

Premiers pas en Python

Université de Lille - L2 Économie

Installer Python et VS Code

1. Installer Python

1. Rendez-vous sur le site officiel : python.org/downloads.
2. Téléchargez la version recommandée (**Python 3.x**, pas Python 2).
3. Lancez le programme d'installation :
 - Sur Windows : cochez la case **Add Python to PATH** avant d'appuyer sur **Install Now**.
 - Sur macOS : ouvrez le fichier **.pkg** et suivez les instructions.
 - Sur Linux : Python est déjà installé dans la plupart des cas (`python3 --version` pour vérifier).
4. Vérifiez l'installation dans un terminal :

```
python3 --version
```

Vous devriez obtenir un numéro de version comme **Python 3.11.6**.

2. Installer Visual Studio Code (VS Code)

1. Téléchargez VS Code sur code.visualstudio.com.
2. Installez-le en suivant les instructions pour votre système.
3. Ouvrez VS Code et installez l'extension officielle **Python** (icône bleue avec un serpent) :
 - Cliquez sur l'icône **Extensions** (à gauche ou **Ctrl+Shift+X**).
 - Cherchez **Python** et installez celle de Microsoft.
4. Facultatif mais utile : installez aussi l'extension **Jupyter** pour travailler avec des notebooks.

3. Premier test

1. Ouvrez un dossier de travail dans VS Code (Fichier → Ouvrir dossier).
2. Créez un fichier `test.py` avec le contenu suivant :

```
print("Bonjour, Python est bien installé !")
```

3. Cliquez sur le bouton Run (triangle vert en haut à droite) ou tapez dans le terminal :

```
python3 test.py
```

Astuce pratique

Si la commande `python3` ne fonctionne pas sous Windows, essayez simplement `python`.

Conseils pour commencer

- Ouvrez VS Code et créez un fichier `tp1.py`.
- Tapez vos instructions dans le fichier puis lancez-le avec le bouton Run.
- Testez aussi directement dans l'interpréteur Python (`>>>`) comme avec une calculatrice.
- Faites beaucoup d'essais, même faux : les messages d'erreur aident à comprendre !

Organisation et outils

Exercice 1 - Créer un dossier de TP

Dans un terminal, tapez :

```
cd ~/Bureau
mkdir Python_L2
cd Python_L2
mkdir TP1 data scripts notes
ls    # (macOS/Linux) ; dir (Windows)
```

Puis ouvrez le dossier dans VS Code :

```
code .
```

- Quelle commande permet de revenir en arrière dans un dossier (`..`) ?
- Quelle est la différence entre `ls` (Linux/macOS) et `dir` (Windows) ?

Exercice 2 - Script vs Interpréteur

1. Créez `scripts/hello.py` contenant `print("Bonjour Python !")`.
 2. Lancez-le dans le terminal : `python3 scripts/hello.py`.
 3. Comparez avec l'interpréteur Python (`python3`) en tapant directement `print("Bonjour")`.
- Quelle est la différence entre exécuter un script et taper une commande dans l'interpréteur ?
 - Dans quel cas préférez-vous utiliser un script plutôt que l'interpréteur ?

Découverte de Python

Exercice 3 - Python comme calculatrice

```
2 + 5
10 - 3
4 * 7
```

```
20 / 6
20 // 6    # division entière
20 % 6     # reste
2 ** 5     # puissance
```

- Quelle différence entre / et // ?
- Que fait l'opérateur % ?
- Quel est le type du résultat de 20/6 ?

Exercice 4 - Ordre des opérations

Prédisez puis testez :

```
2 + 3 * 4
(2 + 3) * 4
```

- Pourquoi les résultats de `2 + 3 * 4` et `(2+3)*4` diffèrent-ils ?
- Comment Python choisit-il l'ordre des opérations ?

Exercice 5 - Chaînes de caractères

```
"Bonjour"
"Bonjour" + " le monde"
"Python" * 3
len("Université de Lille")
```

- Que fait l'opérateur * avec une chaîne ?
- Que renvoie `len("Université de Lille")` ?
- Quelle est la première lettre de "Python" ? Et la dernière ?

Exercice 6 - Créer des variables

```
nom = "Alice"
age = 22
revenu = 1450.75
etudiant = True
```

- Quel est le type de chaque variable ?
- Quelle différence entre un `int` et un `float` ?
- Comment afficher la valeur de toutes vos variables en une seule ligne avec `print(...)` ?

Exercice 7 - Conversions

```
age_str = "22"
print(int(age_str) + 3)
print(str(age) + " ans")
```

- Pourquoi faut-il convertir ?
- Que se passe-t-il si vous essayez `int("3.14")` ?
- Que renvoie `float("3.14")` ?

Exercice 8 - Affichage formaté

```
print(nom, "a", age, "ans")
print(f"{nom} a {age} ans")
```

- Quelle différence entre `print(nom, "a", age)` et `print(f"nom a age")` ?
- Laquelle est la plus pratique si vous avez beaucoup de variables ?

Exercice 9 - Listes

```
notes = [12, 15, 9]
print(notes[0])
notes.append(14)
print(notes)
```

- Que fait `append()` ?
- Peut-on remplacer la première note par 20 ? Comment ?
- Comment calculer la moyenne de la liste avec `sum()` et `len()` ?

Exercice 10 - Indexation de chaînes

```
mot = "Python"
print(mot[0])
print(mot[-1])
print(mot[0:3])
print(mot[::-1]) # inversé
```

- Que renvoie `mot[2:]` ?
- Que fait `mot[::-1]` ?
- Comment afficher uniquement les 3 dernières lettres du mot ?

Exercice 11 - Mise à jour de variable

```
x = 5
x = x + 1
x += 1
```

- Quelle est la valeur finale de `x` ?
- Quelle est la différence entre `x = x + 1` et `x += 1` ?

Exercice 12 - Booléens simples

```
x = 10
print(x > 5)
print(x == 10)
print(x != 8)
```

- Quelle est la différence entre `=` et `==` ?
- Quel est le résultat de `x < 5` quand `x = 10` ?

Exercice 13 - Opérateurs logiques

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

- Quelle est la table de vérité de `and` et `or` ?
- Que fait l'opérateur `not` ?

Exercice 14 - Booléens implicites

```
print(bool(0))
print(bool(42))
print(bool(" "))
print(bool("Python"))
```

- Pourquoi `bool(0)` vaut-il `False` ?
- Que renvoie `bool(" ")` (espace) ?
- Comment Python décide-t-il si une valeur est `True` ou `False` ?

Exercice 15 - Mini-défis

1. Créez deux variables `x` et `y`, affichez somme, différence, produit et quotient.
2. Demandez à l'utilisateur son prénom avec `input()` et affichez "Bonjour ..." avec son prénom.
3. Créez une liste avec 3 prénoms et affichez le deuxième.
4. Essayez `"2" + 3` puis `int("2")+3`. Pourquoi une erreur dans le premier cas ?
 - Que se passe-t-il si vous essayez `"2" * 3` ?
 - Comment afficher la longueur d'une liste de prénoms ?

À retenir

- Terminal $\neq$ Python : le premier exécute des commandes système, le second du code Python.
- Types de base : `int`, `float`, `str`, `bool`.
- Une variable se crée avec `=`.
- Les listes regroupent plusieurs valeurs, les chaînes sont indexées.
- Les booléens servent à comparer et tester des conditions.