Cours de Programmation Python

Niveau Débutant

Département de Formation Professionnelle
Mahrasoft Innovations

8 juillet 2025

Table des matières

1 Introduction à la Formation

Formation 100% Pratique

Bienvenue dans notre formation Python niveau débutant ! Cette formation est conçue pour être **100% pratique**. Chaque concept théorique sera immédiatement mis en application à travers des exercices concrets et des projets réels.
À la fin de cette formation, vous recevrez une **attestation de completion** qui certifiera votre maîtrise des fondamentaux de la programmation Python.

1.1 Objectifs de la Formation

À l'issue de cette formation, vous serez capable de :

- Comprendre les concepts fondamentaux de la programmation
- Maîtriser la syntaxe de base de Python
- Créer des programmes simples et efficaces
- Manipuler les structures de données
- Gérer les erreurs et les exceptions
- Travailler avec les fichiers
- Développer des projets pratiques

1.2 Prérequis

- Aucune expérience préalable en programmation requise
- Connaissance de base de l'utilisation d'un ordinateur
- Motivation et curiosité pour apprendre

1.3 Durée et Organisation

- Durée totale : 6 semaines de formation
- Répartition : 20 modules
- Format : Cours théoriques courts + Exercices pratiques intensifs + Travaux Pratiques + Projets Pratique
- Évaluation : Projets pratiques et quiz

2 Module 1 : Introduction à Python et Installation

2.1 Qu'est-ce que Python ?

Python est un langage de programmation de haut niveau, interprété et polyvalent. Il est reconnu pour sa simplicité et sa lisibilité, ce qui en fait un excellent choix pour les débutants.

Avantages de Python :

— Syntaxe simple et intuitive

— Large communauté et écosystème

— Polyvalence (web, data science, IA, automatisation)

— Gratuit et open source

2.2 Installation de Python

2.2.1 Installation sur Windows

1. Rendez-vous sur `https://www.python.org/downloads/` 2. Téléchargez la dernière version stable 3. Exécutez l'installateur 4. Cochez "Add Python to PATH" 5. Cliquez sur "Install Now"

2.2.2 Vérification de l'installation

Ouvrez l'invite de commande et tapez :

```
python --version
```

2.3 Environnement de Développement

2.3.1 IDLE (Environnement par défaut)

IDLE est l'environnement intégré fourni avec Python. Il comprend :

— Un interpréteur interactif

— Un éditeur de code

— Un débogueur

2.3.2 Alternatives recommandées

— **PyCharm Community** : IDE complet et gratuit

— **Visual Studio Code** : Éditeur léger avec extensions Python

— **Thonny** : Parfait pour débutants

2.4 Premier Programme Python

```
print("Bonjour le monde !")
print("Bienvenue dans la programmation Python")
```

Listing 1 – Hello World en Python

Exercice Pratique 1.1 : Créez un programme qui affiche votre nom et votre âge.

```
nom = "Jean Dupont"
age = 25
print("Je m'appelle", nom)
print("J'ai", age, "ans")
```

Listing 2 – Solution Exercice 1.1

3 Module 2 : Variables et Types de Données

3.1 Les Variables

Une variable est un conteneur qui stocke une valeur. En Python, vous n'avez pas besoin de déclarer le type d'une variable.

```
# Variables num riques
age = 25
prix = 19.99
nombre_complex = 3 + 4j

# Variables textuelles
nom = "Alice"
message = 'Bonjour tout le monde'

# Variables bool ennes
est_majeur = True
est_actif = False
```

Listing 3 – Déclaration de variables

3.2 Types de Données Fondamentaux

3.2.1 Types Numériques

```
# Entiers (int)
nombre_entier = 42
print(type(nombre_entier))  # <class 'int'>

# Nombres    virgule flottante (float)
nombre_decimal = 3.14159
print(type(nombre_decimal))  # <class 'float'>

# Nombres complexes (complex)
nombre_complexe = 2 + 3j
print(type(nombre_complexe))  # <class 'complex'>
```

Listing 4 – Types numériques

3.2.2 Chaînes de Caractères (str)

```
# Diff rentes fa ons de d finir une cha ne
nom = "Python"
langage = 'Programmation'
description = """Python est un langage
de programmation polyvalent"""

# Concat nation
message = "Bonjour " + nom
print(message)  # Bonjour Python

# Formatage de cha nes
age = 25
texte = f"J'ai {age} ans"
```

```python
print(texte)  # J'ai 25 ans
```

Listing 5 – Chaînes de caractères

3.2.3 Booléens (bool)

```python
# Valeurs bool ennes
vrai = True
faux = False

# Op rations bool ennes
result1 = True and False  # False
result2 = True or False   # True
result3 = not True        # False
```

Listing 6 – Valeurs booléennes

3.3 Opérations sur les Variables

```python
# Op rations de base
a = 10
b = 3

addition = a + b         # 13
soustraction = a - b     # 7
multiplication = a * b   # 30
division = a / b         # 3.333...
division_entiere = a // b  # 3
modulo = a % b           # 1
puissance = a ** b       # 1000
```

Listing 7 – Opérations arithmétiques

Exercice Pratique 2.1 : Créez un programme qui calcule l'aire d'un rectangle.

```python
# Calcul de l'aire d'un rectangle
longueur = 12.5
largeur = 8.3

aire = longueur * largeur
perimetre = 2 * (longueur + largeur)

print(f"Longueur : {longueur} cm")
print(f"Largeur : {largeur} cm")
print(f"Aire : {aire} cm ")
print(f"P rim tre : {perimetre} cm")
```

Listing 8 – Solution Exercice 2.1

4 Module 3 : Structures de Contrôle

4.1 Instructions Conditionnelles

4.1.1 if, elif, else

```
age = 18

if age >= 18:
    print("Vous  tes  majeur")
elif age >= 16:
    print("Vous pouvez conduire")
else:
    print("Vous  tes  mineur")
```

Listing 9 – Structure conditionnelle

4.1.2 Opérateurs de Comparaison

```
a = 10
b = 5

print(a == b)  # False ( gal )
print(a != b)  # True (diff rent)
print(a > b)   # True (sup rieur)
print(a < b)   # False (inf rieur)
print(a >= b)  # True (sup rieur ou  gal )
print(a <= b)  # False (inf rieur ou  gal )
```

Listing 10 – Opérateurs de comparaison

4.1.3 Opérateurs Logiques

```
age = 25
salaire = 3000

# ET logique (and)
if age >= 18 and salaire >= 2000:
    print(" ligible  pour le pr t")

# OU logique (or)
if age < 18 or salaire < 1000:
    print("Conditions sp ciales")

# NON logique (not)
if not (age < 18):
    print("Majeur")
```

Listing 11 – Opérateurs logiques

Exercice Pratique 3.1 : Créez un programme qui détermine si un nombre est positif, négatif ou nul.

```
nombre = float(input("Entrez un nombre : "))

if nombre > 0:
```

```
    print("Le nombre est positif")
elif nombre < 0:
    print("Le nombre est n gatif")
else:
    print("Le nombre est nul")
```

Listing 12 – Solution Exercice 3.1

4.2 Boucles

4.2.1 Boucle for

```
# Boucle sur une s quence de nombres
for i in range(5):
    print(f"It ration {i}")

# Boucle sur une cha ne
mot = "Python"
for lettre in mot:
    print(lettre)

# Boucle avec range personnalis
for nombre in range(1, 11, 2):  # de 1    10, pas de 2
    print(nombre)  # 1, 3, 5, 7, 9
```

Listing 13 – Boucle for

4.2.2 Boucle while

```
# Boucle while simple
compteur = 0
while compteur < 5:
    print(f"Compteur : {compteur}")
    compteur += 1

# Boucle while avec condition
nombre = 1
while nombre <= 100:
    print(nombre)
    nombre *= 2  # 1, 2, 4, 8, 16, 32, 64
```

Listing 14 – Boucle while

4.2.3 Contrôle de Boucle

```
# Utilisation de break
for i in range(10):
    if i == 5:
        break
    print(i)  # 0, 1, 2, 3, 4

# Utilisation de continue
for i in range(10):
    if i % 2 == 0:
```

```
        continue
    print(i)  # 1, 3, 5, 7, 9
```

Listing 15 – break et continue

Exercice Pratique 3.2 : Créez un programme qui affiche la table de multiplication d'un nombre.

```
nombre = int(input("Entrez un nombre : "))

print(f"Table de multiplication de {nombre} :")
for i in range(1, 11):
    resultat = nombre * i
    print(f"{nombre}    {i} = {resultat}")
```

Listing 16 – Solution Exercice 3.2

5 Module 4 : Structures de Données

5.1 Listes

5.1.1 Création et Manipulation

```
# Cr ation d'une liste
fruits = ["pomme", "banane", "orange"]
nombres = [1, 2, 3, 4, 5]
mixte = ["Python", 3.14, True, 42]

# Acc s aux    lments
print(fruits[0])     # pomme
print(fruits[-1])    # orange (dernier    lment   )

# Modification
fruits[1] = "fraise"
print(fruits)        # ['pomme', 'fraise', 'orange']

# Ajout d'   lments
fruits.append("kiwi")
fruits.insert(1, "mangue")
print(fruits)        # ['pomme', 'mangue', 'fraise', 'orange', 'kiwi']
```

Listing 17 – Manipulation des listes

5.1.2 Méthodes de Liste

```
nombres = [3, 1, 4, 1, 5, 9, 2, 6]

# M thodes utiles
print(len(nombres))           # 8 (longueur)
print(max(nombres))           # 9 (maximum)
print(min(nombres))           # 1 (minimum)
print(sum(nombres))           # 31 (somme)
print(nombres.count(1))       # 2 (occurrence de 1)

# Tri
nombres.sort()
print(nombres)                # [1, 1, 2, 3, 4, 5, 6, 9]

# Suppression
nombres.remove(1)             # Supprime le premier 1
del nombres[0]                # Supprime par index
element = nombres.pop()       # Supprime et retourne le dernier
```

Listing 18 – Méthodes des listes

5.2 Tuples

```
# Cr ation d'un tuple
coordonnees = (10, 20)
couleurs = ("rouge", "vert", "bleu")

# Les tuples sont immutables
```

```python
# coordonnees[0] = 15  # Erreur !

# D composition de tuple
x, y = coordonnees
print(f"x = {x}, y = {y}")

# Tuple avec un seul    lment
singleton = (42,)  # Attention    la virgule
```

Listing 19 – Tuples

5.3 Dictionnaires

```python
# Cr ation d'un dictionnaire
personne = {
    "nom": "Dupont",
    "prenom": "Jean",
    "age": 30,
    "ville": "Paris"
}

# Acc s aux valeurs
print(personne["nom"])          # Dupont
print(personne.get("age"))      # 30
print(personne.get("email", "Non renseign "))  # Non renseign

# Modification et ajout
personne["age"] = 31
personne["email"] = "jean.dupont@email.com"

# Parcours
for cle, valeur in personne.items():
    print(f"{cle}: {valeur}")
```

Listing 20 – Dictionnaires

5.4 Ensembles (Sets)

```python
# Cr ation d'un ensemble
nombres = {1, 2, 3, 4, 5}
lettres = set("abcdefg")

# Op rations sur les ensembles
ensemble1 = {1, 2, 3, 4}
ensemble2 = {3, 4, 5, 6}

union = ensemble1 | ensemble2           # {1, 2, 3, 4, 5, 6}
intersection = ensemble1 & ensemble2    # {3, 4}
difference = ensemble1 - ensemble2      # {1, 2}
```

Listing 21 – Ensembles

Exercice Pratique 4.1 : Créez un programme de gestion de stock.

```python
# Gestion de stock
stock = {
```

```
    "pommes": 50,
    "bananes": 30,
    "oranges": 25
}

def afficher_stock():
    print("\n--- Stock actuel ---")
    for produit, quantite in stock.items():
        print(f"{produit}: {quantite} unit s")

def ajouter_stock(produit, quantite):
    if produit in stock:
        stock[produit] += quantite
    else:
        stock[produit] = quantite
    print(f"Ajout  {quantite} {produit}")

def vendre_produit(produit, quantite):
    if produit in stock and stock[produit] >= quantite:
        stock[produit] -= quantite
        print(f"Vendu {quantite} {produit}")
    else:
        print("Stock insuffisant")

# Test du programme
afficher_stock()
ajouter_stock("pommes", 20)
vendre_produit("bananes", 10)
afficher_stock()
```

Listing 22 – Solution Exercice 4.1

6 Module 5 : Fonctions

6.1 Définition et Appel de Fonctions

```python
# Fonction simple
def saluer():
    print("Bonjour !")

# Appel de la fonction
saluer()

# Fonction avec param tres
def saluer_personne(nom):
    print(f"Bonjour {nom} !")

saluer_personne("Alice")

# Fonction avec valeur de retour
def addition(a, b):
    return a + b

resultat = addition(5, 3)
print(f"5 + 3 = {resultat}")
```

Listing 23 – Fonctions de base

6.2 Paramètres et Arguments

```python
# Param tres avec valeurs par d faut
def presenter(nom, age=25, ville="Paris"):
    print(f"Je suis {nom}, j'ai {age} ans et j'habite    {ville}")

presenter("Alice")
presenter("Bob", 30)
presenter("Claire", 28, "Lyon")

# Arguments nomm s
presenter(ville="Marseille", nom="David", age=35)

# Fonction avec nombre variable d'arguments
def calculer_moyenne(*nombres):
    if len(nombres) == 0:
        return 0
    return sum(nombres) / len(nombres)

print(calculer_moyenne(10, 15, 20))        # 15.0
print(calculer_moyenne(5, 10, 15, 20, 25)) # 15.0
```

Listing 24 – Paramètres avancés

6.3 Portée des Variables

```python
# Variable globale
compteur_global = 0
```

```
3
4  def incrementer():
5      global compteur_global
6      compteur_global += 1
7      print(f"Compteur global: {compteur_global}")
8
9  def fonction_locale():
10     compteur_local = 10
11     print(f"Compteur local: {compteur_local}")
12
13 incrementer()  # Compteur global: 1
14 fonction_locale()  # Compteur local: 10
```

Listing 25 – Portée des variables

6.4 Fonctions Lambda

```
1  # Fonction lambda simple
2  carre = lambda x: x ** 2
3  print(carre(5))  # 25
4
5  # Utilisation avec map()
6  nombres = [1, 2, 3, 4, 5]
7  carres = list(map(lambda x: x ** 2, nombres))
8  print(carres)  # [1, 4, 9, 16, 25]
9
10 # Utilisation avec filter()
11 nombres = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
12 pairs = list(filter(lambda x: x % 2 == 0, nombres))
13 print(pairs)  # [2, 4, 6, 8, 10]
```

Listing 26 – Fonctions lambda

Exercice Pratique 5.1 : Créez une calculatrice avec des fonctions.

```
1  def addition(a, b):
2      return a + b
3
4  def soustraction(a, b):
5      return a - b
6
7  def multiplication(a, b):
8      return a * b
9
10 def division(a, b):
11     if b == 0:
12         return "Erreur: Division par z ro"
13     return a / b
14
15 def calculatrice():
16     print("=== Calculatrice ===")
17     print("1. Addition")
18     print("2. Soustraction")
19     print("3. Multiplication")
20     print("4. Division")
21
22     choix = input("Choisissez une op ration (1-4): ")
23
```

```
    if choix in ['1', '2', '3', '4']:
        a = float(input("Premier nombre: "))
        b = float(input("Deuxi me nombre: "))

        if choix == '1':
            resultat = addition(a, b)
        elif choix == '2':
            resultat = soustraction(a, b)
        elif choix == '3':
            resultat = multiplication(a, b)
        elif choix == '4':
            resultat = division(a, b)

        print(f"R sultat: {resultat}")
    else:
        print("Choix invalide")

# Test de la calculatrice
calculatrice()
```

Listing 27 – Solution Exercice 5.1