Programmation avancée Python

Chapitre 1

Notions fondamentales & initiation

30 septembre 2025

Aperçu du chapitre 1

Introduction & installation de Python
Systèmes d’exploitation : rôles et exemples
Réseaux informatiques : IP, DNS & Internet
Programmation de base : variables, types, conditions, boucles
Structures de données : listes, tuples, ensembles, dictionnaires
Contrôle de flux, fonctions & modules
I/O, exceptions & programmation objet
Mini-jeu et exercices

[1] [5]

Introduction:

Un langage de programmation est un langage qui sert à décrire les actions qu'un ordinateur doit réaliser. Pour cela, l'ordinateur utilise des programmes. Au début les programmeurs écrivaient des programmes en binaire. Il constitués de suites de 0 et 1 ordonnées de façon précise, afin que le processeur sache quoi faire.

Par exemple 01000010011011110110111001101010011011110111010101110010 signifie **Bonjour.**

Alors discuter en binaire avec un ordinateur peut être très très long!!!

Heureusement pour nous, aujourd'hui les développeurs écrivent leurs programmes sous forme de textes, avec une syntaxe particulière à respecter, mais toujours convertis en langage machine pour exécution.

Pourquoi choisir Python???

Python est un langage de programmation créé par **Guido van Rossum,** dont la première version est sortie en 1991. Ce langage est désormais géré par **la Python Software Foundation** https://www.python.org/psf/.

Le langage Python est un très bon choix pour débuter en programmation car c'est
- un langage de haut niveau, il demande peu de connaissance sur le fonctionnement de l'ordinateur.
- un langage open-source Python est gratuit et son environnement est riche en librairies.
- un langage multiplateforme, Python fonctionne sur toutes les plateformes les plus courantes, Windows, Linux et Mac Os, ainsi que sur des plateformes mobiles telles que Maemo ou Android.
- le langage de programmation le plus facile à apprendre.
La syntaxe du code utilise l'indentation et elle obéit à moins de règles par rapport à d'autres langages (pas d'accolades, crochets, points-virgules, etc...), ce qui facilite la lecture et la compréhension.

Python a une syntaxe qui permet aux développeurs d'écrire des programmes avec moins de lignes.

```python
print("Hello, World!")
```

programme en Python

```c
int main()
{
   printf("Hello, World! \n");
   return 0;
}
```

programme en C

- un langage à typage dynamique, le programmeur n'a pas besoin de déclarer le type des variables.

-**un langage utilisé dans de nombreux domaines:**

-L'administration système,

- Les applications de gestion,

-Les sites et les applications web comme YouTube Microsoft, Instagram…

- Jeux vidéos

Introduction & installation

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

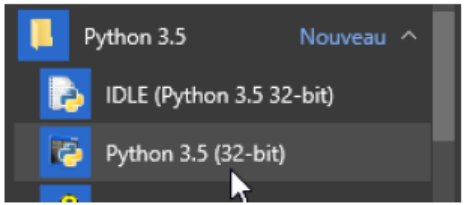
Concepts de base : matériel (CPU, mémoire) & logiciel
Installer Python via python.org ou un gestionnaire de paquets
Modes d’utilisation : interactif (shell) & script (.py)

[24] [25]

Installation de Python

L'installation de Python est assez simple, il suffit de télécharger gratuitement la version qui correspond à votre système d'exploitation à partir du site web officiel à l'adresse: http://www.Python.org.

L'installation comporte le langage en lui-même ainsi qu'un environnement de développement (IDLE).



Comme tout environnement Python comporte une console et un éditeur.
* **La console** (avec l'invite de commande) permet d'exécuter des instructions, des calculs etc.
* **L'éditeur de fichiers** permet de taper le texte d'un programme.

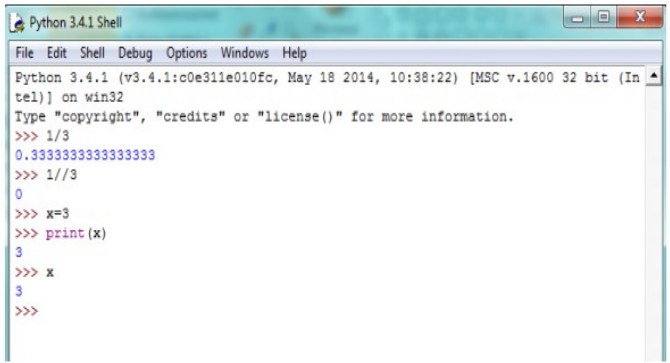
Sous Windows, voici les étapes à suivre :

•Téléchargez le fichier d'installation (avec l'extension .**msi**, compatible avec Windows Installer) correspondant à la configuration de votre ordinateur.

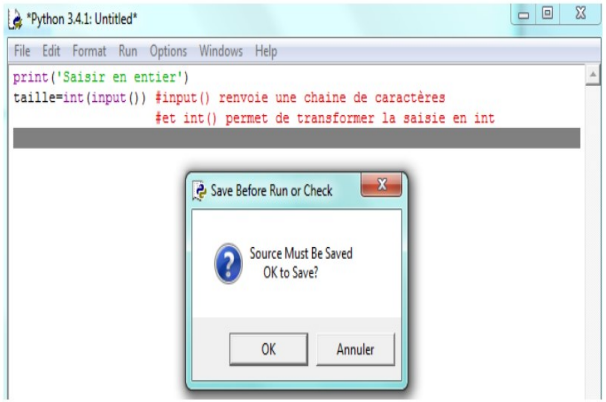
•Lancez l'installation en double-cliquant sur le fichier téléchargé.

L'installation de Python comprend également l'installation d'une interface appelée IDLE (Python GUI). Cette interface permet de saisir des commandes en ligne, ainsi que d'exécuter des programmes Python enregistrés dans des fichiers.

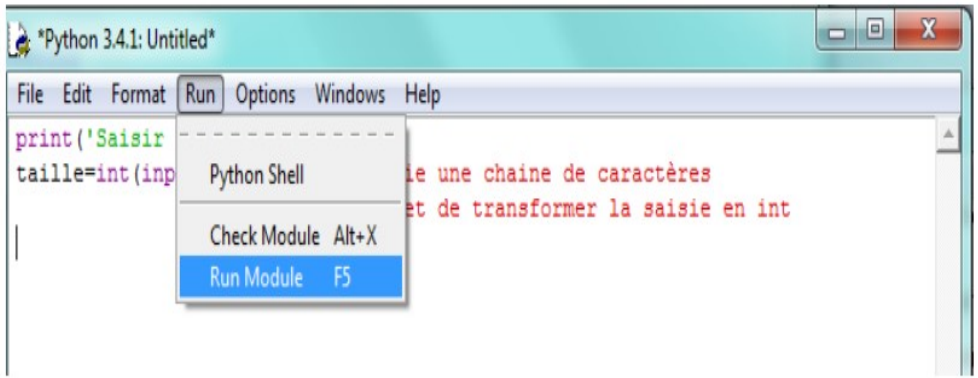
Une fois la dernière version de Python installée, vous pouvez accéder à IDLE depuis le menu Démarrer (dans le dossier Python correspond à la version de Python installée). Il suffit de cliquer sur IDLE (Python GUI)— pour ouvrir l'interface graphique, où vous pourrez entrer vos instructions Python directement dans la ligne de commande.



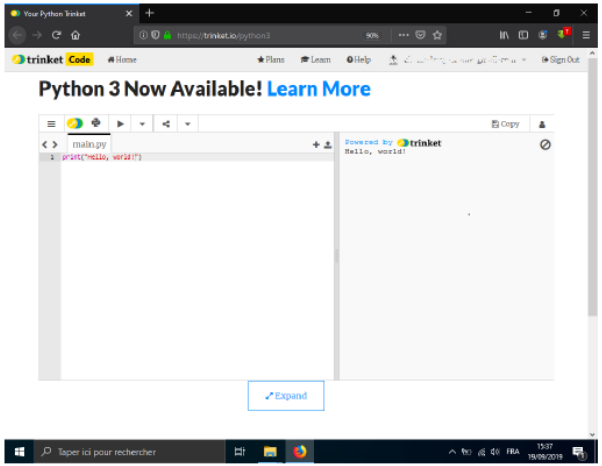
Pour écrire un programme dans un fichier, ouvrez le menu File et sélectionnez New File. Une nouvelle fenêtre s'affichera. Saisissez votre programme Python dans cette fenêtre en veillant à respecter les indentations.

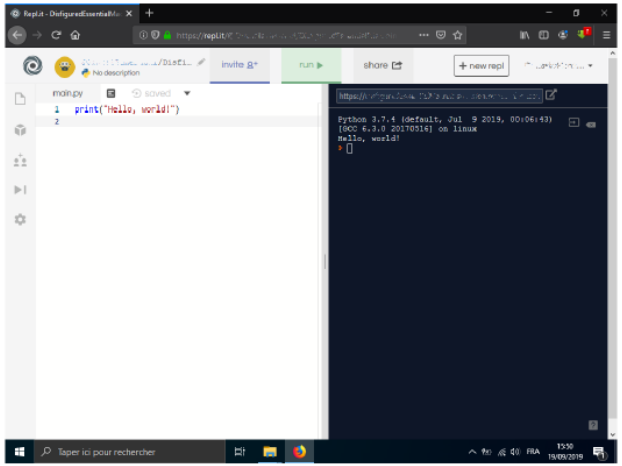


Pour exécuter votre programme, accédez au menu ***Run*** et sélectionnez ***Run Module*** (ou appuyez sur F5). Avant l'exécution, il vous sera demandé d'enregistrer votre fichier, généralement avec l'extension ***.py***. Une fois enregistré, votre programme s'exécutera dans la fenêtre de commande ouverte précédemment.



Différents sites proposent d'exécuter un programme en Python directement depuis un navigateur. Ils permettent de disposer dans un simple navigateur web d'un éditeur et d'un interpréteur Python sans aucune installation. Cela permet notamment aux étudiants de réaliser leurs programmes chez eux sans avoir à installer quoi que ce soit.
Par exemple https://trinket.io/python3 et https://repl.it/languages/python3

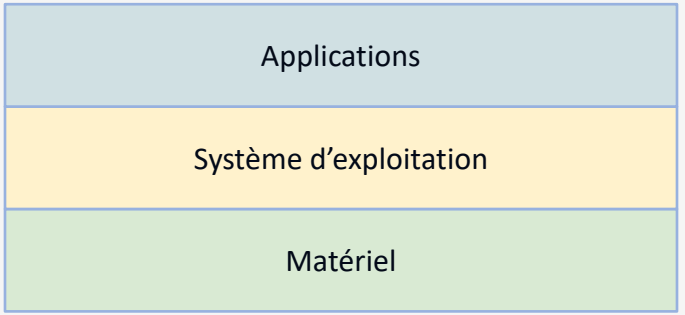




L'interface de programmation est divisée en deux parties :
- à gauche, la fenêtre de script pour écrire des programmes complets qui pourront être sauvegardés;
- à droite, la console dans laquelle le code est exécuté.

Systèmes d’exploitation

Un système d’exploitation est un logiciel essentiel qui agit comme **intermédiaire entre le matériel et les applications**. Sans lui, un ordinateur ou un smartphone ne pourrait pas fonctionner.



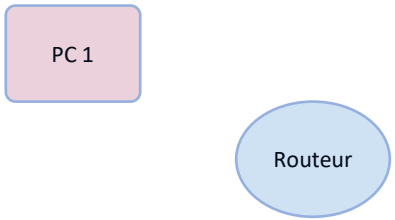
Charge et exécute les programmes
Alloue mémoire, CPU et périphériques aux processus
Gère les interruptions et assure la sécurité
Exemples : Windows, Linux, Android

[21]

Réseaux informatiques

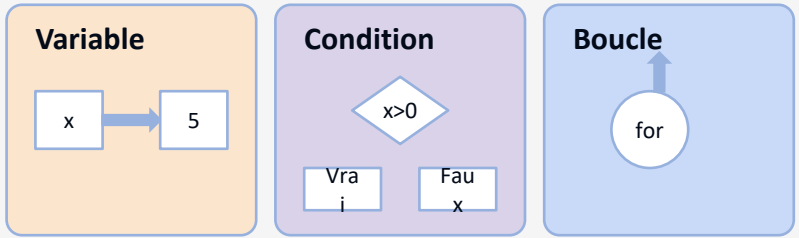
Réseau : ordinateurs interconnectés qui échangent des données
Adresse IP : identifie chaque machine (IPv4 : 4 nombres séparés par des points)
DNS : traduit un nom de domaine en adresse IP mémorable
Exemples : envoyer un e-mail, visiter un site web

```
import socket
print(socket.gethostbyname('python.org'))
```



[22] [23]

Programmation de base



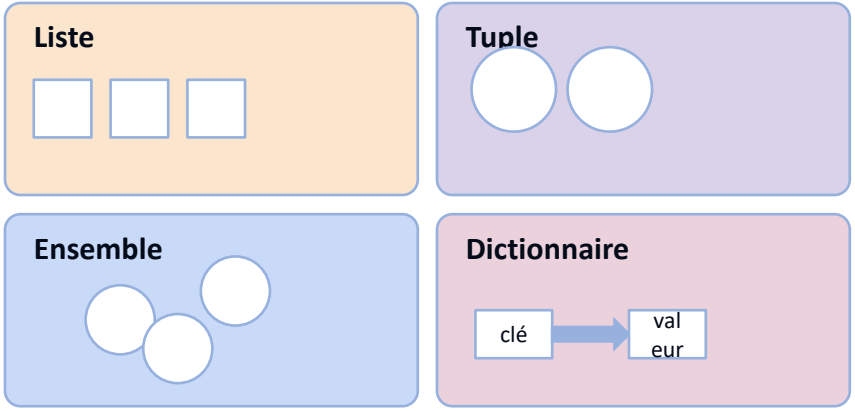
Variables & types : int, float, str
Affectation par =
Opérateurs : + − × ÷ ; booléens and/or/not
Conditions : if/elif/else
Boucles : for, while & range

```
x=5; y=3.0
if x+y>7:

print('Somme haute')
for i in range(3):

print(i)
```

[26] [27] [5] [6] [7]

Structures de données

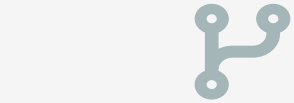
Listes : séquences ordonnées modifiables (append, pop).
Exemple : liste de courses
Tuples : séquences immuables.
Exemple : coordonnées (x, y)
Ensembles : collection d'éléments uniques.
Exemple : fruits sans doublons
Dictionnaires : paires clé-valeur.
Exemple : annuaire {nom : numéro}




```
courses = ['lait', 'pain']
courses
.append('œufs')
prix
= {'lait': 100, 'pain': 50, 'œufs': 120}
print(len(courses), prix['lait'])
```

[1] [2] [3] [4]

Contrôle de flux



if/elif/else : choisir selon une condition
for & range : itérer sur les éléments ou compter
Exemples : choisir un vêtement ; parcourir une liste

```
temperature = 15
if temperature < 20:

print("Il fait froid, prends un manteau.")
else:

print("Temps agréable!")

taches
= ['apprendre', 'manger', 'dormir']
for t in taches:

print(t)
```

[5] [6] [7]

Fonctions & Modules

Fonctions déclarées avec def et paramètres ; première ligne = docstring
Le corps est indenté ; return facultatif (None renvoyé)
Variables locales isolées ; arguments passés par référence d'objet
Module = fichier .py avec définitions ; importer via import/from
Modules standards : math, random pour calculs et nombres aléatoires

```
def prix_avec_taxe(prix, taux=0.19):

"""Calcule le prix TTC."""

return prix * (1 + taux)
print(prix_avec_taxe(100))

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

[8] [9] [10] [28] [29]

Entrées/sorties & Exceptions

f-strings/format() : afficher des valeurs
str() vs repr() : lisible ou exact
open() : modes r/w/a, encodage utf-8
SyntaxError vs exceptions (ZeroDivisionError, NameError)
try/except pour capturer les erreurs

```
with open('data.txt', 'w') as f:
    f
.write('lait\npain')
print(open('data.txt').read())
try:
    n
= int(input('n : '))

print(n * 2)
except ValueError:

print('Erreur!')
```

[11] [12] [13] [14] [15]

Classes & Objets

Regroupe données et fonctions dans un type
Chaque objet a ses attributs/méthodes
Supporte héritage et modifications dynamiques

```
class Personne:

def __init__(self, nom, age):
        self
.nom,self.age=nom,age

def presenter(self):

print(f"{self.nom}, {self.age} ans")
p
=Personne('Ali',30)
p
.presenter()
```

[16]

Mini-jeu : Devine le nombre

Utilise random, boucle while, conditions et exceptions
Devine un nombre 1-10 jusqu'à la réussite
Affiche un indice et compte les essais
Capture ValueError pour entrées non numériques

```
import random
s
=random.randint(1,10); c=0
while True:

try:
        n
=int(input('Devine 1-10: '))

except ValueError:

print('Nombre invalide'); continue
    c
+=1

if n==s: break

print('Trop petit' if n < s else 'Trop grand')
print(f'Bravo en {c} essais')
```

[17] [18] [19]

Résumé & prochaines étapes

Les structures (listes, tuples, ensembles, dictionnaires) organisent vos données.
Les conditions, boucles et fonctions permettent de construire des algorithmes clairs.
Les modules, l'I/O et la gestion des exceptions rendent vos programmes robustes et modulaires.
Les classes préparent la programmation orientée objet.
Explorez des bibliothèques : math, random, NumPy, Pandas et pratiquez via des exercices.
Pour progresser, amusez-vous avec des mini-projets !

[20]