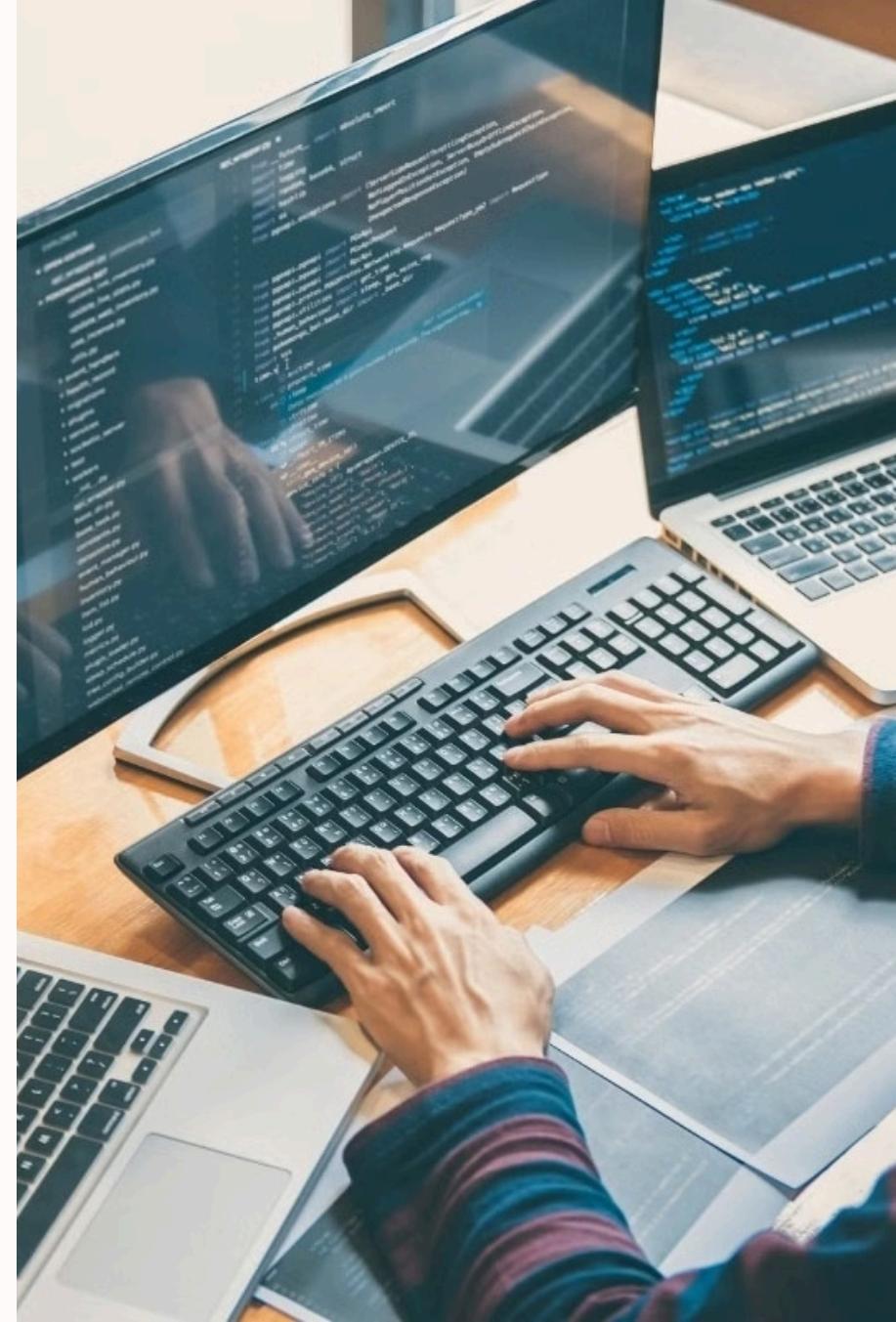


Les 3 niveaux fondamentaux pour le développement





Variables, Classes et Fonctions



Les variables en programmation

Une variable est un espace de stockage dans la mémoire d'un programme, utilisé pour conserver une valeur qui peut être modifiée au cours de l'exécution du programme. Chaque variable a un nom, un type (comme un nombre entier, une chaîne de caractères, etc.), et une valeur associée.

Exemple **En Python** :

```
# Variable de type entier (int)
age = 25

# Variable de type flottant (float)
taille = 1.75

# Variable de type chaîne de caractères (string)
nom = "Alice"

# Variable de type booléen (boolean)
est_majeur = True

# Variable de type liste (list)
notes = [15, 18, 20, 14]
```

Les Classes en Programmation Orientée Objet :

Une **classe** est un modèle ou un plan à partir duquel des objets peuvent être créés. Elle définit des **attributs** (variables) et des **méthodes** (fonctions) qui caractérisent l'objet. En Programmation Orientée Objet (POO), les classes permettent de structurer le code en représentant des entités du monde réel sous forme d'objets.

Exemple en **Python** :

```
# Définition d'une classe Voiture
class Voiture:
    def __init__(self, marque, annee):
        self.marque = marque # Attribut marque
        self.annee = annee # Attribut annee

# Instanciation d'un objet de la classe Voiture
ma_voiture = Voiture("Toyota", 2020)

# Accès aux attributs de l'objet
ma_voiture.marque # Toyota
ma_voiture.annee # 2020
```

Exemple en **PHP** :

```
<?php  
// Définition d'une classe Voiture  
class Voiture {  
    public $marque;  
    public $annee;  
  
    // Constructeur de la classe  
    public function __construct($marque, $annee) {  
        $this->marque = $marque;  
        $this->annee = $annee;  
    }  
}  
  
// Instanciation d'un objet de la classe Voiture  
$ma_voiture = new Voiture("Toyota", 2020);  
  
// Accès aux attributs de l'objet  
echo $ma_voiture->marque; // Toyota  
echo $ma_voiture->annee; // 2020  
?>
```

Les Fonctions en Programmation :

Une **fonction** est un bloc de code réutilisable qui effectue une tâche spécifique. Elle peut prendre des **paramètres** en entrée et retourner une **valeur** en sortie. Les fonctions permettent de rendre le code plus modulaire et plus facile à maintenir.

Exemple en **Python** :

```
# Définition d'une fonction qui calcule la somme de deux nombres
def addition(a, b):
    return a + b

# Utilisation de la fonction
resultat = addition(5, 3) # 8
```

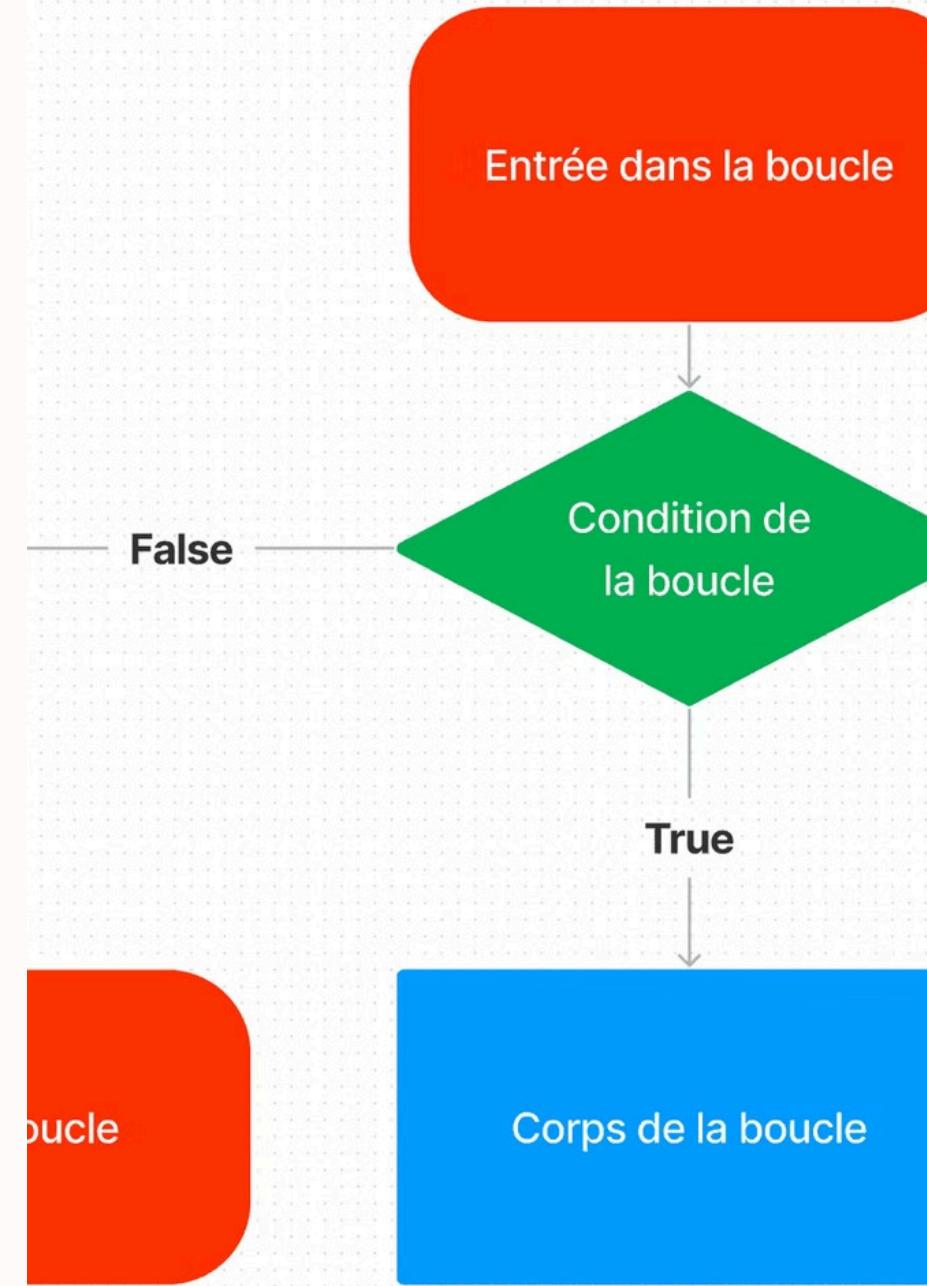
Exemple en **PHP** :

```
<?php
// Définition d'une fonction qui calcule la somme de deux nombres
function addition($a, $b) {
    return $a + $b;
}

// Utilisation de la fonction
$resultat = addition(5, 3); // 8
?>
```



Les Conditions et Boucles



Les Conditions en Programmation :

Les **conditions** permettent de contrôler le flux d'exécution d'un programme. Elles permettent d'exécuter différentes parties du code en fonction du résultat d'une expression booléenne (vrai ou faux). Les structures conditionnelles courantes incluent `if`, `else`, et `elif` (ou `elseif`).

Exemple en **Python** :

```
# Condition simple avec if, elif et else
age = 20

if age >= 18:
    statut = "Majeur"
elif age == 17:
    statut = "Presque majeur"
else:
    statut = "Mineur"
```

Exemple en **PHP** :

```
<?php
// Condition simple avec if, elseif et else
$age = 20;

if ($age >= 18) {
    $statut = "Majeur";
} elseif ($age == 17) {
    $statut = "Presque majeur";
} else {
    $statut = "Mineur";
}
?>
```

Les Boucles en Programmation :

Les **boucles** permettent de répéter l'exécution d'un bloc de code tant qu'une condition est vraie (comme avec `while`), ou pour parcourir des éléments d'une collection (comme avec `for` et `foreach`). Elles permettent d'automatiser des tâches répétitives.

Exemple en **Python** :

Boucle `for`

```
# Boucle for pour parcourir une plage de nombres
for i in range(1, 6):
    print(i)
```

Boucle `foreach`

```
# Boucle for pour parcourir une liste
nombres = [1, 2, 3, 4, 5]

for nombre in nombres:
    print(nombre)
```

Boucle `while`

```
# Boucle while qui continue tant que la condition est vraie
compteur = 0

while compteur < 5:
    print(compteur)
    compteur += 1
```

Exemple en **PHP** :

Boucle `for`

```
<?php
// Boucle for pour parcourir une plage de nombres
for ($i = 1; $i <= 5; $i++) {
    echo $i;
}
?>
```

Boucle `foreach`

```
<?php
// Boucle foreach pour parcourir un tableau
$nombres = [1, 2, 3, 4, 5];

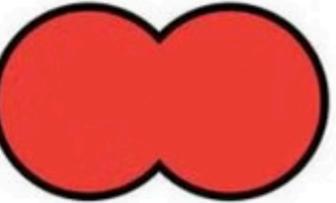
foreach ($nombres as $nombre) {
    echo $nombre;
}
?>
```

Boucle `while`

```
<?php
// Boucle while qui continue tant que la condition est vraie
$compteur = 0;

while ($compteur < 5) {
    echo $compteur;
    $compteur++;
}
?>
```

Les opérateurs booléens

Effet	Résult
= 	Croiser A et
= 	Associer A, B, A
= 	Exclude A sar

(A OR a) AND (B OR B OF



Les Opérateurs

Opérateurs arithmétiques:

- + (Addition): Ajoute deux valeurs.
- (Soustraction): Soustrait une valeur d'une autre.
- * (Multiplication): Multiplie deux valeurs.
- / (Division): Divise une valeur par une autre.
- % (Modulo): Renvoie le reste d'une division.
- ** (Exponentiation): Élève une valeur à la puissance d'une autre (similaire dans de nombreux langages).

Opérateurs d'affectation:

- = (Affectation): Affecte la valeur de droite à la variable de gauche.
- +=, -=, *=, /=, %=: Combinaison de l'opérateur arithmétique et de l'affectation (commun dans de nombreux langages).

Opérateurs de comparaison:

== (Égalité): Vérifie si deux valeurs sont égales (en Python, == est similaire, mais en JavaScript, == effectue une conversion de type).

=== (Identité): Vérifie si deux valeurs sont égales et de même type (spécifique à PHP et JavaScript).

!= (Non égalité): Vérifie si deux valeurs ne sont pas égales.

!== (Non identité): Vérifie si deux valeurs ne sont pas égales ou pas du même type.

> (Plus grand que), < (Moins que), >= (Plus grand ou égal à), <= (Moins ou égal à): Comparaison numérique standard (similaire dans la plupart des langages).

Opérateurs logiques:

&& ou and (Et logique): Vrai si les deux opérandes sont vrais.

|| ou or (Ou logique): Vrai si l'un des opérandes est vrai.

! (Non logique): Inverse la valeur de vérité (commun dans de nombreux langages).

Opérateur de concaténation:

. (Concaténation): Fusionne deux chaînes de caractères (en JavaScript, + est utilisé à la place pour la concaténation).

Opérateurs de contrôle d'erreur:

Opérateurs d'incrément et de décrément:

++, -- (Incrément et décrément): Augmente ou diminue une valeur de 1 (commun dans de nombreux langages).

Opérateur de coalescence nulle:

?? (Coalescence nulle): Renvoie l'opérande de gauche si celui-ci n'est ni null ni undefined, sinon renvoie l'opérande de droite (similaire dans de nombreux langages modernes).

Opérateur ternaire:

condition ? expr1 : expr2: Renvoie expr1 si la condition est vraie, sinon renvoie expr2 (commun dans de nombreux langages).

Classes et Fonctions en PHP

```
<?php

class Employe {

    private $nom;
    private $age;
    private $departement;

    public function __construct($nom, $age, $departement) {
        $this->nom = $nom;
        $this->age = $age;
        $this->departement = $departement;
    }

    public function afficherInfos() {
        echo "Nom: " . $this->nom . ", Age: " . $this->age . ", Département: " . $this->departement . "\n";
    }
}

class Departement {

    private $nom;
    private $employes = [];

    public function __construct($nom) {
        $this->nom = $nom;
    }

    public function ajouterEmploye($employe) {
        $this->employes[] = $employe;
    }

    public function afficherEmployes() {
        echo "Employés du département " . $this->nom . ":\n";
        foreach ($this->employes as $employe) {
            $employe->afficherInfos();
        }
    }
}

// Utilisation des classes
$departementIT = new Departement("Informatique");

$employe1 = new Employe("Alice", 30, "Informatique");
$employe2 = new Employe("Bob", 35, "Informatique");

$departementIT->ajouterEmploye($employe1);
$departementIT->ajouterEmploye($employe2);

$departementIT->afficherEmployes();

?>
```

Classes et Fonctions en C#

```
using System;

public class Voiture
{
    private string marque;
    private int annee;

    // Constructeur
    public Voiture(string marque, int annee)
    {
        this.marque = marque;
        this.annee = annee;
    }

    // Méthode pour afficher les informations de la voiture
    public void AfficherInfos()
    {
        Console.WriteLine("Marque: " + marque + ", Année: " + annee);
    }
}

class Program
{
    static void Main()
    {
        Voiture maVoiture = new Voiture("Toyota", 2020);
        maVoiture.AfficherInfos();
    }
}
```

Conditions et Boucles en C#

```
using System;

class Program
{
    static void Main()
    {
        // Exemple de condition
        int age = 20;
        if (age >= 18)
        {
            Console.WriteLine("Vous êtes majeur.");
        }
        else
        {
            Console.WriteLine("Vous êtes mineur.");
        }

        // Exemple de boucle
        for (int i = 0; i < 5; i++)
        {
            Console.WriteLine("Numéro : " + i);
        }
    }
}
```

Opérateurs en C#

```
using System;

class Program
{
    static void Main()
    {
        // Opérateurs mathématiques
        int a = 10, b = 5;
        Console.WriteLine("Addition : " + (a + b)); // 15
        Console.WriteLine("Soustraction : " + (a - b)); // 5

        // Opérateurs logiques
        bool estMajeur = true, aPermis = false;
        Console.WriteLine("Peut conduire : " + (estMajeur && aPermis)); // false

        // Opérateurs relationnels
        Console.WriteLine("Égal à : " + (a == b)); // false
        Console.WriteLine("Différent de : " + (a != b)); // true
    }
}
```

Exercice 1: Calculatrice Simple

Demandez à l'utilisateur de saisir deux nombres.

1. Demandez à l'utilisateur de choisir une opération (addition, soustraction, multiplication, division).
2. Calculez le résultat de l'opération choisie sur les deux nombres.
3. Affichez le résultat.

Exemple de Code :

```
<?php  
// Imaginons que les entrées viennent d'un formulaire  
$nombre1 = $_POST['nombre1'];  
$nombre2 = $_POST['nombre2'];  
$operation = $_POST['operation']; // 'addition', 'soustraction', etc.  
  
switch ($operation) {  
    case 'addition':  
        echo $nombre1 + $nombre2;  
        break;  
    case 'soustraction':  
        echo $nombre1 - $nombre2;  
        break;  
    // Ajoutez ici les cas pour multiplication et division  
}  
  
?>
```

Exercice 2 : Gestionnaire de Notes

Objectif : Écrire un script PHP qui calcule la moyenne d'une série de notes.

```
$notes = [15, 17, 20, 13, 19]; // Tableau de notes
```

1. Créez un tableau contenant plusieurs notes (par exemple, des notes sur 20).
2. Calculez la somme de toutes les notes.
3. Calculez la moyenne des notes.
4. Affichez la moyenne.

Exercice 2 : Gestionnaire de Notes

```
<?php  
$notes = [15, 17, 20, 13, 19]; // Tableau de notes  
$somme = 0;  
  
foreach ($notes as $note) {  
    $somme += $note;  
}  
  
$moyenne = $somme / count($notes);  
echo "La moyenne est : " . $moyenne;  
?>
```

Algorithmes en Python

Les fonctions à utiliser dans ces exercices en Python:

1. input()

Définition : La fonction `input()` permet de demander une entrée à l'utilisateur via le clavier et retourne cette entrée sous forme de chaîne de caractères.

Exemple d'utilisation :

```
nombre1 = input("Entrez un nombre : ")
```

2. int() / float()

Définition : La fonction `int()` convertit une chaîne de caractères en nombre entier, et `float()` la convertit en nombre à virgule flottante.

Exemple d'utilisation :

```
nombre1 = int(input("Entrez un nombre entier : "))  
nombre2 = float(input("Entrez un nombre à virgule : "))
```

3. print()

Définition : La fonction `print()` permet d'afficher un message ou une valeur sur la sortie standard (généralement l'écran).

Exemple d'utilisation :

```
print("Le résultat est :", resultat)
```

4. sum()

Définition : La fonction `sum()` renvoie la somme des éléments d'une séquence (comme une liste).

Exemple d'utilisation :

```
notes = [15, 17, 20]  
somme = sum(notes) # Calcule la somme des notes
```

5. len()

Définition : La fonction `len()` retourne la longueur d'une séquence (comme une liste, une chaîne de caractères, etc.).

Exemple d'utilisation :

```
nombre_notes = len(notes) # Retourne le nombre d'éléments dans la liste 'notes'
```

6. max()

Définition : La fonction `max()` retourne la plus grande valeur d'une séquence.

Exemple d'utilisation :

```
max_nombre = max([3, 58, 11, 21]) # Retourne 58
```

7. for (boucle)

Définition : Une boucle `for` permet d'itérer sur les éléments d'une séquence (comme une liste) et d'exécuter du code pour chaque élément.

Exemple d'utilisation :

```
for note in notes:  
    somme += note # Additionne chaque note à la somme
```

8. while (boucle)

Définition : La boucle `while` exécute un bloc de code tant qu'une condition est vraie.

Exemple d'utilisation :

```
compteur = 0  
while compteur < 5:  
    compteur += 1
```

9. split()

Définition : La méthode `split()` divise une chaîne de caractères en une liste de mots en utilisant un séparateur donné (par défaut, un espace).

Exemple d'utilisation :

```
phrase = "Le test logiciel est essentiel"  
mots = phrase.split(' ') # Divise la phrase en mots
```

10. return

Définition : La fonction `return` dans une fonction permet de renvoyer une valeur au lieu de la simplement afficher.

Exemple d'utilisation :

```
def addition(a, b):  
    return a + b
```

11. range()

Définition : La fonction `range()` génère une séquence de nombres. Elle est souvent utilisée dans les boucles `for`.

Exemple d'utilisation :

```
for i in range(1, 6):  
    print(i) # Affiche les nombres de 1 à 5
```

12. reversed()

Définition : La fonction `reversed()` renvoie une version inversée d'une séquence.

Exemple d'utilisation :

```
chaine_inversee = "join(reversed(chaine)) # Inverse la chaîne de caractères"
```

13. join()

Définition : La méthode `join()` est utilisée pour concaténer une séquence en une chaîne de caractères.

Exemple d'utilisation :

```
chaine_inversee = ".join(chaine_inversee) # Joint les caractères pour former une chaîne"
```

Ces fonctions sont essentielles pour réaliser les exercices que tu as mentionnés, et elles couvrent des aspects comme l'entrée utilisateur, la manipulation des données, les boucles et les conditions en Python.

Exercice 1: Inversion de Chaîne

Écrire un programme qui inverse une chaîne de caractères.

- Exemple :
 - Entrée : "Bonjour"
 - Sortie : "ruojnoB"

Exercice 1: Inversion de Chaîne

```
def inverser_chaine(chaine):
    chaine_inversee = ""
    for i in range(len(chaine) - 1, -1, -1):
        chaine_inversee += chaine[i]
    return chaine_inversee

print(inverser_chaine("Bonjour")) # Affiche "ruojnoB"
```

Exercice 2 : Somme des Éléments d'une Liste

Créer un algorithme qui calcule la somme des éléments d'une liste.

- Exemple :
 - Entrée : [1, 2, 3, 4]
 - Sortie : 10

Exercice 2 : Somme des Éléments d'une Liste

```
def somme_liste(liste):
    somme = 0
    for element in liste:
        somme += element
    return somme

print(somme_liste([1, 2, 3, 4])) # Affiche 10
```

Exercice 3 : Recherche du Plus Grand Nombre

Écrire un script qui trouve le plus grand nombre dans une liste.

- Exemple :
 - Entrée : [3, 58, 11, 21]
 - Sortie : 58

Exercice 3 : Recherche du Plus Grand Nombre

```
def plus_grand_nombre(liste):
    max_nombre = liste[0]
    for nombre in liste:
        if nombre > max_nombre:
            max_nombre = nombre
    return max_nombre

print(plus_grand_nombre([3, 58, 11, 21])) # Affiche 58
```

Exercice 4 : Calcul de la Factorielle

Développer une fonction qui calcule la factorielle d'un nombre.

- Exemple :
 - Entrée : 5
 - Sortie : 120 (car $5! = 5 \times 4 \times 3 \times 2 \times 1$)

Exercice 4 : Calcul de la Factorielle

```
def factorielle(n):
    resultat = 1
    for i in range(1, n + 1):
        resultat *= i
    return resultat

print(factorielle(5)) # Affiche 120
```

Exercice 5 : Vérification de Palindrome

Créer un programme qui vérifie si un mot est un palindrome (se lit de la même manière dans les deux sens).

- Exemple :
 - Entrée : "radar"
 - Sortie : True

Exercice 5 : Vérification de Palindrome

```
def est_palindrome(mot):
    longueur = len(mot)
    for i in range(longueur // 2):
        if mot[i] != mot[longueur - i - 1]:
            return False
    return True

print(est_palindrome("radar")) # Affiche True
```

Exercice 6 : Compteur de Mots

Écrire un script qui compte le nombre de mots dans une phrase.

- Exemple :
 - Entrée : "Le test logiciel est essentiel"
 - Sortie : 5

Exercice 6 : Compteur de Mots

```
def compter_mots(phrase):
    mots = phrase.split(' ')
    return len(mots)

print(compter_mots("Le test logiciel est essentiel")) # Affiche 4
```