d'une variable de type 'nombre réel'. Nom de la variable : prix, valeur : 12.99.

```
prix = 12.99
```

3. Déclaration et initialisation d'une variable de type 'chaine de caractère. Nom de la variable : prenom, valeur : Alice.

```
prenom  = 'Alice'
```

Types de variables en Python

Python est un langage de programmation dynamiquement typé, ce qui signifie que les variables n'ont pas besoin d'être explicitement typées lors de leur déclaration.
Le type de données est déterminé automatiquement en fonction de la valeur attribuée à la variable.
Voici quelques types de données courants en Python :

- **Nombres entiers (en python *int*)**. Exemple : 5, -10, 100).
- **Nombres réels (en python *float*)**. Exemple 3.14, -0.5, 10.0
  Attention, le séparateur de la partie entière et de la partie décimale est un point (système anglo-saxon)
- **Chaîne de caractères ou texte (en python *str*)**. Exemple "Bonjour", 'Python', "123".
- **Booléen (en python *bool*)**. 2 valeurs possibles : **True** ou **False**.

Utilisation des variables

Une fois les variables déclarées et initialisées, vous pouvez les utiliser dans des expressions, des calculs et des opérations. Voici quelques exemples :

1. Utilisation des variables pour effectuer un calcul numérique :

```
prix_unitaire = 2.5
quantite = 23
prix_total = prix_unitaire * quantite
```

2. Utilisation des variables pour manipuler des chaines de caractères.

```
prenom = 'Alice'
message = 'Bonjour, je m'appelle ' + prenom
```

Affichage de variables

En Python, vous pouvez afficher le contenu d'une variable à l'aide de la fonction ***print()***.

1. Exemple 1 :

```
prix_unitaire = 2.5
quantite = 23
prix_total = prix_unitaire * quantite
print(prix_total)
print("Le prix total est de ",prix_total," Euros)
```

```
#OUTPUT
57.5
Le prix total est de 57.5 Euros
```

2. Exemple 2 :

```
prenom = 'Alice'
message = 'Bonjour, je m'appelle ' + prenom
print(message)
```

```
#OUTPUT
Bonjour, je m'appelle Alice
```

Saisie de variables

Pour demander à un utilisateur de saisir des données à l'aide du clavier, vous pouvez utiliser la fonction ***input()***.
Cette fonction attend que l'utilisateur entre une valeur depuis le clavier, puis renvoie cette valeur sous forme d'une **chaîne de caractères** que vous devez stocker dans une variable. Vous pouvez ensuite convertir cette chaîne en d'autres types de données selon vos besoins.

1. Exemple 1 :

```
prenom = input("Entrez votre prenom")
print("Bonjour ",prenom)
```

2. Exemple 2 :

```
age = int(input("Entrez votre age"))
print("Bonjour, vous avez ",age," ans")
```

- L'instruction *int* devant l'instruction *input* permet de convertir ce qui a été saisi par l'utilisateur en un nombre entier.
- Cette opération est obligatoire si l'on souhaite ultérieurement faire des calculs numériques avec la variable *age*.

Règles de nommage des variables

Lorsque vous choisissez des noms pour vos variables en Python, il est important de suivre certaines règles de nommage pour garantir la lisibilité et la compréhension et la maintenabilité de votre code.
Cela facilite la collaboration avec d'autres développeurs et la maintenance du code à long terme.
Voici quelques conventions couramment utilisées pour nommer les variables en Python :

1. **Utiliser des noms significatifs** : Choisissez des noms de variables qui décrivent clairement leur but ou leur contenu. Évitez les noms génériques comme "x", "y", "z" qui ne fournissent pas d'informations sur la donnée qu'ils représentent.
2. **Utiliser des minuscules** : En Python, les variables sont généralement écrites en minuscules. Si le nom de la variable est composé de plusieurs mots, utilisez des underscores pour séparer les mots afin d'améliorer la lisibilité. Par exemple : nom_complet, age_personne, total_vendu.
3. **Éviter les mots réservés** : Évitez d'utiliser des mots réservés du langage Python comme des noms de variables, car ils ont des significations spéciales dans le langage. Par exemple, if, else, for, while, def, import.
4. **Choisir des noms courts et descriptifs** : Essayez de choisir des noms de variables courts mais descriptifs qui capturent l'essence de la donnée qu'ils représentent. Évitez les noms excessivement longs qui pourraient rendre le code difficile à lire.
5. **Suivre la convention snake_case** : En Python, la convention recommandée pour nommer les variables est le snake_case, où les mots sont séparés par des underscores. Cette convention est largement utilisée dans la communauté Python et contribue à la cohérence du code. Par exemple : temperature_celsius, nom_utilisateur, total_commande.

5) Exercices sur les programmes et variables

Exercice 1

Voici un programme python :

```
a = 5.2
b = 7
duree = 60
vitesse = 34.6
a = duree * vitesse
```

1. Combien y-a-t'il de variables dans ce programme ?
2. Donnez le type de chacune des variables de ce programme ?
3. Quelle sera la valeur de la variable a à la fin de l'exécution de ce programme ?
4. Quelle sera la valeur de la variable b à la fin de l'exécution de ce programme ?

Exercice 2

Voici deux programmes python :

```
#PROGRAMME 1
message = 'Hello world !"
print(message)
```

```
#PROGRAMME 2
message = 'Hello world !"
print('message')
```

1. Quelle est le type de la variable *message* ?
2. Que va afficher à l'écran le programme 1 ?
3. Que va afficher à l'écran le programme 2 ?

Exercice 3

Voici deux programmes python :

```
#PROGRAMME 1
prix_unitaire = 2.6
quantite = input("Entrez le nombre de kilogramme de tomates")
prix_total = prix_unitaire * quantite
```

```
#PROGRAMME 2
prix_unitaire = 2.6
quantite = int(input("Entrez le nombre de kilogramme de tomates"))
prix_total = prix_unitaire * quantite
```

L'un de ces deux programmes affichera un message d'erreur. Lequel ? Justifiez votre réponse.

Exercice 4

Voici un programme écrit en pseudo-code.
En t'inspirant du tableau de la page 2 :

1. Construis un tableau permettant de visualiser le contenu des variables A, B et C après l'exécution de chacune des instructions du programme.
2. Quelles sont les valeurs affichées à la fin de l'exécution de ce programme ?
3. Ecris un programme python correspondant à ce programme.

Variables A, B, C
Début
  A ← - 3
  B ← 2,5
  C ← - 1
  A ← A + B
  B ← B × C
  C ← C - B
  Afficher A, B et C
Fin

Exercice 5

Pour cet exercice, vous copierez et collerez les programmes suivants dans l'éditeur **Basthon** et vous completerez les pointillés si besoin.

1. Complétez les pointillés du programme suivant pour qu'il affiche votre prénom.

```
prenom = ...
print(...)
```

2. Complétez les pointillés du programme suivant pour qu'il affiche la somme, la différence, le produit des variables x et y ainsi que le quotient de x par y.

```
x = 5
y = 7
somme = ...
difference = ...
produit = ...
quotient = ...
print(...)
print(...)
print(...)
print(...)
```

3. Complétez les pointillés du programme suivant pour qu'il demande à l'utilisateur de saisir son année de naissance. Vous devrez afficher l'âge de l'utilisateur.

```
annee = ...
age = 2025 - ...
print(...)
```

4. Complétez les pointillés du programme suivant pour qu'il affiche la circonférence du cercle de rayon r et l'aire du disque de rayon r. On prendra $pi = 3.1459$.

```
r = ...
pi = ...
circonference = ...
aire = ...
print(...)
print(...)
```

5. Complétez les pointillés du programme suivant pour qu'il affiche la moyenne des 2 notes de mathématiques *note_maths_1* et *note_maths_2*

```
note_maths_1 = 16.5
note_maths_2 = 15
moyenne = ...
print(...)
```

6. Complétez les pointillés du programme suivant pour qu'il demande à un utilisateur de saisir sa ville de naissance puis d'afficher un message commencant par "Vous êtes né(e) dans la ville de " suivi de la ville saisie par l'utilisateur.

```
ville = ...
message = ...
print(...)
```

Exercice 6

1. Recopiez le code suivant dans votre éditeur Basthon, exécutez le programme et notez les valeurs affichées.

```
from math import *
a = 2
b = a**2
c = sqrt(a)
print("b=",b)
print("c=",c)
```

2. Modifier votre programme en changeant uniquement la valeur de la variable *a* afin de compléter le tableau de valeur suivant :

a	-1	0	1	2	3	5	7	10	15
a**2									
sqrt(a)									

3. En observant les résultats de chacune des exécutions de votre programme :
   - A quelle fonction mathématique correspond l'instruction ***2* ?
   - A quelle fonction mathématique correspond l'instruction *sqrt()* ?
   - Que veut dire "sqrt" ?

Exercice 7

1. Recopiez le code suivant dans votre éditeur Basthon, exécutez le programme et notez les valeurs affichées.

```
from math import *
a = 17
b = 3
c = a/b
d = a//b
e = a%b
print("a=",a)
print("b=",b)
print("c=",c)
print("d=",d)
print("e=",e)
```

2. Modifiez votre programme en changeant la valeur des variables *a* et *b*.
   Vous noterez dans un tableau les valeurs des variables *a*, *b*, *c*, *d* et *e* après chaque exécution du programme.
   - a = 22 et b = 4
   - a = 22 et b = 2
   - a = 56 et b = 5
   - a = 55 et b = 5
   - a = 1234 et b = 56
3. En observant les résultats de chacune des exécutions de votre programme :
   - A quelle fonction mathématique correspond l'instruction '/' ?
   - A quelle fonction mathématique correspond l'instruction '//' ?
   - A quelle fonction mathématique correspond l'instruction ' %' ?

Exercice 8

Dans l'éditeur Basthon, écrire un programme python qui :

- déclare une variable prix et lui affecte le montant 75.
- ajoute la TVA (20%) à ce montant.
- affiche le prix total.

Exercice 9

1. La formule de conversion de degré Celsius en degré Fahrenheit est la suivante :

$$T_{Fahrenheit} = (T_{Celsius} \times 1.8) + 32$$

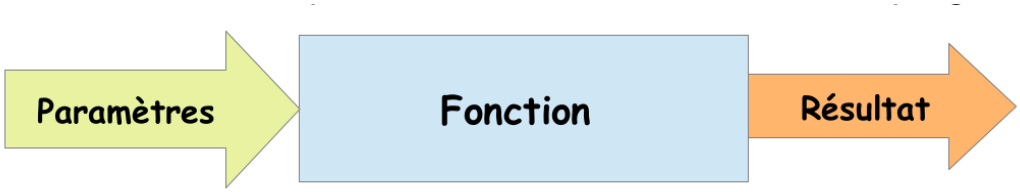
   Dans l'éditeur Basthon, écrire un programme python qui :
   - déclare un variable *t_celsius* et lui affecte la valeur 32.
   - créé une variable *t_fahrenheit* et lui affecte la valeur en degré celsius convertie.
   - affiche les deux températures.
2. Ecrire un programme python qui effectue la conversion inverse :
   degré Fahrenheit $\longrightarrow$ degré Celsius.

II - Fonction

Définition 3

Une **fonction** est une partie d'un programme informatique qui accomplit une tâche spécifique et répétitive.
Une fonction est définie par 4 éléments :

- Un **nom** de fonction.
- Des **paramètres** en entrée de la fonction (des valeurs qui seront utilisées dans le programme de la fonction).
- Une **suite d'instructions** élémentaires.
- Une **valeur de retour**. C'est à dire un résultat calculé par la fonction. Ce résultat pourra ensuite être utilisé dans un programme principal.



Définir une fonction

Cela consiste à déterminer les paramètres, le nom, la valeur de retour et à écrire le code à l'intérieur de la fonction qui permet de retourner la valeur calculée.

Exemple : définir une fonction en Python

Fonction qui calcule la longueur de l'hypoténuse d'un triangle rectangle dont on connaît les longueurs des côtés adjacents à l'angle droit.

- **Nom de la fonction** : calculHypotenuse
- **Paramètres** :
  - a : le premier côté adjacent à l'angle droit.
  - b : le deuxième côté adjacent à l'angle droit.
- **Valeur de retour** (Résultat) : longueur de l'hypoténuse

Code Python de la fonction

```
from math import *

#definition de la fonction
def calculHypotenuse(a, b):
        h = sqrt(a**2+b**2)
        return h
```

- Les paramètres sont écrits entre parenthèses après le nom de la fonction.
- S'il y a plusieurs paramètres, il faut les séparer par des virgules.
- Les paramètres sont des variables dont les valeurs sont inconnues.
- Il faut ajouter *:* après les parenthèses pour indiquer le début du code de la fonction.
- Chaque ligne du code de la fonction doit commencer par une tabulation.
- Pour retourner une valeur, il faut utiliser le mot clé *return* suivi de la valeur calculée par la fonction.

Appeler une fonction

Cela consiste à utiliser dans un programme principal la fonction définie avec des paramètres particuliers. **On peut appeler autant de fois la fonction qui a été définie sans reécrire le code de la fonction**. C'est l'intérêt principal d'une fonction.

Exemple d'appel de fonction en Python

En langage Python, il est possible d'appeler deux types de fonctions :

- Les fonctions définies par un utilisateur de Python. Par exemple, la fonction *calculHypotenuse*.
- Les fonctions déjà existantes en Python et contenues dans une bibliothèque de fonctions. Exemples de bibliothèques de fonctions : math, random, networkx, folium, pillow, tkinter, micro :bit.

Exemple de programme python avec définition et appel de fonction.

```
from math import *

#definition de la fonction
def calculHypotenuse(a, b):
        h = sqrt(a**2+b**2)
        return h

#PROGRAMME PRINCIPAL

#appel de fonction calculHypotenuse avec les parametres 3 et 4
h1 = calculHypotenuse(3, 4)
print("h1=",h1)

#appel de fonction calculHypotenuse avec les parametres 7 et 8
h2 = calculHypotenuse(7, 8)
print("h2=",h2)
```

- La fonction *sqrt* appartient à la bibliothèque *math*. Afin de pouvoir utiliser cette fonction, il faut d'abord importer la bibliothèque *math* à l'aide de l'instruction *from math import **
- Pour appeler une fonction, il faut remplacer les paramètres par des valeurs connues. Dans le premier appel de la fonction *calculHypotenuse*, le paramètre *a* est remplacé par la valeur 3, et le paramètre *b* est remplacé par la valeur 4.
- **Dans le code de la fonction**, le calcul de la valeur *h* sera fait avec les valeurs 3 et 4. La variable *h* contiendra donc la valeur 5.
- La fonction retourne la valeur contenue dans la variable *h*, donc la fonction retourne 5.
- **Dans le programme principal**, la valeur retournée par la fonction *calculHypotenuse* est affectée dans une nouvelle variable *h1*. La variable *h1* contient donc la valeur 5.
- La fonction *calculHypotenuse* est appelée une deuxième fois avec de nouveaux paramètres : 7 et 8. Il n'y a pas besoin de reécrire une deuxième fois le code complet de la fonction : c'est l'intérêt d'une fonction.

Exercices sur les fonctions

Exercice 10

Voici 4 définitions de fonction en python. Pour chacune de ces fonctions vous donnerez : son nom, ses paramètres et le nom de la variable retournée.

```
#FONCTION 1
def moyenne(n1, n2, n3):
        m = (n1+n2+n3)/3
        return m

#FONCTION 2
def aire_rectangle(longueur, largeur):
        aire = longueur * largeur
        return aire

#FONCTION 3
def imc(masse, taille):
        i = masse / (taille**2)
        return i

#FONCTION 4
def age():
        annee = int(input("Entrez votre annee de naissance"))
        age = 2025 - annee
        return annee
```

Exercice 11

Voici la définition de 3 fonctions python. Chacune de ces définitions comporte une erreur : à vous de la trouver et de la corriger.

```
#FONCTION 1
def fonction_affine(a, b, x)
        y = a*x +b
        return y

#FONCTION 2
fonction_lineaire(a, x):
        y = a*x
        return y

#FONCTION 3
def fonction_carre(x):
        y=x**2
```

Exercice 12 Voici la definition d'un fonction python calculant l'image d'une valeur x par une fonction affine de paramètres a et b.
Le programme principal appelle cette fonction 3 fois. Chaque appel comporte une erreur : à vous de la trouver et de la corriger.

```
#Definition de la fonction
def fonction_affine(a, b, x):
        y = a*x +b
        return y

#Programme principal

#APPEL 1
y1 = fonction_affine(2,3)
print("Image de 5 par la fonction affine : ",y1)

#APPEL 2
y2 = fonction_aff(2,3,5)
print("Image de 5 par la fonction affine : ",y2)

#APPEL 3
fonction_affine(2,3,5)
print("Image de 5 par la fonction affine : ",y3)
```

Exercice 13

Voici la définition en python de deux fonctions mystères.
Pour chacune de ces 2 fonctions, indiquez ce que fait cette fonction et proposez un nouveau nom de fonction et des noms de paramètres plus explicites.

```
#Definition 1
def fonction_mystere_1(x, y):
        z = (a*b)/2
        return z

#Definition 2
#parametre aujourdhui : nombre entier de 0 a 6.
# 0 represente le 1er jour de la semaine : lundi
# 6 represente le dernier jour de la semaine : dimanche
def fonction_mystere_2(aujourdhui,n):
        r = (aujourdhui + n) % 7
        return r
```

Exercice 14 **Cercle, disque, boule**

1. Voici un code Python (incomplet) qui définie une fonction calculant la circonférence d'un cercle de rayon r (exprimé en cm) puis un programme principal qui appelle cette fonction pour calculer la circonférence d'un cercle de rayon 2 cm puis 18 cm.

```
from math import *

#Definition de la fonction circonference
def circonference(r):
        c = ... * math.pi * ...
        return ...

#PROGRAMME PRINCIPAL avec appel des fonctions
r1 = 2
r2 = 18
c1 = circonference(...)
print("circonference d'un cercle de rayon ",r1," cm = ",..., "cm")
c2 = circonference(...)
print("circonference d'un cercle de rayon ",r2," cm = ",..., "cm")
```

Copier le code ci-dessus dans l'éditeur basthon et complétez les pointillés dans le code pour que celui-ci soit fonctionnel..

2. Sur le même modèle que la question précédente, écrire la définition d'une fonction Python calculant l'aire d'un disque de rayon r (exprimé en cm). Dans un programme principal, vous appelerez votre fonction pour calculer l'aire d'un disque de rayon 5 cm puis d'un disque de rayon 14.5 cm.
3. Sur le même modèle que la question précédente, écrire la définition d'une fonction Python calculant le volume d'une boule de rayon r (exprimé en cm). Dans un programme principal, vous appelerez votre fonction pour calculer le volume d'une boule de rayon 2.5 cm puis d'une boulle de rayon 10.3 cm.

Exercice 15 **Concentration en soluté**

La concentration en masse γ (en $g.L^{-1}$) d'un soluté est la masse m (en g) de soluté dissous dans le volume V (en L) de la solution, avec :

$$\gamma = \frac{m}{V}$$

1. Dans votre éditeur basthon, définir une fonction python prenant en paramètres la masse d'un soluté m (exprimée en g) et le volume d'une solution V (exprimé en L) et qui retourne la concentration en masse du soluté (exprimée en $g.L^{-1}$).

```
def concentration(m,v):
        ...
        return ...

#PROGRAMME PRINCIPAL avec appel des fonctions
...
...
```

2. Dans votre programme principal, appelez votre fonction pour calculer la concentration en soluté de 1 g de sel dans 1/2 L de soupe et afficher le résultat. Vérifiez que vous obtenez bien $2\ g.L^{-1}$.
3. Un flacon d'antibiotique de 60 mL indique qu'après ajout d'eau jusqu'au trait, la solution fabriquée aura 500 mg d'antibiotique pour 5 mL de solution.
   En appelant votre fonction python avec les bons paramètres, calulez et affichez la concentration (en g/L) de la solution ?

Exercice 16

Dans un repère orthonormé, la distance euclidienne entre un point A et un point B de coordonnées respectives (x_a, y_a) et $x_b, y_b)$ est donné par la formule :

$$d = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}$$

1. Dans l'éditeur Bashton, écrire la définition d'une fonction dont le nom est *distance_euclidienne*, avec 4 paramètres (les coordonnées de 2 points dans un repère orthonormé) : *x_a*, *y_a*, *x_b*, *y_b*. Cette fonction retourne la distance euclidienne enre les deux points du repère.
2. Dans un programme principal, appelez votre fonction pour qu'elle calcule la distance euclidienne entre les points $A(2, -1)$ et $B(5, 2)$. Vous afficherez le résultat et vous vérifierez que la valeur affichée est environ 4,24.
3. Dans ce même programme principal, appelez la fonction avec des paramètres de votre choix. Vérifiez le résultat affiché en calculant manuellement la distance euclidienne.