1. Présentation de Python / Overview of Python

1.1 Introduction / Introduction

Python est un langage de programmation interprété, créé à la fin des années 1980 par Guido van Rossum.
Python is an interpreted programming language created in the late 1980s by Guido van Rossum.

Il est connu pour sa simplicité et sa lisibilité, ce qui en fait un excellent choix pour les débutants.
It is known for its simplicity and readability, making it an excellent choice for beginners.

1.2 Caractéristiques principales / Main Features

- **Langage interprété / Interpreted language** : le code est exécuté ligne par ligne, sans compilation préalable.
- **Langage multiplateforme / Cross-platform** : Python fonctionne sur Windows, macOS et Linux.
- **Typage dynamique / Dynamic typing** : il n'est pas nécessaire de déclarer le type des variables.
- **Large bibliothèque standard / Large standard library** : Python dispose de nombreux modules intégrés.
- **Lisibilité du code / Code readability** : la syntaxe de Python est claire et concise.

1.3 Domaines d'application / Areas of Application

Python est utilisé dans de nombreux domaines :
Python is used in many fields:

- **Développement web / Web development** : frameworks tels que Django, Flask.
- **Intelligence artificielle / Artificial Intelligence** : bibliothèques comme TensorFlow, scikit-learn.
- **Analyse de données / Data analysis** : Pandas, NumPy, Matplotlib.
- **Automatisation / Automation** : scripts pour automatiser les tâches répétitives.
- **Éducation / Education** : apprentissage de la programmation pour débutants.

1.4 Avantages de Python / Advantages of Python

1. Facile à apprendre et à utiliser. / Easy to learn and use.
2. Grande communauté d'utilisateurs. / Large and active community.
3. Compatible avec d'autres langages (C, Java). / Compatible with other languages (C, Java).
4. Large bibliothèque de modules. / Large library of modules.

1.5 Premier programme Python / First Python Program

Exemple / Example :

```
# Ceci est un commentaire en Python / This is a comment in Python
# Le programme affiche "Hello, World!"
print("Hello, World!")
```

Explication / Explanation :

- `#` introduit un commentaire qui n'est pas exécuté.
- `print()` est une fonction intégrée qui affiche du texte à l'écran.
- Les guillemets (" ") délimitent une chaîne de caractères.

2. Installation et préparation d'un environnement de développement

2.1 Installation de Python / Installing Python

Python peut être téléchargé depuis le site officiel : https://www.python.org/downloads.
Choisissez la version la plus récente (par exemple Python 3.x) et suivez les étapes d'installation.
Python can be downloaded from the official website. Choose the latest version (e.g. Python 3.x) and follow the installation steps.

☑ **Vérification / Verification :**
Ouvrez un terminal ou l'invite de commandes et tapez :

```
python --version
```

Cela affiche la version installée de Python.
This command displays the installed version of Python.

2.2 Environnement de développement / Development Environment

Pour écrire et exécuter des programmes Python, plusieurs outils peuvent être utilisés :

Outil / Tool	Description
IDLE	Environnement livré avec Python. Simple et rapide pour débuter.
VS Code	Éditeur moderne avec extensions Python.
PyCharm	IDE complet pour projets professionnels.

Outil / Tool	Description
Jupyter Notebook	Idéal pour les démonstrations et l'enseignement.

3. Syntaxe de base de Python / Basic Python Syntax

Python se distingue par sa **syntaxe claire et lisible**.
Python stands out for its **clear and readable syntax**.

3.1 Indentation

En Python, **l'indentation (espaces en début de ligne)** remplace les accolades `{}`.
Indentation is essential — it defines code blocks.

Exemple / Example :

```
if True:
    print("Indented block executed")
```

⚠ Une erreur d'indentation entraîne une erreur d'exécution.

3.2 Commentaires / Comments

Les commentaires commencent par le symbole `#`.
Comments start with `#`.

```
# Ceci est un commentaire / This is a comment
print("Bonjour Python!")  # Commentaire sur la même ligne
```

4. Variables et constantes / Variables and Constants

Une **variable** est une zone mémoire qui stocke une valeur.
A variable is a memory location that stores a value.

4.1 Création de variables / Creating Variables

Aucune déclaration de type n'est nécessaire en Python.
No need to declare the type explicitly.

```
nom = "Ali"
age = 20
pi = 3.14
```

4.2 Constantes / Constants

Python ne dispose pas de mot-clé `const`. Par convention, on écrit les constantes en majuscules :

```
PI = 3.14159
TAUX_TVA = 0.19
```

5. Types de données fondamentaux / Fundamental Data Types

Type	Exemple	Description
int	10	Entier / Integer
float	3.14	Réel / Floating-point number
bool	True, False	Booléen / Boolean value
str	"Bonjour"	Chaîne de caractères / String

Exemple :

```
a = 5           # int
b = 2.5         # float
c = True        # bool
d = "Python"    # str
```

6. Opérateurs / Operators

6.1 Arithmétiques / Arithmetic

Opérateur	Signification	Exemple
`+`	Addition	`a + b`
`-`	Soustraction	`a - b`
`*`	Multiplication	`a * b`
`/`	Division	`a / b`
`//`	Division entière	`a // b`
`%`	Modulo (reste)	`a % b`

Opérateur	Signification	Exemple
`**`	Puissance	`a ** b`

6.2 Comparaison / Comparison

Opérateur	Signification	Exemple
`==`	Égal à / Equal to	`a == b`
`!=`	Différent de / Not equal to	`a != b`
`>`	Supérieur à / Greater than	`a > b`
`<`	Inférieur à / Less than	`a < b`
`>=`	Supérieur ou égal / Greater or equal	`a >= b`
`<=`	Inférieur ou égal / Less or equal	`a <= b`

6.3 Logiques / Logical

Opérateur	Signification	Exemple
`and`	Et logique / Logical AND	`a > 0 and b < 10`
`or`	Ou logique / Logical OR	`a > 0 or b < 0`
`not`	Négation / Negation	`not a == b`

7. Instructions d'entrée et de sortie / Input and Output

7.1 Sortie / Output

La fonction `print()` affiche des données à l'écran.
The `print()` function displays data.

```
print("Bonjour le monde!")   # Hello world!
```

On peut combiner plusieurs éléments :

```
nom = "Ali"
age = 20
print("Nom:", nom, "- Âge:", age)
```

7.2 Entrée / Input

La fonction `input()` permet de lire des données saisies par l'utilisateur.
The `input()` function reads user input.

```
nom = input("Entrez votre nom : ")
print("Bonjour", nom)
```

⚠ Les données lues avec `input()` sont **de type chaîne de caractères (str)**.
Pour les nombres, il faut les convertir :

```
age = int(input("Entrez votre âge : "))
```

7.3 Exercice / Exercise

Énoncé / Statement
Écrivez un programme qui demande le nom et l'âge de l'utilisateur, puis affiche :

Bonjour <nom>, vous avez <âge> ans.
Write a program that asks the user for their name and age, then displays:
Hello <name>, you are <age> years old.

Correction / Solution

```
# Lecture des données utilisateur / Read user input
name = input("Entrez votre nom / Enter your name: ")
age = input("Entrez votre âge / Enter your age: ")

# Affichage du résultat / Display the result
print("Bonjour", name, ", vous avez", age, "ans.")
print("Hello", name, ", you are", age, "years old.")
```

8. Affectation et expressions / Assignment and Expressions

L'opérateur = affecte une valeur à une variable.
The = operator assigns a value to a variable.

```
x = 5
y = x + 3
z = y * 2
```

Affectations multiples / Multiple Assignments

```
a, b, c = 1, 2, 3
```

Affectation augmentée / Augmented Assignment

```
x = 5
x += 2   # équivaut à x = x + 2
x *= 3   # équivaut à x = x * 3
```

Exercice pratique / Practical Exercise

Énoncé / Task

Écrire un programme qui :

1. Demande deux nombres à l'utilisateur.
2. Calcule et affiche leur somme, leur produit et leur quotient.

Correction / Solution

```
a = float(input("Entrez le premier nombre : "))
b = float(input("Entrez le deuxième nombre : "))

print("Somme =", a + b)
print("Produit =", a * b)
print("Quotient =", a / b)
```