PYTHON — Cours et exercices

Logiciels pour coder avec Python

Thonny

Installation

1. Aller sur le site officiel : https://thonny.org
2. Télécharger la version pour Windows / macOS / Linux.
3. Installer en suivant les instructions (l'installation inclut Python).

Vérification

- Ouvrir Thonny.
- Dans la console intégrée, taper :

```
print("Bonjour, Python fonctionne !")
```

- Si le message s'affiche, tout est prêt.

Utilisation

- Créer un nouveau fichier `.py` pour chaque exercice.
- Copier-coller les exercices.
- Exécuter avec le bouton **Run** (ou F5).
- Observer la sortie dans la console.

Avantages :

- Bon pour tester variables, conditions, boucles, fonctions, etc.
- Affiche clairement les erreurs pour faciliter la correction.

Jupyter Notebook

Installation via Anaconda

1. Télécharger Anaconda : https://www.anaconda.com/products/distribution
2. Installer en suivant les instructions.
3. Ouvrir **Anaconda Navigator**, puis lancer **Jupyter Notebook**.

Ou installation minimale avec pip

```

```

```
pip install notebook
jupyter notebook
```

- Cela ouvrira Jupyter dans le navigateur.
- Créer un nouveau **Notebook Python 3** pour les exercices.

Utilisation

- Chaque cellule peut contenir du code ou du texte explicatif (Markdown).
- Permet d'intégrer directement :
  - Le code.
  - La solution dans une autre cellule.

Avantages :

- Support interactif.
- Mélange texte, code, images, et graphiques.

VS Code pour Python

Installer les outils nécessaires

1. **Installer Python**
   - Télécharger depuis python.org.
   - Pendant l'installation, **cocher "Add Python to PATH"**.
2. **Installer Visual Studio Code (VS Code)**
   - Télécharger depuis code.visualstudio.com.
   - Installation standard.

Installer les extensions Python dans VS Code

1. Ouvrir VS Code.
2. Aller dans l'onglet **Extensions** (ou `Ctrl+Shift+X`).
3. Chercher et installer :
   - **Python** (Microsoft) → indispensable pour coder en Python.
   - **Pylance** → autocomplétion et vérification des erreurs.
   - **Jupyter** → facultatif, si on veut des notebooks interactifs.

Configurer l'interpréteur Python

1. Ouvrir la palette de commandes : `Ctrl+Shift+P`.

2. Taper et sélectionner : **Python: Select Interpreter**.
3. Choisir la version de Python installée (ex. `Python 3.11`).

> Cela permet à VS Code de savoir quel Python utiliser pour exécuter les programmes.

Créer et exécuter un programme Python

1. Créer un nouveau fichier avec l'extension `.py` (ex. `mon_programme.py`).
2. Écrire du code Python, par exemple :

```
# mon_programme.py
prenom = input("Quel est votre prénom ? ")
age = int(input("Quel est votre âge ? "))
print(f"Bonjour {prenom}, vous avez {age} ans !")
```

3. Exécuter le programme :
    - Clic droit → **Run Python File in Terminal**
    - Ou appuyer sur `F5` pour exécuter avec le débogueur.

Utiliser le terminal intégré

- VS Code possède un terminal intégré (`Ctrl+` ou `Menu → Terminal → New Terminal`).
- On peut exécuter les scripts avec :

```
python mon_programme.py
```

Structure générale d'un programme Python

```
# Ceci est un commentaire (ignoré par Python)

print("Bonjour Python !")  # affichage à l'écran
```

Remarques

- Python s'exécute **ligne par ligne**
- Les **commentaires** commencent par `#`
- L'**indentation est obligatoire** (elle remplace les `{}` d'autres langages)

Variables et types de base

```

```

```
a = 5          # int (entier)
b = 2.5        # float (flottant)
c = "Hello"    # str (chaîne)
d = True       # bool
```

Typage dynamique

- Le type est déterminé automatiquement
- Une variable peut changer de type

```
x = 5
x = "Python"  # valide
```

Connaître le type

```
print(type(a))  # <class 'int'>
```

Lecture et affichage

Entrée utilisateur

```
age = int(input("Quel âge avez-vous ? "))
```

> `input()` retourne **toujours une chaîne**

Conversions courantes :

```
int("12")
float("3.14")
str(25)
```

Affichage avec f-string

```
# A1
prenom = input("Prénom : ")
print(f"Bonjour {prenom} !")
```

Conditions

```
# C1
age = 20
if age >= 18:
    print("Majeur")
elif age >= 13:
    print("Adolescent")
else:
    print("Enfant")
```

Opérateurs logiques

```
and   # ET
or    # OU
not   # NON
```

Exemple :

```
# O1
age = 19
if age >= 18 and age < 65:
    print("Adulte actif")
```

Boucles

Boucle `for`

```
# F1
for i in range(5):
    print(i)
```

> `range(5)` → `0 1 2 3 4`

Autres formes :

```
range(1, 6)     # 1 à 5
range(0, 10, 2) # 0 à 8 avec un pas de 2
```

Boucle `while`

```
# W1
i = 0
```

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

> Attention aux **boucles infinies**

Listes

```
# L1
notes = [12, 15, 18]
```

Accès et modification

```
# L1 (suite)
print(notes[0])
notes[1] = 14
notes.append(16)
```

Parcours

```
# L2
for note in notes:
    print(note)
```

Fonctions utiles

```
# L3
len(notes)
min(notes)
max(notes)
sum(notes)
```

Slicing

```
# S1
notes = [10, 12, 14, 16, 18]
print(notes[1:4])  # [12, 14, 16]
```

Chaînes de caractères

```
# CH1
mot = "Python"
print(mot[0])   # P
print(len(mot)) # 6
```

Chaînes immuables

> Impossible de modifier un caractère directement

```
# mot[0] = "p"  → ERREUR
```

Fonctions

```
# F1
def somme(a, b):
    return a + b
```

Appel :

```
# F2
res = somme(3, 4)
```

Retour multiple

```
# R1
def min_max(l):
    return min(l), max(l)

mn, mx = min_max([3, 1, 4])
print("Min = ",mn)
print("Max = ",mx)
```

Passage par référence

```
# P1
def ajouter(liste):
    liste.append(10)

l = []
ajouter(l)
print(l)  # [10]
```

- Listes et dictionnaires sont modifiés
- Int, float, str ne le sont pas

Dictionnaires

```
# D1
eleve = {
    "nom": "Charles",
    "age": 19,
    "moyenne": 14.5
}
```

Accès :

```
# D2
print(eleve["nom"])
```

Parcours :

```
# D3
for cle, valeur in eleve.items():
    print(cle, valeur)
```

Gestion des erreurs

```
try:
    x = int(input("Nombre : "))
except ValueError:
    print("Erreur de saisie")
```

> Évite que le programme plante

Modules

```
# M1
import math
print(math.sqrt(16))
```

Ou :

```
# MR1
from math import sqrt
print(sqrt(16))
```

Bonnes pratiques

- noms clairs (`age`, `notes`)
- indentation cohérente (tabulation = 4 espaces)
- commentaires utiles
- fonctions courtes
- éviter le code dupliqué

Exercices (avec corrigés)

Exercice 1 — Affichage simple

Énoncé : Écrire un programme qui affiche :

```
Bonjour, je débute en Python !
```

Correction

```
print("Bonjour, je débute en Python !")
```

Exercice 2 — Variables et calcul

Énoncé : Créer deux variables `a` et `b`, puis afficher leur somme, leur différence et leur produit.

Correction

```
a = 10
b = 3

s = a + b
d = a - b
p = a * b

print("Somme :", s)
print("Différence :", d)
print("Produit :", p)
# OU
print("Somme :", a + b)
```

```
print("Différence :", a - b)
print("Produit :", a * b)
```

Exercice 3 — Entrée utilisateur

Énoncé : Demander le prénom et l'âge de l'utilisateur, puis afficher :

```
Tu t'appelles … et tu as … ans.
```

Correction

```
prenom = input("Prénom : ")
age = int(input("Âge : "))

print(f"Tu t'appelles {prenom} et tu as {age} ans.")
```

Exercice 4 — Condition simple

Énoncé : Demander un nombre et afficher s'il est **positif**, **négatif** ou **nul**.

Correction

```
n = float(input("Nombre : "))

if n > 0:
    print("Positif")
elif n < 0:
    print("Négatif")
else:
    print("Nul")
```

Exercice 5 — Boucle `for`

Énoncé : Afficher les nombres de **1 à 10**.

Correction

```
for i in range(1, 11):
    print(i)
```

Exercice 6 — Boucle `while`

Énoncé : Afficher les nombres de **10 à 1**.

Correction

```
i = 10
while i >= 1:
    print(i)
    i -= 1
```

Exercice 7 — Liste

Énoncé : Créer une liste de notes, afficher la moyenne.

Correction

```
notes = [12, 15, 9, 18]
moyenne = sum(notes) / len(notes)
print("Moyenne :", moyenne)
```

Exercice 8 — Recherche dans une liste

Énoncé : Demander un nombre et dire s'il est dans la liste `[3, 6, 9, 12]`.

Correction

```
liste = [3, 6, 9, 12]
n = int(input("Nombre : "))

if n in liste:
    print("Présent")
else:
    print("Absent")
```

Exercice 9 — Fonction

Énoncé : Créer une fonction `carre(n)` qui retourne le carré d'un nombre.

Correction

```
def carre(n):
    return n ** 2

print(carre(5))
```

Exercice 10 — Fonction avec liste

Énoncé : Créer une fonction qui retourne le **maximum** d'une liste.

Correction

```
def maximum(liste):
    return max(liste)

print(maximum([4, 8, 2, 9]))
```

Exercice 11 — Dictionnaire

Énoncé : Créer un dictionnaire contenant le nom et l'âge d'un élève, puis l'afficher.

Correction

```
eleve = {
    "nom": "Carnus",
    "age": 19
}

print(eleve)
```

Exercice 12 — Parcours de dictionnaire

Énoncé : Afficher chaque clé et sa valeur.

Correction

```
for cle, valeur in eleve.items():
    print(cle, ":", valeur)
```

Exercice 13

Énoncé : Écrire un programme qui :

1. Demande 5 notes
2. Les stocke dans une liste
3. Affiche la moyenne
4. Indique si l'élève est admis (moyenne ≥ 10)

Correction

```
notes = []

for i in range(5):
    note = float(input(f"Note {i+1} : "))
    notes.append(note)

moyenne = sum(notes) / len(notes)
print("Moyenne :", moyenne)

if moyenne >= 10:
    print("Admis")
else:
    print("Ajourné")
```

Courbes et graphiques avec Matplotlib

1. Introduction à Matplotlib

Matplotlib est une bibliothèque Python permettant de :

- tracer des **courbes**
- créer des **graphiques** (lignes, barres, points…)
- visualiser des données facilement

Installation (si nécessaire)

```
pip install matplotlib
```

2. Importation de la bibliothèque

On utilise généralement le module `pyplot` :

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

3. Tracer une courbe simple

Exemple de base

```
# E1
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]
```

```
plt.plot(x, y)
plt.show()
```

> `plt.show()` affiche la fenêtre du graphique

4. Ajouter des éléments au graphique

Titres et légendes

```
# E2-
plt.plot(x, y)
plt.title("Exemple de courbe")
plt.xlabel("Axe des x")
plt.ylabel("Axe des y")
plt.show()
```

Couleur, style et marqueurs

```
# E3-
plt.plot(x, y, color="red", marker="o", linestyle="--")
plt.show()
```

Quelques options courantes :

- `color` : `"red"`, `"blue"`, `"green"`
- `marker` : `"o"`, `"x"`, `"+"`
- `linestyle` : `"-"`, `"--"`, `":"`

5. Tracer plusieurs courbes

```
# T1
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]
y2 = [1, 4, 9, 16, 25]

plt.plot(x, y, label="2x")
plt.plot(x, y2, label="x²")
plt.legend()
plt.show()
```

> `label` + `plt.legend()` affichent la légende

6. Graphique avec `range`

```
# G1
import matplotlib.pyplot as plt
x = range(0, 11)
y = []

for i in x:
    y.append(i ** 2)

plt.plot(x, y)
plt.show()
```

7. Nuage de points (scatter)

```
# N1
import matplotlib.pyplot as plt
x = range(0, 11)
y = []

for i in x:
    y.append(i ** 2)
plt.scatter(x, y)
plt.title("Nuage de points")
plt.show()
```

8. Diagramme en barres

```
# D1
import matplotlib.pyplot as plt
noms = ["Admin", "Carnus", "Charles"]
notes = [14, 16, 12]

plt.bar(noms, notes)
plt.title("Notes des étudiants")
plt.show()
```

9. Sauvegarder un graphique

```
# S1
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]
plt.plot(x, y)
```

```
plt.savefig("courbe.png")
plt.show()
```

> Le fichier est enregistré dans le dossier du programme

Courbes avec NumPy et Matplotlib

1. NumPy permet de :

- créer facilement des **tableaux numériques**
- effectuer des **calculs mathématiques sur des listes entières**
- écrire un code **plus court et plus lisible**

> Avec NumPy, pas besoin de boucle `for` pour les calculs simples.

2. Importation des bibliothèques

```
import numpy as np
import matplotlib.pyplot as plt
```

3. Créer des données avec NumPy

`np.array()`

```
x = np.array([1, 2, 3, 4, 5])
```

`np.arange()` **(équivalent de** `range`**)**

```
x = np.arange(0, 11, 1)  # de 0 à 10
```

`np.linspace()` **(très utilisé en math)**

```
x = np.linspace(0, 10, 100)  # 100 points entre 0 et 10
```

> Plus il y a de points, plus la courbe est **lisse**

4. Tracer une courbe simple

Exemple : `y = 2x`

```
# T1
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(0, 11)
y = 2 * x

plt.plot(x, y)
plt.show()
```

- Calcul direct
- Pas de boucle

5. Fonctions mathématiques avec NumPy

NumPy fournit des fonctions mathématiques prêtes à l'emploi :

```
np.sin(x)
np.cos(x)
np.exp(x)
np.sqrt(x)
```

Exemple : fonction carrée

```
# FC1
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-5, 5, 100)
y = x ** 2

plt.plot(x, y)
plt.title("y = x²")
plt.show()
```

6. Tracer plusieurs courbes

```
# TP1
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 100)

plt.plot(x, x, label="y = x")
plt.plot(x, x**2, label="y = x²")
```

```
plt.legend()
plt.show()
```

7. Fonctions trigonométriques

> Les angles sont en **radians**

```
# FT1
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 2*np.pi, 100)

plt.plot(x, np.sin(x), label="sin(x)")
plt.plot(x, np.cos(x), label="cos(x)")
plt.legend()
plt.show()
```

8. Nuage de points avec NumPy

```
# NP1
import numpy as np
import matplotlib.pyplot as plt

x = np.random.rand(20)
y = np.random.rand(20)

plt.scatter(x, y)
plt.show()
```

9. Sauvegarde d'un graphique

```
# S1
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 10)

plt.plot(x, x**2)
plt.savefig("parabole.png")
plt.show()
```

Exercices Matplotlib (avec corrigés)

Exercice 1 — Courbe simple

Énoncé : Tracer la courbe représentant la fonction $y = x$ pour x allant de 0 à 10.

Correction

```
import matplotlib.pyplot as plt

x = range(0, 11)
y = x

plt.plot(x, y)
plt.title("y = x")
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```

Exercice 2 — Fonction carrée

Énoncé : Tracer la courbe de $y = x^2$ pour x de -5 à 5.

Correction

```
import matplotlib.pyplot as plt

x = range(-5, 6)
y = []

for i in x:
    y.append(i ** 2)

plt.plot(x, y)
plt.title("y = x²")
plt.show()
```

Exercice 3 — Deux courbes

Énoncé : Tracer sur le même graphique :

- $y = x$
- $y = x^2$

Correction

```
import matplotlib.pyplot as plt

x = range(0, 6)
```

```
y1 = x
y2 = []

for i in x:
    y2.append(i ** 2)

plt.plot(x, y1, label="y = x")
plt.plot(x, y2, label="y = x²")
plt.legend()
plt.show()
```

Exercice 4 — Nuage de points

Énoncé : Tracer un nuage de points à partir des données :

```
x = [1, 2, 3, 4, 5]
y = [2, 1, 4, 3, 5]
```

Correction

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [2, 1, 4, 3, 5]

plt.scatter(x, y)
plt.show()
```

Exercice 5 — Diagramme en barres

Énoncé : Afficher les notes suivantes sous forme de barres :

```
Maths : 14
Physique : 12
Info : 16
```

Correction

```
import matplotlib.pyplot as plt

matieres = ["Maths", "Physique", "Info"]
notes = [14, 12, 16]

plt.bar(matieres, notes)
```

```
plt.title("Notes")
plt.show()
```

Exercice 6

Énoncé :

1. Demander 5 valeurs à l'utilisateur
2. Les stocker dans une liste
3. Tracer la courbe correspondante

Correction

```
import matplotlib.pyplot as plt

y = []

for i in range(5):
    val = float(input(f"Valeur {i+1} : "))
    y.append(val)

x = range(1, 6)

plt.plot(x, y, marker="o")
plt.title("Courbe des valeurs")
plt.show()
```

Résumé

- `plt.plot()` → courbe
- `plt.scatter()` → points
- `plt.bar()` → barres
- `plt.title()`, `xlabel()`, `ylabel()` → lisibilité
- `plt.show()` obligatoire

Exercices NumPy + Matplotlib (avec corrigés)

Exercice 1 — Fonction linéaire

Énoncé : Tracer la courbe `y = 3x` pour `x` entre 0 et 10.

Correction

```
import numpy as np
import matplotlib.pyplot as plt
```

```
x = np.arange(0, 11)
y = 3 * x

plt.plot(x, y)
plt.title("y = 3x")
plt.show()
```

Exercice 2 — Fonction carrée

Énoncé : Tracer $y = x^2$ pour x entre -5 et 5.

Correction

```
x = np.linspace(-5, 5, 200)
y = x ** 2

plt.plot(x, y)
plt.title("y = x²")
plt.show()
```

Exercice 3 — Deux fonctions

Énoncé : Tracer sur le même graphique :

- $y = x$
- $y = \sqrt{x}$ pour $x \geq 0$

Correction

```
x = np.linspace(0, 10, 100)

plt.plot(x, x, label="y = x")
plt.plot(x, np.sqrt(x), label="y = √x")
plt.legend()
plt.show()
```

Exercice 4 — Sinus

Énoncé : Tracer la courbe du sinus entre 0 et 2π.

Correction

```
x = np.linspace(0, 2*np.pi, 200)
y = np.sin(x)
```

```
plt.plot(x, y)
plt.title("y = sin(x)")
plt.show()
```

Exercice 5 — Cosinus et sinus

Énoncé : Tracer `sin(x)` et `cos(x)` sur le même graphique.

Correction

```
x = np.linspace(0, 2*np.pi, 200)

plt.plot(x, np.sin(x), label="sin(x)")
plt.plot(x, np.cos(x), label="cos(x)")
plt.legend()
plt.show()
```

Exercice 6

Énoncé : Tracer la fonction :

y = x^3 - 3x

pour `x` entre `−3` et `3`.

Correction

```
x = np.linspace(-3, 3, 200)
y = x**3 - 3*x

plt.plot(x, y)
plt.title("y = x³ - 3x")
plt.show()
```

Comparaison sans / avec NumPy

Sans NumPy

```
y = []
for i in x:
    y.append(i**2)
```

Avec NumPy

```
y = x**2
```

> Plus court, Plus lisible, Plus rapide
