Python

Notions basiques .

SAGIM 2024/2025

CES COURS SONT DESTINEES AUX ETUDIANTS
DE L'ECOLE SAGIM OPTION TSDI

Introduction à Python

Python est un langage de programmation haut niveau, interpr□t□ et polyvalent, c□l□bre pour sa simplicit□, sa lisibilit□ et sa flexibilit□.

Cr□□ par Guido van Rossum en 1991, Python met l'accent sur la lisibilité du code grâce à une syntaxe épurée et à l'utilisation de l'indentation.

SAGIM

Introduction à Python

PYTHON est largement utilisé dans des domaines variés :

- Développement Web (Django, Flask)
- Data Science & Machine Learning (Pandas, TensorFlow, scikit-learn)
- Automatisation/Scripting
- Intelligence Artificielle (IA)
- Calcul scientifique
- Développement de jeux (Pygame)

SAGIM

Caractéristiques Clés

Syntaxe intuitive : Proche de l'anglais, idéale pour les d□butants.
Multiplateforme : Fonctionne sur Windows, macOS, Linux
Open Source : Gratuit et personnalisable.
Biblioth□que standard □tendue : Modules prêts à l'emploi (fichiers, requ□tes HTTP, etc.).
Typage dynamique : Aucune d□claration explicite des types de variables.
□cosyst□me riche : Des milliers de packages via **PyPI** (Python Package Index).

SAGIM

Pourquoi Python ?

Accessible : Parfait pour apprendre la programmation.

D□veloppement rapide : Moins de lignes de code pour des r□sultats efficaces.

Flexible : Compatible avec les paradigmes proc□dural, orient□ objet et fonctionnel.

SAGIM

Pour Commencer

Installez Python : T□l□chargez-le depuis le site officiel.

□crivez du code : Utilisez un IDE (VS Code, PyCharm) ou Jupyter Notebook.

Explorez les biblioth□ques : Installez des packages avec : pip install nom-du-package.

SAGIM

Premier programme en Python

Ouvrez un éditeur de texte ou un IDE (comme PyCharm, VS Code…) et tapez le code suivant :

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

Enregistrez le fichier avec l'extension .py (par exemple, bonjour.py) et exécutez-le. Vous devriez voir le texte "Bonjour, monde!" s'afficher dans la console.

SAGIM

Variables et Types de Données

Qu'est-ce qu'une variable ?

Une variable est un espace en m□moire o□ tu peux stocker une valeur.
Chaque variable a un nom (identifiant) et une valeur.

Caract□ristiques des variables en Python :

- **Pas besoin de d□clarer le type :**
  Python devine automatiquement le type de la variable en fonction de la valeur que tu lui attribues.
- **variables sont r□utilisables :**
  Tu peux changer la valeur d'une variable □ tout moment.
- **Les noms des variables :**
  Doivent commencer par une lettre ou un underscore (_).
  Peuvent contenir des lettres, des chiffres et des underscores, mais pas commencer par un chiffre.

SAGIM

Variables et Types de Données

En Python, vous n'avez pas besoin de déclarer le type d'une variable. Le type est inféré automatiquement.

```
# Déclaration de variables
nombre = 10          # Entier (int)
pi = 3.14            # Flottant (float)
texte = "Python"     # Chaîne de caractères (str)
est_vrai = True      # Booléen (bool)

# Affichage des variables
print(nombre)
print(pi)
print(texte)
print(est_vrai)
```

SAGIM

Variables et Types de Données

Exemple (2) :

```
age = 25
nom = "Alice"
est_adulte = True
print(f"Nom : {nom}, âge : {age}, Adulte : {est_adulte}")
```

Explication :

```
print(f"Nom : {nom}, âge : {age}, Adulte : {est_adulte}")
```

f"" : Indique que c'est une f-string (ou formatted string literal)
{nom} : Insère la valeur de la variable nom dans la chaîne.
{age} : Insère la valeur de la variable age.
{est_adulte} : Insère la valeur de la variable est_adulte.

SAGIM

Variables et Types de Données

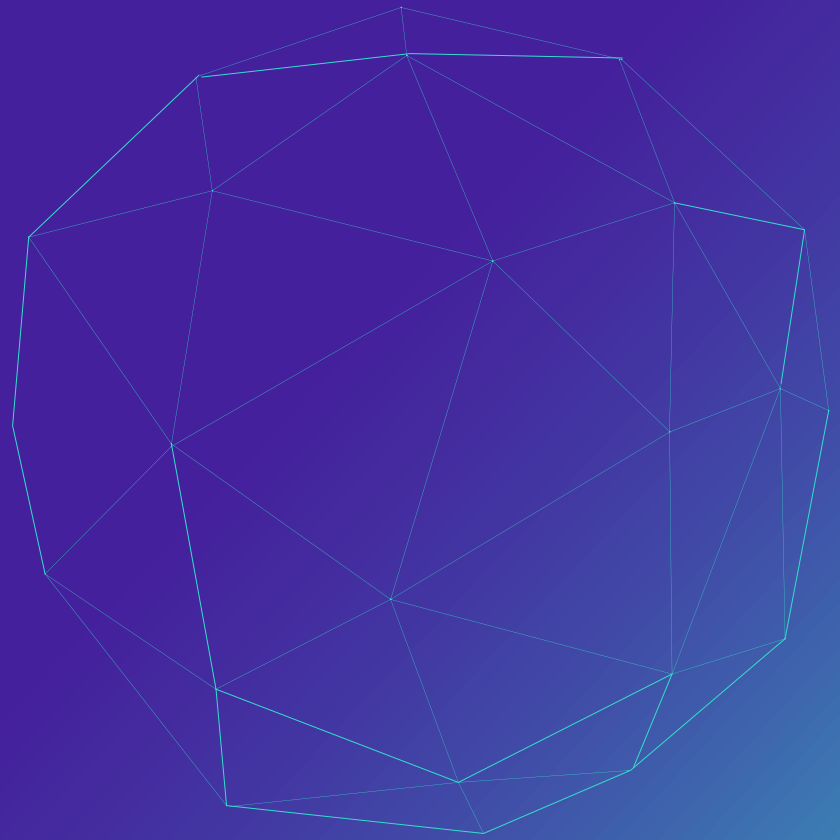
Type	Exemple	Description
`int`	`x = 10`	Nombre entier
`float`	`y = 3.14`	Nombre à virgule flottante
`str`	`nom = "Alice"`	Chaîne de caractères
`bool`	`vrai = True`	Valeur booléenne (`True` ou `False`)
`list`	`l = [1, 2, 3]`	Liste de valeurs
`tuple`	`t = (1, 2, 3)`	Tuple (immuable)
`dict`	`d = {"nom": "Alice", "age": 25}`	Dictionnaire (clé-valeur)

SAGIM

PARTIE 2 :

Lecture & Ecriture

Introdution au langage PYTHON



Lecture & Ecriture

la lecture (input) et l'écriture (output) sont des opérations fondamentales pour interagir avec l'utilisateur ou manipuler des données.

SAGIM

Lecture & Ecriture

1. L'□criture (Output)

L'□criture en Python se fait principalement avec la fonction print().
Elle permet d'afficher du texte, des variables ou des r□sultats □ l'□cran.

Exemple :

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

Explications :

print() est une fonction int□gr□e en Python.

Elle affiche tout ce qui est pass□ entre parenth□ses.

Vous pouvez afficher des cha□nes de caract□res, des nombres, des variables, etc.

SAGIM

Lecture & Ecriture

2. La lecture (Input):

La lecture en Python se fait avec la fonction input().
Elle permet de r□cup□rer des donn□es saisies par l'utilisateur.

Exemple :

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

Explications :

input() est une fonction qui attend que l'utilisateur saisisse quelque chose au clavier.
Elle renvoie toujours une cha□ne de caract□res (m□me si l'utilisateur entre un nombre).
Pour convertir l'entr□e en un autre type (comme un entier ou un nombre d□cimal), vous devez utiliser des fonctions comme int() ou float().

SAGIM

3. Combinaison de lecture et d'□criture

Vous pouvez combiner input() et print() pour cr□er des programmes interactifs.

Exemple :

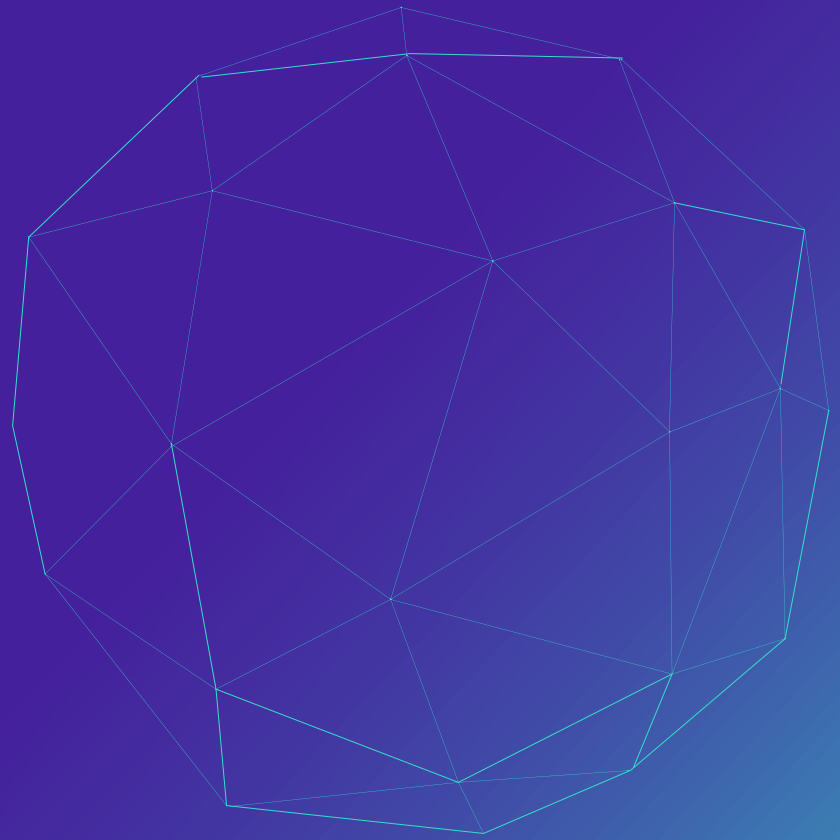
```
# Demander deux nombres □ l'utilisateur
nombre1 = float(input("Entrez le premier nombre : "))
nombre2 = float(input("Entrez le deuxi□me nombre : "))

# Calculer la somme
somme = nombre1 + nombre2

# Afficher le r□sultat
print("La somme de", nombre1, "et", nombre2, "est", somme)
```

SAGIM

PARTIE 3 :

Les Opérateurs :

Introdution au langage PYTHON



Opérations de Base

les op□rateurs arithm□tiques sont utilis□s pour effectuer des op□rations math□matiques de base sur des nombres.

Voici une explication des principaux op□rateurs arithm□tiques en Python :

SAGIM

Opérations de Base

Addition (+) : Cet opérateur permet d'ajouter deux nombres.

```
resultat = 5 + 3  # resultat vaudra 8
```

Soustraction (-) : Cet opérateur permet de soustraire un nombre d'un autre.

```
resultat = 10 - 4  # resultat vaudra 6
```

Multiplication (*) : Cet opérateur permet de multiplier deux nombres.

```
resultat = 7 * 2  # resultat vaudra 14
```

SAGIM

Opérations de Base

Division (/) : Cet opérateur permet de diviser un nombre par un autre. Le résultat est toujours un nombre flottant (même si la division est exacte).

```
resultat = 8 / 2  # resultat vaudra 4.0
```

Division entière (//) : Cet opérateur effectue une division et retourne le quotient entier (en ignorant le reste).

```
resultat = 10 // 3  # resultat vaudra 3
```

Modulo (%) : Cet opérateur retourne le reste de la division entière.

```
resultat = 10 % 3  # resultat vaudra 1
```

Exponentiation (**) : Cet opérateur permet d'élever un nombre à la puissance d'un autre.

```
resultat = 2 ** 3  # resultat vaudra 8
```

SAGIM

Opérations de Base

Remarques :

- ✓ **Les op□rateurs arithm□tiques** peuvent □galement □tre utilis□s avec des nombres □ virgule flottante (float).
- ✓ L'ordre des op□rations suit les r□gles math□matiques standard (priorit□ des op□rations), mais vous pouvez utiliser des parenth□ses pour modifier l'ordre d'□valuation.

```
resultat = (5 + 3) * 2  # resultat vaudra 16
```

SAGIM

Les opérateurs de comparaison

les op□rateurs logiques et les op□rateurs de comparaison sont utilis□s pour □valuer des conditions et prendre des d□cisions dans vos programmes.

Les op□rateurs de comparaison :

permettent de comparer deux valeurs. Ils retournent un bool□en (True ou False) selon que la condition est vraie ou fausse.

SAGIM

Les opérateurs de comparaison

Opérateur	Description	Exemple	Résultat
==	Égal à	5 == 5	True
!=	Différent de	5 != 3	True
>	Strictement supérieur à	10 > 5	True
<	Strictement inférieur à	10 < 5	False
>=	Supérieur ou égal à	10 >= 10	True
<=	Inférieur ou égal à	5 <= 3	False

SAGIM

Les opérateurs de comparaison

Exemples :

```
a = 10
b = 5

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

SAGIM

Opérateurs logiques

Les opérateurs logiques sont utilisés pour combiner plusieurs conditions.
Ils retournent également un booléen (**True** ou **False**).

SAGIM

Opérateurs logiques

Les opérateurs logiques sont utilisés pour combiner plusieurs conditions.
Ils retournent également un booléen (**True** ou **False**).

SAGIM

Opérateurs logiques

Opérateur	Description	Exemple	Résultat
and	Retourne True si toutes les conditions sont vraies	(5 > 3) and (10 < 20)	True
or	Retourne True si au moins une condition est vraie	(5 > 3) or (10 > 20)	True
not	Inverse le résultat de la condition	not (5 > 3)	False

SAGIM

Opérateurs logiques

EXEMPLE

```
x = 5
y = 10

# and
print(x > 0 and y < 20)  # True (les deux conditions sont vraies)

# or
print(x > 0 or y > 20)   # True (au moins une condition est vraie)

# not
print(not (x > 0))       # False (inverse la condition True)
```

SAGIM

Combinaison des opérateurs

Vous pouvez combiner les op□rateurs de comparaison et les op□rateurs logiques pour cr□er des conditions complexes.

Exemple :

```
age = 25
salaire = 40000

# Condition : l'□ge est sup□rieur □ 18 ET le salaire est inf□rieur □ 50000
if (age > 18) and (salaire < 50000) :
    print("Condition remplie !")
else:
    print("Condition non remplie.")
```

SAGIM

Priorité des opérateurs

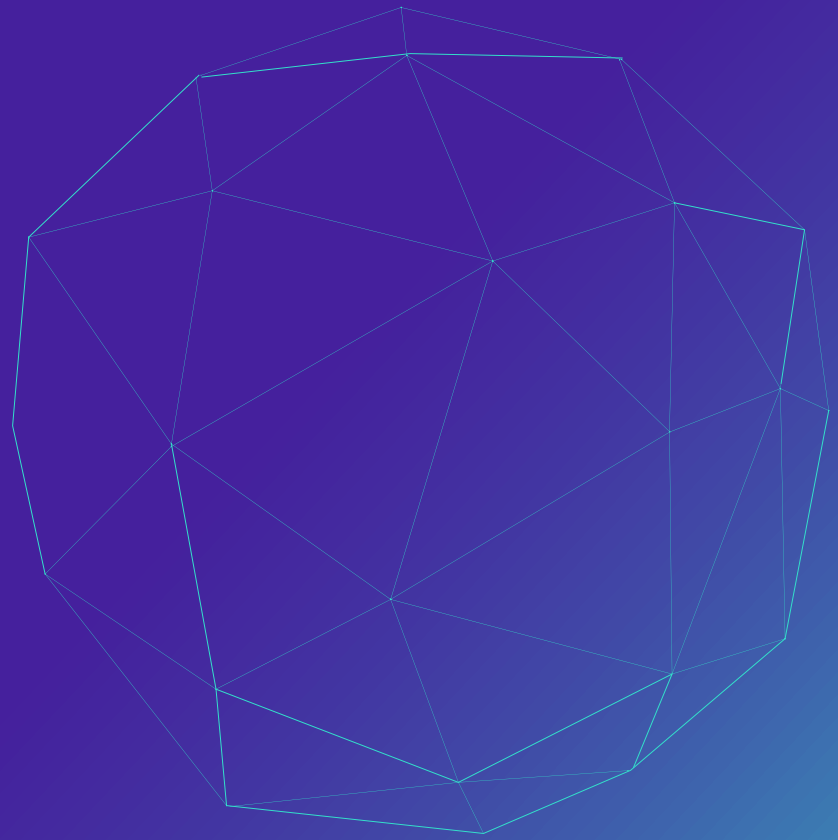
Lorsque vous combinez plusieurs op□rateurs, Python suit un ordre de priorit□ :

- ✓ Les op□rateurs de comparaison sont □valu□s en premier.
- ✓ Les op□rateurs logiques sont □valu□s ensuite, dans l'ordre suivant :
  not, and, or.

Exemple :

```
resultat = (5 > 3) and (10 < 20) or (not (2 == 2))
            # □tape 1 : (5 > 3) → True
            # □tape 2 : (10 < 20) → True
            # □tape 3 : (not (2 == 2)) → False
            # □tape 4 : True and True → True
            # □tape 5 : True or False → True
print(resultat)  # True
```

SAGIM



PARTIE 4 :

Structures de Contrôle

Introdution au langage PYTHON

Les structures de contrôle

Les structures de choix :

Les conditions en Python permettent d'ex□cuter des blocs de code sp□cifiques en fonction de la v□racit□ d'une expression.
La structure de base des conditions en Python utilise if, elif et else.
Structure de base des conditions :
if : Si la condition est vraie, ce bloc de code est ex□cut□.
elif (else if) : Permet de tester une autre condition si la premi□re est fausse.
else : Si aucune des conditions pr□c□dentes n'est vraie, ce bloc de code est ex□cut□.

SAGIM

Les structures de contrôle

Exemple :

```
x = 10
if x > 0:
    print("x est positif")
elif x < 0:
    print("x est n□gatif")
else:
    print("x est z□ro")
```

SAGIM

Exercice

□crire un programme qui demande □ l'utilisateur d'entrer son □ge et qui affiche s'il est mineur (moins de 18 ans), majeur (18-65 ans) ou senior (plus de 65 ans).

Les structures de contrôle

Solution

```
age = int(input("Entrez votre âge : "))
if age < 18:
    print("Vous êtes mineur.")
elif 18 <= age <= 65:
    print("Vous êtes majeur.")
else:
    print("Vous êtes senior.")
```

SAGIM

Les structures de contrôle

Solution

```
age = int(input("Entrez votre âge : "))
if age < 18:
    print("Vous êtes mineur.")
elif 18 <= age <= 65:
    print("Vous êtes majeur.")
else:
    print("Vous êtes senior.")
```

SAGIM