



# ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR

Membre de **HONORIS UNITED UNIVERSITIES**

## Introduction à Python - Les Bases

Dr. Hamza Abouabid  
H.abouabid@emsi.ma

2025-2026

# Objectifs du Cours

- **Objectifs Généraux :**

- Comprendre les concepts de base de la programmation en Python.
- Apprendre à écrire et exécuter des scripts Python simples.
- Se familiariser avec les structures de données et les types de données en Python.
- Développer des compétences en résolution de problèmes à l'aide de Python.

- **Objectifs Spécifiques :**

- Avantages de Python.
- Différences Python 2 vs 3.
- Utilisation de l'interpréteur Python.
- Avantages de l'interpréteur IPython.

- **Compétences à Acquérir :**

- Syntaxe de base et structures de contrôle.
- Utilisation des listes, tuples, dictionnaires.
- Manipulation de fichiers et données (JSON, CSV, Pandas).
- Organisation du code en modules et packages.

# Partie 1 : Introduction à Python

## Aperçu du cours

- Objectifs du cours et attentes.
- Brève présentation de l'importance de Python.
- Différences entre Python 2 et Python 3.
- Installation de Python.
- Votre premier programme Python.

# Qu'est-ce que Python ?

## ● Introduction à Python :

- Python est un *langage de programmation interprété*, ce qui signifie que les instructions sont exécutées directement sans compilation préalable. Cette caractéristique le rend particulièrement adapté pour le prototypage rapide et les itérations de développement.
- En tant que *langage de haut niveau*, Python permet une abstraction significative par rapport au langage machine, rendant le code plus lisible, compréhensible et maintenable.

# Qu'est-ce que Python ?

## • Caractéristiques Principales de Python :

- *Facilité d'apprentissage* : Sa syntaxe claire et sa communauté active en font un choix excellent pour les débutants en programmation.
- *Interprétable* : Python est exécuté dans un environnement d'exécution, ce qui facilite le test et le débogage de programmes complexes.
- *Polyvalence* : Utilisé dans une vaste gamme d'applications, de l'automatisation de scripts simples au développement de systèmes complexes, Python est extrêmement adaptable.
- *Applications diverses* : Python est omniprésent dans le développement web (via des frameworks comme Django et Flask), l'analyse de données (avec des outils tels que Pandas et NumPy), l'intelligence artificielle (grâce à des bibliothèques comme TensorFlow et PyTorch), et dans le calcul scientifique.

# Qu'est-ce que Python ?

- **Python dans la Science des Données :**

- Python est largement utilisé dans la science des données pour des tâches telles que le nettoyage de données, l'analyse statistique, la visualisation de données, et le machine learning.
- Des bibliothèques comme Pandas pour la manipulation de données, Matplotlib et Seaborn pour la visualisation, et Scikit-learn pour le machine learning, rendent Python particulièrement puissant pour l'analyse de données.
- Sa capacité à intégrer avec d'autres langages et outils, comme SQL pour les bases de données, et son support pour le calcul parallèle et distribué, le rendent idéal pour travailler avec de grandes ensembles de données.

# Différences entre Python 2 et Python 3

## print : Instruction vs Fonction

- **Python 2** : print est une instruction, il ne nécessite pas de parenthèses.
  - Exemple : `print "Bonjour"`
- **Python 3** : print est une fonction, les parenthèses sont obligatoires.
  - Exemple : `print("Bonjour")`

# Différences entre Python 2 et Python 3

## Chaînes de caractères : Unicode vs ASCII

- **Python 2** : Par défaut, les chaînes de caractères (`str`) sont des séquences d'octets ASCII.
- **Python 3** : Les chaînes de caractères (`str`) sont Unicode par défaut, facilitant la gestion des caractères internationaux.
- Pour obtenir des chaînes Unicode en Python 2, il fallait utiliser le type `unicode`.

# Différences entre Python 2 et Python 3

## Division des Entiers

- **Python 2** : La division entre deux entiers (`5 / 2`) retourne un entier (2).
- **Python 3** : La division entre deux entiers (`5 / 2`) retourne un nombre flottant (2.5).
- Pour obtenir une division entière en Python 3, utilisez l'opérateur `//` :
  - Exemple : `5 // 2` donne 2.

# Différences entre Python 2 et Python 3

## Fonction `range()`

- **Python 2 :** `range()` retourne une liste. Utilisez `xrange()` pour obtenir un générateur.
- **Python 3 :** `range()` retourne un objet de type générateur, ce qui est plus efficace en termes de mémoire.

# Différences entre Python 2 et Python 3

## Gestion des Exceptions

- **Python 2** : La syntaxe pour lever une exception est : `except ValueError, e`
- **Python 3** : La syntaxe a changé pour : `except ValueError as e`

# Différences entre Python 2 et Python 3

## input() vs raw\_input()

- **Python 2 :**

- `input()` évalue l'entrée comme du code Python.
- `raw_input()` est utilisé pour lire une chaîne de caractères en entrée.

- **Python 3 :**

- `input()` est utilisé pour lire des chaînes de caractères (comportement de `raw_input()`).

# Différences entre Python 2 et Python 3

## Métaclasses

- **Python 2 :** Les métaclasses sont définies avec la syntaxe :  
`__metaclass__ = MetaClassName`
- **Python 3 :** Utilisez la syntaxe dans la définition de la classe : `class MyClass(metaclass=MetaClassName)`

# Différences entre Python 2 et Python 3

## Bibliothèques Standard Modernisées

- Certaines bibliothèques ont été renommées ou modifiées.
- **Exemple :**
  - ConfigParser (Python 2) est devenu configparser en Python 3.
  - cPickle a été fusionné avec pickle.

# Différences entre Python 2 et Python 3

## Support et Maintenance

- **Python 2** : La dernière version était Python 2.7, et le support officiel a pris fin en janvier 2020.
- **Python 3** : Python 3 est activement maintenu et mis à jour, avec de nouvelles fonctionnalités et améliorations.

# Différences entre Python 2 et Python 3

## Résumé des Différences

- Python 2 est obsolète et ne reçoit plus de support.
- Python 3 est le présent et l'avenir du langage, avec de meilleures performances, une meilleure gestion de la mémoire, et un support accru des technologies modernes.

# Installation de Python

- **Guide d'Installation étape par étape :**

- *Téléchargement* : Accédez au site officiel de Python, [python.org](https://python.org), et téléchargez la dernière version stable pour votre système d'exploitation.
- *Installation* : Lancez l'installateur téléchargé. Sur Windows, assurez-vous de cocher l'option "Add Python to PATH" avant de commencer l'installation.
- *Configuration Post-Installation* : Il peut être nécessaire de configurer certaines variables d'environnement, surtout sous Linux et macOS, pour faciliter l'accès aux commandes Python et Pip (gestionnaire de paquets Python) depuis le terminal.

- **Vérification de l'Installation de Python :**

- Après l'installation, ouvrez un terminal ou une invite de commande et tapez 'python –version' (ou 'python3 –version' sur certains systèmes Linux/macOS). Si Python est correctement installé, cette commande affichera la version installée.
- Il est également recommandé de vérifier l'installation de Pip, le gestionnaire de paquets de Python, en exécutant 'pip –version'.

# Installation de Python

- **Installation de Python sur Différentes Plateformes :**
  - *Windows* : L'installateur inclut l'option d'ajouter Python au PATH, facilitant l'accès depuis l'invite de commande.
  - *macOS* : Python peut être installé via l'installateur téléchargé ou en utilisant des gestionnaires de paquets comme Homebrew.
  - *Linux* : Python est souvent pré-installé sur de nombreuses distributions Linux. Des versions spécifiques peuvent être installées via le gestionnaire de paquets de la distribution, comme apt pour Ubuntu ou yum pour Fedora.

# Votre premier programme Python

```
print("Bonjour le monde!")
```

- **Structure du Programme :**

- Ce programme est un exemple classique de "Hello World" dans le monde de la programmation.
- En une seule ligne de code, il démontre la capacité de Python à exécuter une tâche simple : afficher un message à l'utilisateur.

- **Explication de la Syntaxe :**

- *La fonction print* : 'print()' est une fonction intégrée en Python, utilisée pour afficher le texte ou la valeur de variable que l'utilisateur souhaite voir.
- *Guillemets pour les chaînes de caractères* : Les guillemets (" ") sont utilisés pour délimiter des chaînes de caractères en Python.
- *Parenthèses* : Les parenthèses après 'print' indiquent qu'il s'agit d'une fonction.
- *Importance de la simplicité* : Ce programme illustre également la nature concise et lisible de Python, qui rend le langage accessible aux débutants tout en étant puissant pour les développeurs avancés.



# Exercice 1

## Tâche

écrire un programme pour imprimer votre nom et la date du jour.

# Correction de l'Exercice 1

```
import datetime

# Afficher le nom
print("Nom: Hamza Abouabid")

# Afficher la date du jour
print("Date: ", datetime.date.today())
```

# Partie 2

## Les Bases de Python :

- Variables et Types de Données
- Opérateurs en Python
- Structures de Contrôle : If-else
- Structures de Contrôle : Boucles

# Variables et Types de Données

- **Définition des variables**

- Une variable est un conteneur qui stocke des données.
- Elle permet de référencer et manipuler ces données dans le programme.

- **Types de données courants**

- `int` : Entiers (ex. : 5, -3, 42)
- `float` : Nombres à virgule flottante (ex. : 3.14, -0.001, 2.0)
- `string` : Chaînes de caractères (ex. : "Bonjour", 'A', "123")
- `boolean` : Valeurs booléennes (True ou False)

- Exemple : Déclaration et utilisation de différentes variables en Python

- Déclaration de variables

- age = 25 # Variable entière
    - temperature = 23.5 # Variable flottante
    - nom = "Alice" # Variable chaîne de caractères
    - est\_etudiant = True # Variable booléenne

- Utilisation des variables

- Calculer l'année de naissance

```
annee_actuelle = 2024
annee_naissance = annee_actuelle - age
print("annee de naissance :",
      annee_naissance)
```

- Afficher un message personnalisé

```
print("Bonjour, je m'appelle", nom, "et j' 
      ai", age, "ans.")
```

# Exemple de Code Complet

## Script Python Exemple

```
# Declaration des variables
age = 25
temperature = 23.5
nom = "Alice"
est_etudiant = True

# Calcul de l'annee de naissance
annee_actuelle = 2024
annee_naissance = annee_actuelle - age
print("Annee de naissance :", annee_naissance)

# Affichage d'un message personnalisé
print("Bonjour, je m'appelle", nom, "et j'ai", age, "ans.")
```

# Exercice 2

## Tâche

Créer des variables de différents types et les imprimer.

## Consignes

- ① Déclarez une variable entière nommée `nombre`.
- ② Déclarez une variable flottante nommée `prix`.
- ③ Déclarez une variable chaîne de caractères nommée `produit`.
- ④ Déclarez une variable booléenne nommée `en_stock`.
- ⑤ Assignez des valeurs appropriées à chaque variable.
- ⑥ Imprimez chacune des variables avec un message explicatif.

## Exemple de Résultat Attendu

```
nombre = 10
prix = 19.99
produit = "Livre"
en_stock = True

print("Nombre d'articles : ", nombre)
print("Prix unitaire : ", prix, "DHs")
print("Produit : ", produit)
print("En stock : ", en_stock)
```

# Fonctions type et id en Python

- **Fonction type()**

- Retourne le type d'une variable.
- Utile pour vérifier le type de données stocké dans une variable.

- **Fonction id()**

- Retourne l'identifiant unique de l'objet en mémoire.
- Permet de vérifier si deux variables pointent vers le même objet.

# Exemples avec type() et id()

## Utilisation de type()

```
print(type(age))          # <class 'int'>
print(type(temperature))  # <class 'float'>
print(type(nom))          # <class 'str'>
print(type(est_etudiant)) # <class 'bool'>
```

## Utilisation de id()

```
print(id(age))           # Exemple : 140705302930848
print(id(temperature))   # Exemple : 140705302930944
print(id(nom))           # Exemple : 140705302930992
print(id(est_etudiant))  # Exemple : 140705302931040
```

# Exemple de Code avec type() et id()

## Script Python Exemple

```
# Utilisation de type()
print("Type de age :", type(age))
print("Type de temperature :", type(temperature))
print("Type de nom :", type(nom))
print("Type de est_etudiant :", type(est_etudiant))

# Utilisation de id()
print("ID de age :", id(age))
print("ID de temperature :", id(temperature))
print("ID de nom :", id(nom))
print("ID de est_etudiant :", id(est_etudiant))
```

## Résultat Attendu

```
Type de age : <class 'int'>
Type de temperature : <class 'float'>
Type de nom : <class 'str'>
Type de est_etudiant : <class 'bool'>
ID de age : 140705302930848
ID de temperature : 140705302930944
ID de nom : 140705302930992
ID de est_etudiant : 140705302931040
```

# Nommage des Variables en Python

## • Règles de Base

- Les noms de variables doivent commencer par une lettre ou un underscore (\_).
- Ils ne peuvent pas commencer par un chiffre.
- Les noms peuvent contenir des lettres, des chiffres et des underscores.
- Les noms sont sensibles à la casse (age et Age sont différents).

## • Bonnes Pratiques

- Utiliser des noms significatifs et descriptifs.
- Suivre la convention `snake_case` pour la lisibilité.
- éviter les abréviations obscures.
- Ne pas utiliser de mots réservés par Python (comme `class`, `def`, etc.).

## • Mauvais Exemples

- `a`, `b1`, `temp`
- `NombreArticles`, `PrixUnit`

## • Bons Exemples

- `nombre_articles`, `prix_unitaire`
- `annee_naissance`, `est_etudiant`

# Exemples de Nommage des Variables

## Mauvais Nommage

```
a = 25
b1 = "Alice"
temp = 23.5
NombreArticles = 10
PrixUnit = 19.99
```

## Bon Nommage

```
age = 25
nom_etudiant = "Alice"
temperature = 23.5
nombre_articles = 10
prix_unitaire = 19.99
```

# Conseils Supplémentaires pour le Nommage

- **Utiliser des Noms en Anglais**

- Favorisez l'anglais pour une meilleure compatibilité internationale.
- Exemple : `is_student` au lieu de `est_etudiant`.

- **éviter les Noms Trop Longs ou Trop Courts**

- Trouvez un équilibre entre descriptivité et concision.
- Exemple : `date_de_naissance` peut être trop long, préférez `dob` (date of birth) si le contexte le permet.

- **Utiliser des Préfixes ou Suffixes Pertinents**

- Exemple : `is_active` pour une variable booléenne indiquant l'état actif.

- **éviter les Noms Répétitifs ou Ambigus**

- Choisissez des noms qui reflètent clairement le contenu ou le rôle de la variable.

# Opérateurs en Python

## Introduction aux Opérateurs en Python

- Les opérateurs sont des symboles ou des mots réservés utilisés pour effectuer des opérations sur des valeurs ou des variables.
- Ils permettent de manipuler les données et de construire des expressions.
- Python propose plusieurs catégories d'opérateurs :
  - Opérateurs arithmétiques
  - Opérateurs de comparaison
  - Opérateurs logiques
  - Opérateurs d'assignation
  - Opérateurs spéciaux

# Opérateurs Arithmétiques

Opérateur	Description
+	Addition
-	Soustraction
*	Multiplication
/	Division
%	Modulo (reste de la division)
**	Exponentiation
//	Division entière

Table – Opérateurs arithmétiques en Python

## Exemples

```
a = 10
```

```
b = 3
```

```
print(a + b)    # 13
```

```
print(a - b)    # 7
```

```
print(a * b)    # 30
```

```
print(a / b)    # 3.333...
```

```
print(a % b)    # 1
```

```
print(a ** b)   # 1000
```

```
print(a // b)   # 3
```

# Opérateurs de Comparaison

Opérateur	Description
<code>==</code>	Égal à
<code>!=</code>	Différent de
<code>&gt;</code>	Supérieur à
<code>&lt;</code>	Inférieur à
<code>&gt;=</code>	Supérieur ou égal à
<code>&lt;=</code>	Inférieur ou égal à

Table – Opérateurs de comparaison en Python

## Exemples

```
a = 5  
b = 10
```

```
print(a == b)    # False  
print(a != b)    # True  
print(a > b)     # False  
print(a < b)     # True  
print(a >= 5)    # True  
print(b <= 10)   # True
```

# Opérateurs Logiques

Opérateur	Description
and	ET logique
or	OU logique
not	NON logique

Table – Opérateurs logiques en Python

## Exemples

```
a = True  
b = False
```

```
print(a and b) # False  
print(a or b) # True  
print(not a) # False  
print(not b) # True
```

# Opérateurs d'Assignment

Opérateur	Description
=	Attribution
+=	Addition et attribution
-=	Soustraction et attribution
*=	Multiplication et attribution
/=	Division et attribution
%=	Modulo et attribution
**=	Exponentiation et attribution
//=	Division entière et attribution

Table – Opérateurs d'attribution en Python

## Exemples

```
a = 5  
a += 3 # a = a + 3 => 8  
a -= 2 # a = a - 2 => 6  
a *= 4 # a = a * 4 => 24  
a /= 3 # a = a / 3 => 8.0  
a %= 5 # a = a % 5 => 3.0  
a **= 2 # a = a ** 2 => 9.0  
a // = 2 # a = a // 2 => 4.0
```

# Opérateurs Spéciaux

Opérateur	Description
is	Vérifie si deux variables référencent le même objet
is not	Vérifie si deux variables ne référencent pas le même objet
in	Vérifie si une valeur est présente dans une séquence
not in	Vérifie si une valeur n'est pas présente dans une séquence

Table – Opérateurs spéciaux en Python

## Exemples

```
a = 25
b = 25
c = "Alice"
d = "Alice"

print(a is b)          # True (mêmes objets entiers immuables)
print(c is d)          # True (mêmes objets string immuables)
print(30 in [10, 20, 30]) # True
print("Bob" not in "Alice") # True
```

# Exercice 3

## Tâche

écrire un programme pour calculer la surface d'un rectangle.