

**José Ouin**

Ingénieur INSA Toulouse

Ancien élève de l'ENS Cachan

Professeur Agrégé de Génie civil

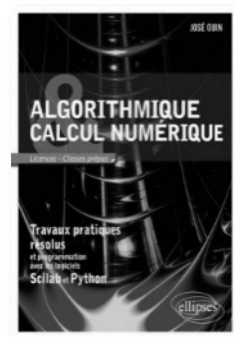
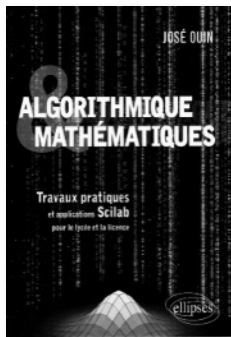
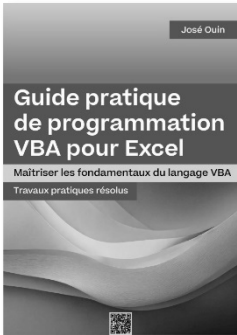
Professeur Agrégé de Mathématiques

# Python pour tous :

Les bases essentielles pour  
programmer en Python



Du même auteur aux Editions Ellipses et sur Amazon



ISBN : 978-2-9593648-7-7

© José OUIN – 2024 – <https://www.joseouin.fr>

Tous droits de traduction, de reproduction et d'adaptation réservés pour tous pays.

La loi du 11 mars 1957 n'autorisant, aux termes des alinéas 2 et 3 de l'article 41, d'une part, que les "copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective" et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, "toute représentation ou reproduction intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants droit ou ayant cause, est illicite" (alinéa 1<sup>er</sup> de l'article 40).

Cette représentation ou reproduction, par quelque procédé que ce soit, sans autorisation de l'auteur ou du Centre français du droit de copie (20, rue des Grands-Augustins 75006 Paris), constituerait donc une contrefaçon sanctionnée par les articles 425 et suivants du Code pénal

## Avant-propos

Cet ouvrage a pour but de vous guider, pas à pas, dans un apprentissage progressif et structuré de Python, en vous offrant les bases solides nécessaires pour comprendre et utiliser ce langage avec confiance.

L'apprentissage progressif est au cœur de ce livre. Chaque chapitre introduit de nouveaux concepts, en commençant par les fondamentaux pour évoluer progressivement vers des notions plus avancées. Grâce à une structure claire et une pédagogie adaptée, chaque section vous propose des exercices pratiques pour mettre en œuvre les notions abordées, ainsi que des travaux pratiques (TP) résolus pour vous permettre d'approfondir vos connaissances et de renforcer vos compétences.

Vous trouverez dans ces pages :

- Des exercices de base, conçus pour consolider votre compréhension et pratiquer les notions au fur et à mesure de leur introduction.
- Des travaux pratiques (TP) complets et résolus, qui vous permettront de travailler sur des projets plus complexes, avec une approche guidée mais encourageant également l'autonomie.

À travers ce parcours, ce livre vise à transformer l'apprentissage de la programmation en Python en une expérience enrichissante et motivante. En progressant à votre rythme, vous verrez qu'avec un peu de persévérance, chaque chapitre et chaque exercice vous rapprocheront de la maîtrise des bases de ce langage puissant.

Bonne découverte de Python, et surtout, bonne pratique ! Que cet ouvrage vous ouvre les portes d'un apprentissage enrichissant et vous inspire à aller encore plus loin dans le monde de la programmation.

José Quin 

- Un avis positif ?

Merci de prendre le temps de laisser votre évaluation (☆☆☆☆☆) sur la page Amazon de ce livre. Vos avis aident les autres lecteurs à mieux comprendre l'ouvrage.

- Un avis négatif, une question, une suggestion ou une remarque ?

N'hésitez pas à m'envoyer un message via le formulaire de contact de mon site Internet. Lien : <https://joseouin.fr/bandeaucontact>

# Table des matières

---

<b>1- Introduction au langage Python .....</b>	<b>9</b>
1-1. Présentation du langage et de ses applications .....	9
1-1.1 Pourquoi choisir Python ? .....	9
1-1.2 Les Applications de Python.....	9
1-2. Installation de Python et des éditeurs recommandés.....	11
1-2.1 Installation de Python .....	11
1-2.2 Editeur PyScripter .....	12
1-2.3 Editeur Spyder .....	15
1-2.4 Editeur PyScripter pour cet ouvrage .....	17
1-3. Premiers pas avec le mode script .....	18
1-3.1 Utiliser le mode script.....	18
1-3.2 Avantages du mode script .....	20
1-3.3 Exemple de Programme en mode script.....	20
<b>2- Bases de la programmation .....</b>	<b>21</b>
2-1. Variables et types de données principaux.....	21
2-1.1 Déclarer et utiliser une variable .....	21
2-1.2 Types de données principaux .....	21
2-1.3 Identifier le type de donnée d'une variable .....	23
2-1.4 Changer le type de donnée d'une variable .....	23
2-2. Opérations mathématiques et logiques .....	24
2-2.1 Opérateurs mathématiques .....	24
2-2.2 Opérateurs logiques .....	26
2-3. Saisie et affichage de données .....	28
2-3.1 La fonction print() : Afficher des informations.....	28
2-3.2 La fonction input() : Recevoir des données de l'utilisateur.....	29
<b>3- Structures de données .....</b>	<b>32</b>
3-1. Listes et manipulations de listes.....	32
3-1.1 Créer une liste .....	32
3-1.2 Accéder aux éléments d'une liste .....	32
3-1.3 Modifier les éléments d'une liste .....	33
3-1.4 Opérations de base sur les listes.....	33
3-1.5 Slices : Extraire des parties d'une liste .....	35
3-1.6 Trier et inverser une liste .....	35

3-2.	Introduction à NumPy : tableau array et opérations de base. ....	37
3-2.1	Créer un tableau NumPy .....	38
3-2.2	Opérations de base avec les tableaux NumPy .....	39
3-2.3	Fonctions mathématiques.....	40
3-2.4	Travailler avec des tableaux multidimensionnels.....	40
3-3.	Comparaison entre les types list et array. ....	42
3-3.1	Différences de base entre list et array .....	42
3-3.2	Exemple de création d'une liste et d'un tableau .....	43
3-3.3	Différence de comportement pour les opérations mathématiques.....	43
<b>4-</b>	<b>Contrôles de flux.....</b>	<b>45</b>
4-1.	Conditions et instructions de contrôle (if, elif, else). ....	45
4-1.1	Structure de base des conditions.....	45
4-1.2	Exemple simple avec if, elif, else .....	46
4-1.3	Opérateurs de comparaison .....	46
4-1.4	Exemple : Vérifier la parité d'un nombre.....	47
4-1.5	Conditions imbriquées .....	47
4-1.6	Utilisation de conditions multiples avec and et or .....	48
4-2.	Boucles : for, while et importance de l'indentation. ....	49
4-2.1	La boucle for .....	49
4-2.2	La boucle while .....	50
4-2.3	Importance de l'indentation.....	52
4-2.4	Comparaison entre for et while.....	52
4-3.	Instructions spéciales : break, continue. ....	53
4-3.1	L'instruction break.....	53
4-3.2	L'instruction continue .....	54
4-3.3	Résumé des utilisations de break et continue .....	55
<b>5-</b>	<b>Fonctions et modules .....</b>	<b>56</b>
5-1.	Définition et utilisation de fonctions personnalisées.....	56
5-1.1	Définition d'une fonction.....	56
5-1.2	Utilité des fonctions : éviter les redondances.....	57
5-1.3	Exemple : Calcul de la factorielle d'un nombre .....	57
5-1.4	Utilisation de return dans les fonctions .....	58
5-1.5	Avantages des fonctions.....	59

5-2.	Importation et utilisation des bibliothèques .....	60
5-2.1	Importation de modules .....	60
5-2.2	La bibliothèque math .....	60
5-2.3	La bibliothèque random .....	61
5-2.4	La bibliothèque NumPy.....	62
5-2.5	Importer des fonctions spécifiques depuis un module .....	64
<b>6-</b>	<b>Programmation Orientée Objet (POO).....</b>	<b>66</b>
6-1.	Introduction à la POO : classes, attributs, méthodes. ....	66
6-1.1	Qu'est-ce qu'une classe ?.....	66
6-1.2	Attributs : Propriétés d'un objet.....	66
6-1.3	Méthodes : Actions d'un objet.....	67
6-1.4	Exemple complet : Créer et utiliser des objets.....	67
6-2.	Création d'objets et utilisation de classes simples. ....	68
6-2.1	Formater les messages avec les f-strings .....	69
6-2.2	Définir la classe Voiture .....	70
6-2.3	Créer et utiliser des objets de la classe Voiture.....	70
6-2.4	Ajouter d'autres méthodes : Freiner et afficher les informations.....	71
<b>7-</b>	<b>Graphiques et visualisation de données .....</b>	<b>73</b>
7-1.	Utilisation de Matplotlib pour tracer des graphiques simples (courbes, nuages de points, histogrammes). ....	73
7-1.1	Tracer une courbe simple .....	74
7-1.2	Créer un nuage de points (Scatter Plot) .....	75
7-1.3	Créer un histogramme .....	77
7-2.	Graphiques avancés : représentation 3D et personnalisation des graphiques.....	79
7-2.1	Tracer une courbe 3D .....	79
7-2.2	Tracer une surface 3D .....	81
7-2.3	Représenter un nuage de points 3D .....	82
7-2.4	Personnalisation des graphiques 3D .....	84
7-2.5	Exemple de surface avec personnalisation complète.....	84
<b>8-</b>	<b>Algèbre linéaire et traitement des données.....</b>	<b>86</b>
8-1.	Opérations de base en algèbre linéaire avec Numpy (multiplication de matrices, déterminants, vecteurs propres). ....	86
8-1.1	Multiplication de matrices.....	86
8-1.2	Calcul du déterminant d'une matrice .....	87
8-1.3	Vecteurs propres et valeurs propres.....	88
8-1.4	Calcul de l'inversion d'une matrice .....	89

8-2.	Applications pratiques pour la résolution de systèmes linéaires.....	90
8-2.1	Résolution d'un système linéaire avec NumPy.....	90
8-2.2	Systèmes linéaires plus complexes.....	92
8-2.3	Résolution d'un système sans solution unique.....	93
<b>9-</b>	<b>Quiz - Validation des acquis en Python.....</b>	<b>94</b>
9-1.	Enoncé du quiz.....	94
9-2.	Solutions du Quiz.....	101
<b>10-</b>	<b>Exercices de base.....</b>	<b>105</b>
10-1.	Enoncés des exercices de base.....	105
10-1.1	Hello, World !.....	105
10-1.2	Addition de deux nombres.....	105
10-1.3	Calculer l'âge en années.....	105
10-1.4	Convertir des kilomètres en miles.....	105
10-1.5	Vérification de la parité d'un nombre.....	105
10-1.6	Affichage des multiples de 3 jusqu'à 30.....	105
10-1.7	Calcul de l'aire d'un rectangle.....	106
10-1.8	Table de multiplication.....	106
10-1.9	Trouver le plus grand de trois Nombres.....	106
10-1.10	Échanger les valeurs de deux variables.....	106
10-1.11	Conversion de Celsius en Fahrenheit.....	106
10-1.12	Vérification d'un mot de passe.....	106
10-1.13	Affichage de nombres pairs.....	106
10-1.14	Vérification de la positivité d'un nombre.....	106
10-1.15	Calcul de la moyenne de trois nombres.....	106
10-1.16	Compter le nombre de voyelles dans une phrase.....	106
10-1.17	Générateur de nombres aléatoires.....	107
10-1.18	Calcul du périmètre d'un cercle.....	107
10-1.19	Affichage d'une liste d'éléments.....	107
10-1.20	Somme des n premiers entiers.....	107
10-2.	Solutions des exercices de base.....	108
10-2.1	Hello, World !.....	108
10-2.2	Addition de deux nombres.....	108
10-2.3	Calculer l'âge en années.....	108
10-2.4	Convertir des kilomètres en miles.....	109
10-2.5	Vérification de la parité d'un nombre.....	109
10-2.6	Affichage des multiples de 3 jusqu'à 30.....	109
10-2.7	Calcul de l'aire d'un rectangle.....	110
10-2.8	Table de multiplication.....	110
10-2.9	Trouver le plus grand de trois Nombres.....	111

10-2.10	Échanger les valeurs de deux variables .....	111
10-2.11	Conversion de Celsius en Fahrenheit.....	112
10-2.12	Vérification d'un mot de passe.....	112
10-2.13	Affichage de nombres pairs.....	113
10-2.14	Vérification de la positivité d'un nombre .....	113
10-2.15	Calcul de la moyenne de trois nombres.....	113
10-2.16	Compter le nombre de voyelles dans une phrase.....	114
10-2.17	Générateur de nombres aléatoires .....	115
10-2.18	Calcul du périmètre d'un cercle .....	115
10-2.19	Affichage d'une liste d'éléments .....	115
10-2.20	Somme des n premiers entiers.....	116

## **11- Les travaux pratiques ..... 117**

TP 1	– Algorithme d'Euclide : détermination du PGCD de deux entiers naturels....	119
TP 2	– Ensemble des diviseurs positifs d'un entier naturel .....	121
TP 3	– Recherche d'une racine carrée par la méthode de dichotomie.....	123
TP 4	– Test de primalité d'un nombre .....	126
TP 5	– Etude de la suite de Syracuse.....	129
TP 6	– Suite de Fibonacci et visualisation.....	133
TP 7	– Somme des chiffres d'un nombre et calcul de la persistance .....	137
TP 8	– Palindrome et inversion de chaîne.....	140
TP 9	– Analyse statistique de données simulées .....	143
TP 10	– Chiffrement/Déchiffrement de Vigenère.....	149
TP 11	– Création d'un générateur de mot de passe aléatoire.....	154
TP 12	– Gestion d'un compte bancaire en Programmation Orientée Objet .....	158

## **12- Téléchargement des ressources de cet ouvrage..... 163**



## 1-2. Installation de Python et des éditeurs recommandés

Pour débiter avec Python, il est essentiel d'installer le langage sur votre machine ainsi qu'un éditeur de code adapté pour écrire et exécuter vos programmes. Voici les étapes pour installer Python et les recommandations d'éditeurs.

### 1-2.1 Installation de Python

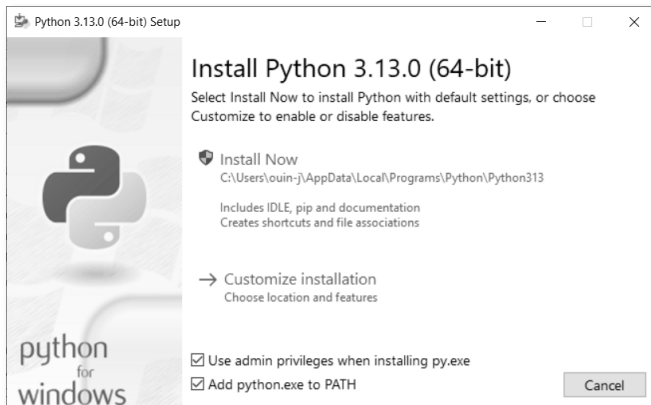
#### • Télécharger Python

- Rendez-vous sur le site officiel de Python <https://www.python.org/downloads>
- Accédez à cette section « Downloads » et choisissez la version appropriée pour votre système d'exploitation (Windows, macOS, ou Linux).
- Téléchargez la version 3.x.

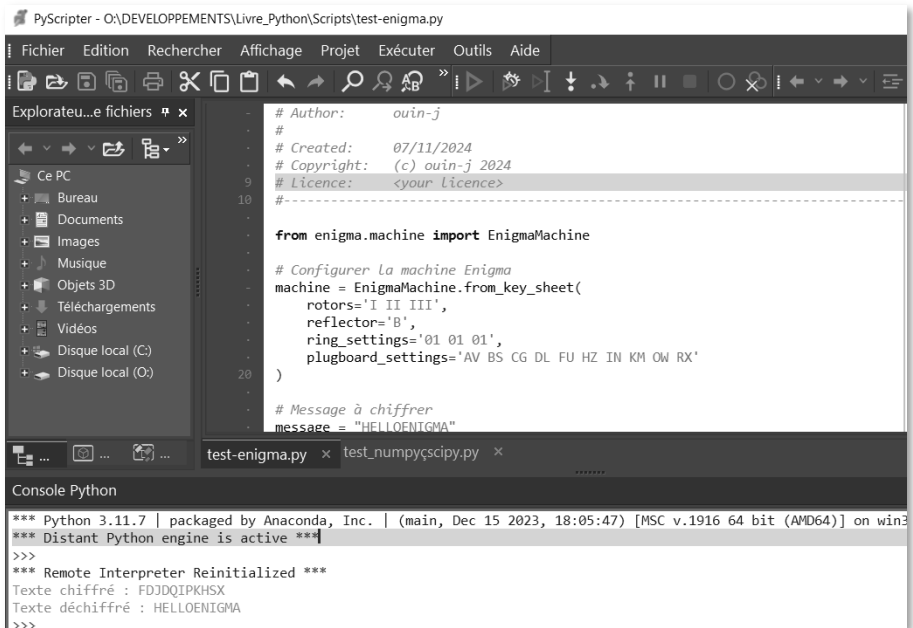
#### • Installer Python

- Lancez l'installateur téléchargé. Pour les utilisateurs de Windows, veillez à cocher la case « Add Python.exe to PATH » avant de continuer. Cela permet de rendre Python accessible depuis la ligne de commande.
- Suivez les instructions pour compléter l'installation. Sur Windows et macOS, l'installateur installe Python, ainsi que le gestionnaire de paquets **pip** (utilisé pour installer des bibliothèques supplémentaires).

La capture d'écran suivante montre les options à cocher lors de l'installation de Python sur Windows :



*Logiciel d'installation de Python pour Windows*



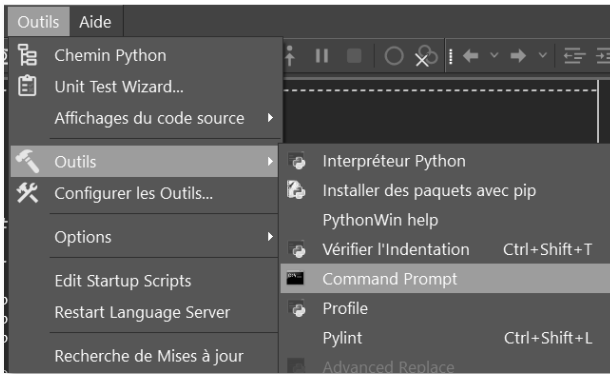
*Interface du logiciel PyScripter*

Pour les menus en français, sélectionner « French » depuis le menu suivant : « View/Language/French »



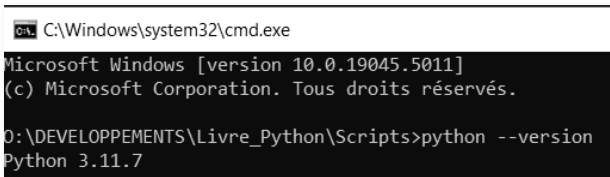
*Menu pour obtenir l'interface en langue Française*

Pour connaître la version de Python qui est installée, sélectionner le menu : « Outils/Outils/Command prompt » puis taper la commande « python --version »



*Accès à l'invite de commande « Command Prompt »*

La version qui s'affiche est : Python 3.11.7

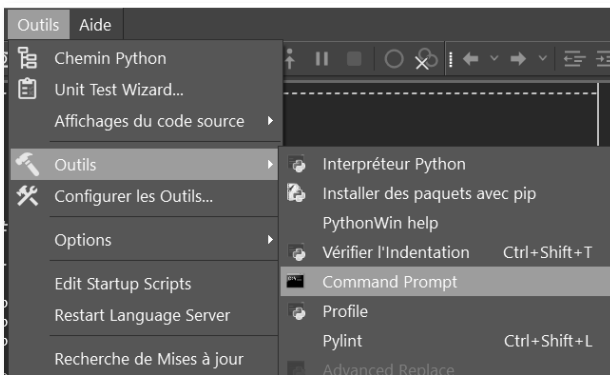


*Affichage de la version dans une invite de commande*

### **Pour installer une bibliothèque Python :**

Exemple d'installation de la bibliothèque NumPy avec PyScripter :

Sélectionner le menu : « Outils/Outils/Command prompt » puis taper la commande « pip install numpy »



*Accès à l'invite de commande « Command Prompt »*

```

# Conversion de chaîne vers flottant
texte_decimal = "3.14"
nombre_decimal = float(texte_decimal)
# nombre_decimal contient 3.14 comme flottant

# Conversion d'un entier vers une chaîne
age = 25
age_str = str(age) # age_str contient "25" comme chaîne

```

Pour conclure, les variables et les types de données sont des éléments fondamentaux pour écrire des programmes. Comprendre les différents types de données et savoir comment les manipuler permet de construire des programmes plus puissants et d'utiliser efficacement Python pour représenter les informations nécessaires à un projet.

## 2-2. Opérations mathématiques et logiques

Dans cette partie, on définit les différents opérateurs mathématiques et logiques en Python. Ces opérateurs sont essentiels pour effectuer des calculs, comparer des valeurs, et prendre des décisions dans les programmes.

### 2-2.1 Opérateurs mathématiques

Les opérateurs mathématiques permettent de réaliser des opérations arithmétiques de base sur des nombres, comme l'addition, la soustraction, la multiplication, et la division.

Voici les opérateurs mathématiques de base en Python :

Opérateur	Description	Exemple	Résultat
+	Addition	5 + 3	8
-	Soustraction	10 - 4	6
*	Multiplication	7 * 2	14
/	Division	15 / 3	5
//	Division entière	15 // 2	7
%	Modulo (reste)	10 % 3	1
**	Exponentiation	3 ** 2	9

## Exemples de scripts :

Voici quelques exemples de scripts utilisant ces opérateurs :

### Script 1 : Opérations mathématiques de base

```
# Script d'exemples d'opérations mathématiques
a = 10
b = 3

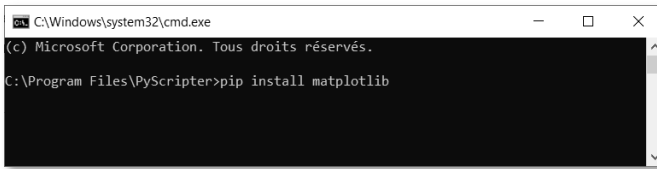
addition = a + b
soustraction = a - b
multiplication = a * b
division = a / b
division_entier = a // b
reste = a % b
puissance = a ** b
print("Addition :", addition)           # Affiche 13
print("Soustraction :", soustraction)   # Affiche 7
print("Multiplication :", multiplication) # Affiche 30
print("Division :", division)           # Affiche 3.333...
print("Division entière :", division_entier) # Affiche 3
print("Reste :", reste)                 # Affiche 1
print("Puissance :", puissance)        # Affiche 1000
```

### Script 2 : Utilisation de l'opérateur modulo

L'opérateur modulo (%) est souvent utilisé pour vérifier si un nombre est pair ou impair :

```
# Script pour vérifier si un nombre est pair ou impair
nombre = 8
if nombre % 2 == 0:
    print(nombre, "est pair")
else:
    print(nombre, "est impair")
```

Dans cet exemple, `nombre % 2` calcule le reste de la division de `nombre` par 2. Si le reste est 0, le nombre est pair, sinon il est impair.



```
C:\Windows\system32\cmd.exe
(c) Microsoft Corporation. Tous droits réservés.
C:\Program Files\PyScripter>pip install matplotlib
```

Saisie de la commande « `pip install matplotlib` » dans une invite de commande

## Importer Matplotlib

La plupart des utilisateurs de Matplotlib importent le sous-module `pyplot`, qui simplifie la création de graphiques en fournissant des fonctions similaires aux commandes graphiques d'autres logiciels, comme Matlab.

```
import matplotlib.pyplot as plt
```

### 7-1.1 Tracer une courbe simple

Une courbe (ou graphe en ligne) est souvent utilisée pour visualiser des données continues, comme des variations de valeurs sur le temps. Voici un exemple d'une simple courbe représentant une fonction mathématique.

#### Exemple : Tracer une fonction quadratique

Le script ci-dessous montre comment tracer la courbe de la fonction  $y = x^2$ .

```
import matplotlib.pyplot as plt

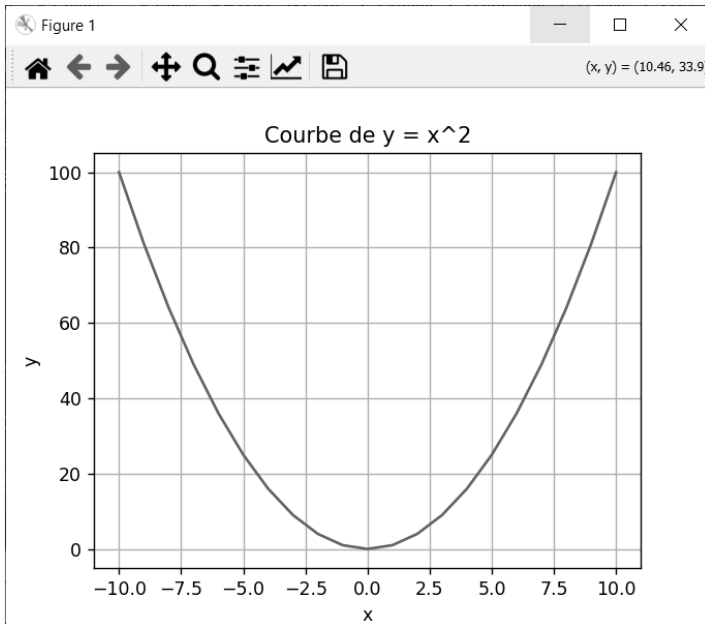
# Données pour x et y
x = range(-10, 11)          # x varie de -10 à 10
y = [i**2 for i in x]      # y = x^2

# Tracé de la courbe
plt.plot(x, y)
plt.title("Courbe de y = x^2") # Titre du graphique
plt.xlabel("x")                # Label pour l'axe des x
plt.ylabel("y")                # Label pour l'axe des y
plt.grid(True)                 # Affichage de la grille
plt.show()
```

### Dans ce script :

- `plt.plot(x, y)` trace la courbe en reliant les points définis par les coordonnées `x` et `y`.
- `plt.title`, `plt.xlabel`, et `plt.ylabel` ajoutent un titre et des légendes aux axes.
- `plt.grid(True)` affiche une grille pour faciliter la lecture des points.

On obtient le graphique suivant :



### 7-1.2 Créer un nuage de points (Scatter Plot)

Un nuage de points est idéal pour visualiser la relation entre deux variables quantitatives. Chaque point représente une paire de valeurs.

#### Exemple : Nuage de points aléatoire

Le script suivant génère un nuage de points en utilisant des valeurs aléatoires.

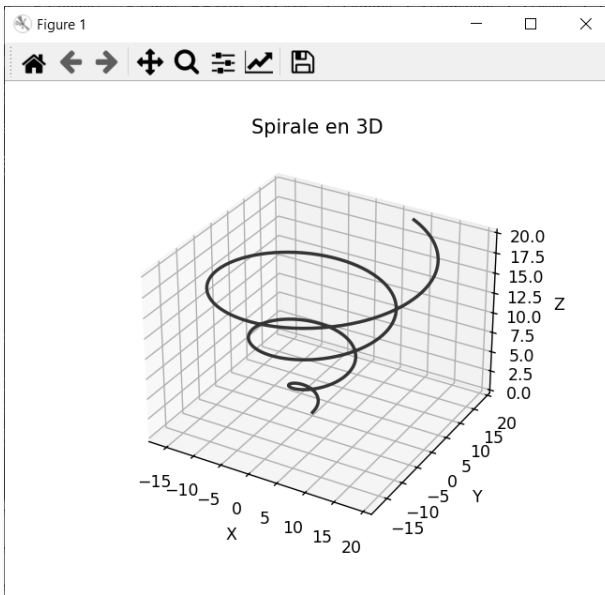
```
# Tracé de La courbe 3D
ax.plot(x, y, z, color='purple', linewidth=2)
ax.set_title("Spirale en 3D")
ax.set_xlabel("X")
ax.set_ylabel("Y")
ax.set_zlabel("Z")

plt.show()
```

**Dans cet exemple :**

- $t$  est un tableau de valeurs allant de 0 à 20. Les fonctions  $x = t * \cos(t)$ ,  $y = t * \sin(t)$ , et  $z = t$  définissent les coordonnées de la spirale en fonction de  $t$ .
- `ax.plot(x, y, z, color='purple', linewidth=2)` trace la courbe avec une couleur personnalisée et une largeur de ligne de 2.

On obtient le graphique suivant :



**Remarque :**

le module NumPy propose un large éventail de fonctions mathématiques similaires au module standard `math` en Python. Cependant, il est optimisé pour travailler avec des tableaux et des opérations vectorielles, ce qui permet d'effectuer des calculs sur des collections de données de manière très efficace. NumPy est bien plus performant que `math` pour le traitement de grandes quantités de données.



## 9- Quiz - Validation des acquis en Python

Ce quiz est conçu pour vous permettre de faire le point sur vos connaissances et de valider votre compréhension des concepts abordés dans les chapitres précédents. Les questions couvrent des thèmes essentiels, tels que les bases du langage Python, la manipulation des structures de données, les contrôles de flux, la programmation orientée objet, et la visualisation de données.

En testant vos acquis, ce quiz vous aidera à identifier les notions maîtrisées ainsi que les points nécessitant une révision. Prenez le temps de répondre à chaque question et consultez les corrigés pour vous assurer une compréhension solide des fondements de Python. Bonne évaluation !

### 9-1. Enoncé du quiz

#### Chapitre : Introduction au langage Python

---

1. **Quelle est l'une des principales applications de Python en programmation ?**
  - A. Réalisation de films uniquement
  - B. Analyse de données, développement web, IA, et plus encore
  - C. Création de jeux uniquement
  - D. Gestion de bases de données uniquement
2. **Quel éditeur est souvent recommandé pour débiter avec Python ?**
  - A. Microsoft Word
  - B. PyScripter
  - C. Adobe Photoshop
  - D. Windows Media Player
3. **Comment exécuter un script Python dans un éditeur comme PyScripter ou Spyder ?**
  - A. En cliquant sur "Exécuter" ou "Run"
  - B. En ouvrant le fichier avec Excel
  - C. En tapant "Exit" dans le terminal
  - D. En renvoyant le script à l'éditeur

# Les exercices de base

## **10- Exercices de base**

Ces exercices de base sont conçus pour vous permettre de mettre en pratique les concepts essentiels de Python abordés dans les chapitres précédents. Chaque exercice vous offre l'occasion d'appliquer directement les notions étudiées et de renforcer votre compréhension du langage.

Prenez le temps de résoudre chaque problème par vous-même. Si un exercice vous semble difficile, n'hésitez pas à relire les sections correspondantes et à vous inspirer des exemples fournis dans ce livre. Revenir aux bases et revoir les explications peut souvent vous apporter la clarté nécessaire pour avancer.

Rappelez-vous : l'apprentissage de la programmation repose sur la pratique et la persévérance. En progressant dans ces exercices, vous construisez peu à peu les bases solides de votre maîtrise de Python. Bon courage !

### **10-1. Énoncés des exercices de base**

#### **10-1.1 Hello, World !**

Écrivez un programme qui affiche le message "Hello, World!" à l'écran.

#### **10-1.2 Addition de deux nombres**

Créez un programme qui demande à l'utilisateur deux nombres et affiche leur somme.

#### **10-1.3 Calculer l'âge en années**

Demandez à l'utilisateur son année de naissance et affichez son âge actuel en soustrayant cette année de l'année en cours.

#### **10-1.4 Convertir des kilomètres en miles**

Écrivez un programme qui demande une distance en kilomètres et la convertit en miles (1 km = 0,621371 mile).

#### **10-1.5 Vérification de la parité d'un nombre**

Créez un programme qui demande un nombre à l'utilisateur et affiche s'il est pair ou impair.

#### **10-1.6 Affichage des multiples de 3 jusqu'à 30**

Écrivez un programme qui affiche tous les multiples de 3 entre 1 et 30.

## 10-2. Solutions des exercices de base

### 10-2.1 Hello, World !

Écrivez un programme qui affiche le message "Hello, World!" à l'écran.

```
# Affiche le message "Hello, World!" à l'écran
print("Hello, World!")
```

### 10-2.2 Addition de deux nombres

Créez un programme qui demande à l'utilisateur deux nombres et affiche leur somme.

```
# Demande à l'utilisateur d'entrer deux nombres
nombre1 = float(input("Entrez le premier nombre : "))
nombre2 = float(input("Entrez le deuxième nombre : "))

# Calcule la somme des deux nombres
somme = nombre1 + nombre2

# Affiche le résultat
print("La somme est :", somme)
```

### 10-2.3 Calculer l'âge en années

Demandez à l'utilisateur son année de naissance et affichez son âge actuel en soustrayant cette année de l'année en cours.

```
# Demande à l'utilisateur son année de naissance
annee_naissance = int(input("Entrez votre année de naissance : "))

# Calcule l'âge en soustrayant l'année de naissance de l'année
actuelle
annee_actuelle = 2024 # Remplacez par l'année actuelle si besoin
age = annee_actuelle - annee_naissance

# Affiche l'âge
print("Votre âge est :", age, "ans")
```

# **Les travaux pratiques**

# TP 1

## Algorithme d'Euclide : détermination du PGCD de deux entiers naturels

### Objectif

Ce TP a pour but de vous familiariser avec l'algorithme d'Euclide, un algorithme classique en mathématiques et en informatique, utilisé pour calculer le Plus Grand Commun Diviseur (PGCD) de deux nombres entiers. En programmant cet algorithme en Python, vous renforcerez vos compétences en manipulation de boucles et de conditions.

### Contexte

Le PGCD de deux nombres entiers est le plus grand entier qui divise ces deux nombres sans laisser de reste. L'algorithme d'Euclide est une méthode efficace et intuitive pour trouver ce PGCD en utilisant des divisions successives.

### Description de l'algorithme

L'algorithme d'Euclide repose sur les étapes suivantes :

1. On prend deux entiers  $a$  et  $b$  avec  $a \geq b$ .
2. Tant que  $b$  n'est pas nul :
  - On remplace  $a$  par  $b$  et  $b$  par le reste de la division de  $a$  par  $b$ .
3. Lorsque  $b$  devient nul,  $a$  contient le PGCD des deux nombres initiaux.

En d'autres termes, chaque itération consiste à réduire le problème en remplaçant les nombres par leurs restes successifs jusqu'à ce que l'un des deux soit nul. Le nombre restant est alors le PGCD recherché.

### Consignes

#### Étape 1 : Initialisation des variables

- Demandez à l'utilisateur de saisir deux entiers  $a$  et  $b$ .

#### Étape 2 : Implémentation de l'algorithme d'Euclide

- Utilisez une boucle `while` pour appliquer l'algorithme d'Euclide :  
Tant que  $b$  n'est pas nul, remplacez  $a$  par  $b$  et  $b$  par  $a \% b$ .

#### Étape 3 : Affichage du résultat

- Affichez la valeur de  $a$  une fois que la boucle est terminée, car elle représente le PGCD des deux nombres saisis.

### Travail demandé

Ecrire le script Python de l'algorithme d'Euclide afin de déterminer le PGCD de deux entiers saisis par l'utilisateur.

## Solution du TP 1

```
# Demande à l'utilisateur de saisir deux entiers
a = int(input("Entrez le premier entier : "))
b = int(input("Entrez le deuxième entier : "))

# Algorithme d'Euclide pour calculer le PGCD
while b != 0:
    a, b = b, a % b

# Affiche le PGCD
print("Le PGCD des deux nombres est :", a)
```

### Détails du script :

- `input()` recueille les deux nombres saisis par l'utilisateur et les convertit en `int`.
- La boucle `while` applique l'algorithme d'Euclide :
  - Tant que `b` n'est pas égal à 0, `a` est remplacé par `b` et `b` par `a % b` (le reste de la division de `a` par `b`).
- Lorsque `b` devient 0, `a` contient le PGCD des deux nombres initiaux.
- `print()` affiche le résultat final, qui est le PGCD des deux nombres saisis.

### On obtient dans la console Python :

```
*** Remote Interpreter Reinitialized ***
Entrez le premier entier : 45
Entrez le deuxième entier : 225
Le PGCD des deux nombres est : 45

*** Remote Interpreter Reinitialized ***
Entrez le premier entier : 273
Entrez le deuxième entier : 987
Le PGCD des deux nombres est : 21

*** Remote Interpreter Reinitialized ***
Entrez le premier entier : 182
Entrez le deuxième entier : 195
Le PGCD des deux nombres est : 13
```

## TP 3

## Recherche d'une racine carrée par la méthode de dichotomie

### Objectif

Ce TP a pour objectif de vous familiariser avec la méthode de dichotomie, une technique de recherche très efficace qui permet de trouver des solutions approchées pour certains types d'équations ou de problèmes mathématiques. Ici, vous l'utiliserez pour déterminer la racine carrée approchée d'un nombre donné par l'utilisateur.

### Contexte

La méthode de dichotomie, aussi appelée méthode de bisection, est une méthode d'approximation. Elle repose sur le principe de réduire progressivement l'intervalle dans lequel se situe la solution, en divisant cet intervalle par deux à chaque étape. C'est une technique simple et très utile pour résoudre des équations ou chercher des valeurs dans un intervalle, tant que la fonction est continue et monotone dans cet intervalle.

Dans ce TP, vous allez appliquer cette méthode pour approximer la racine carrée d'un nombre réel positif  $N$ .

### Description de la méthode de dichotomie pour déterminer une racine carrée

1. On choisit un intervalle  $[a, b]$  qui contient la solution :  
Pour une racine carrée, cet intervalle est souvent choisi entre  $a = 0$  et  $b = N$ , car la racine carrée de  $N$  se trouve dans cet intervalle pour tout  $N \geq 1$ .
2. À chaque étape :  
On calcule le milieu  $m$  de l'intervalle, soit  $m = (a + b) / 2$ .  
On compare  $m*m$  avec  $N$  :
  - Si  $m*m$  est proche de  $N$  (c'est-à-dire dans une marge d'erreur acceptable  $\epsilon$ ), alors  $m$  est une bonne approximation de la racine carrée de  $N$ .
  - Si  $m*m$  est inférieur à  $N$ , cela signifie que la racine carrée est dans l'intervalle  $[m, b]$ . On met donc à jour :  $a = m$
  - Si  $m*m$  est supérieur à  $N$ , cela signifie que la racine carrée est dans l'intervalle  $[a, m]$ . On met donc à jour :  $b = m$

On réduit ainsi l'intervalle de recherche jusqu'à atteindre une précision souhaitée.
3. On répète ce processus jusqu'à ce que l'intervalle soit suffisamment petit, ou que  $m*m$  soit assez proche de  $N$  (défini par une précision  $\epsilon$ ).



## TP 4

## Test de primalité d'un nombre

### Objectif

Ce TP a pour objectif de vous familiariser avec le test de primalité, un algorithme classique en mathématiques et en informatique. Vous allez écrire un programme en Python qui détermine si un nombre entier donné par l'utilisateur est premier ou non. Ce TP vous aidera à comprendre les concepts de divisibilité et d'optimisation dans les boucles, essentiels pour écrire des programmes efficaces.

### Contexte

Un nombre premier est un entier naturel supérieur à 1 qui n'a que deux diviseurs : 1 et lui-même. En d'autres termes, un nombre est premier s'il ne peut pas être divisé par un autre nombre entier sans laisser de reste, sauf par 1 et par lui-même.

### Exemples :

Les nombres premiers entre 1 et 20 sont : 2, 3, 5, 7, 11, 13, 17, 19.

Par contre, 4, 6, 8, 9, 10, etc., ne sont pas premiers car ils ont des diviseurs autres que 1 et eux-mêmes.

Dans ce TP, il s'agit de développer un programme qui vérifie si un nombre donné est premier, en utilisant une technique de réduction de la recherche pour rendre l'algorithme plus efficace.

### Description de l'algorithme

#### Cas simples :

Si le nombre est inférieur à 2, il n'est pas premier.

Si le nombre est 2, il est premier (c'est le seul nombre pair qui est premier).

#### Optimisation de la vérification :

- Pour un nombre  $n$  supérieur à 2, on peut tester la divisibilité uniquement pour les **nombre impairs** jusqu'à la racine carrée de  $n$  :  $\sqrt{n}$ 
  - Si  $n$  est divisible par un nombre dans cet intervalle, il n'est pas premier.
  - Si aucun diviseur n'est trouvé dans cet intervalle, alors  $n$  est premier.
- Cette approche réduit considérablement le nombre de vérifications nécessaires et améliore l'efficacité du programme.

**Théorème :** Soit  $n$  un entier supérieur à 1. Si aucun entier positif  $d$  inférieur ou égal à  $\sqrt{n}$  ne divise  $n$ , alors  $n$  est un nombre premier.

**Objectif**

Ce TP a pour but d'explorer la suite de Syracuse (aussi appelée suite de Collatz ou conjecture de Syracuse), une suite mathématique fascinante et mystérieuse. Il s'agit de programmer cette suite en Python et d'observer son comportement pour différents nombres entiers positifs. Ce TP permet de travailler sur les boucles et conditions, tout en explorant une conjecture mathématique encore non résolue.

**Contexte**

La suite de Syracuse est définie pour un nombre entier positif  $n$  selon les règles suivantes :

- Si  $n$  est pair, alors le terme suivant est  $\frac{n}{2}$ .
- Si  $n$  est impair, alors le terme suivant est  $3n + 1$ .

La conjecture de Syracuse affirme que, quel que soit le nombre initial choisi, la suite finira toujours par atteindre 1, puis entrera dans le cycle 1,4,2,1 indéfiniment. Cependant, cette conjecture reste non prouvée pour l'ensemble des nombres entiers.

**Exemples :**

Pour un point de départ de  $n = 6$ , la suite est : 6, 3, 10, 5, 16, 8, 4, 2, 1.

Pour  $n = 11$ , la suite est : 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1.

**Consignes****Étape 1 : Saisie de l'entier initial**

- Demander à l'utilisateur de saisir un entier positif de départ pour la suite de Syracuse.

**Étape 2 : Génération de la suite**

- Utiliser une boucle pour appliquer les règles de la suite de Syracuse au nombre initial jusqu'à atteindre 1.
- Afficher chaque terme généré dans la suite, pour visualiser l'évolution de la suite jusqu'à ce qu'elle atteigne 1.

**Étape 3 : Calcul du nombre d'itérations**

- Compter et afficher le nombre d'itérations nécessaires pour que la suite atteigne 1.

**Étape 4 : Valeur maximale de la suite**

- Identifier la valeur maximale atteinte dans la suite avant d'atteindre 1.