

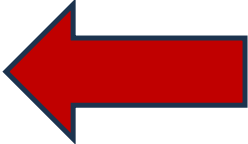
Initiation à la Programmation

Jour 1 : Fondamentaux

Programmation & Langages

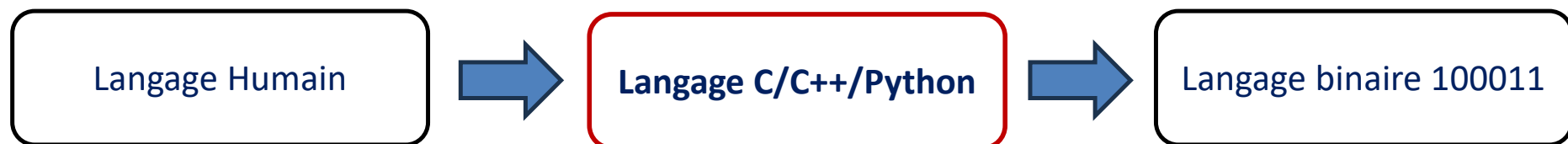
Programmation, Programmeur, Programme & Langage

Programmation

- **Qu'est-ce que la programmation ?**
 - Procédé consistant, à partir d'une idée, à **conceptualiser et rédiger des algorithmes** dans un **langage de programmation**
 1. Analyse du problème
 2. Choix des solutions et algorithmes et description sous forme de diagrammes
 3. Ecriture du code 
 4. Test du code et analyse des résultats produits
 5. Maintenance du programme

Programmation

- Qu'est-ce que le rôle d'un programmeur?
 - Donner des ordres à une machine pour qu'elle fasse une suite d'actions utiles dans un ordre précis
 1. Machine : parle uniquement le binaire < > Langage humain
 2. Langage : ➔ Le programmeur doit apprendre des **langages intermédiaires pour communiquer** avec elle : C++, Swift, PHP, Javascript, Python...
 3. Compilation/Interprétation : Ces langages sont plus proches du binaire mais ne sont toujours pas compréhensibles par votre ordinateur ou votre smartphone ➔ Traduction en code binaire

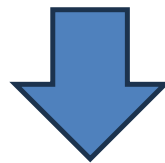


Programmation

- Qu'est-ce qu'un programme informatique ?

- Un programme informatique est un **ensemble d'instructions et opérations** qui doivent être exécutées par un ordinateur

- **Programme source:** Code écrit dans un langage de programmation. Il peut être compilé ou interprété.



- **Programme binaire:** Ensemble d'instructions en langage machine à exécuter par un microprocesseur

Programmation

- Qu'est-ce qu'un langage de programmation ?
 - Un langage de programmation est un **vocabulaire et un ensemble de règles** servant à instruire un ordinateur et effectuer des tâches spécifiques
 - Simplement du texte avec une **syntaxe** particulière

```
# Cryptage de César
def cryptage_cesar(chaine_a_crypter: str, cle_privee: int) -> str:
    # on prend chaque lettre une par une
    resultat_cryptage = list()
    for lettre_originale in chaine_a_crypter:
        lettre_originale_ascii = ord(lettre_originale) # string vers entier ascii
        lettre_cryptee = chr(lettre_originale_ascii + cle_privee) # entier ascii vers string
        resultat_cryptage.append(lettre_cryptee) # sauvegarde de la lettre qu'on vient de crypter
    return ''.join(resultat_cryptage) # Permet de fusionner la liste en 1 seule str

def decryptage_cesar(chaine_a_decrypter: str, cle_privee: int) -> str:
    return cryptage_cesar(chaine_a_decrypter, -cle_privee)
```

Outils nécessaires

Outils nécessaires

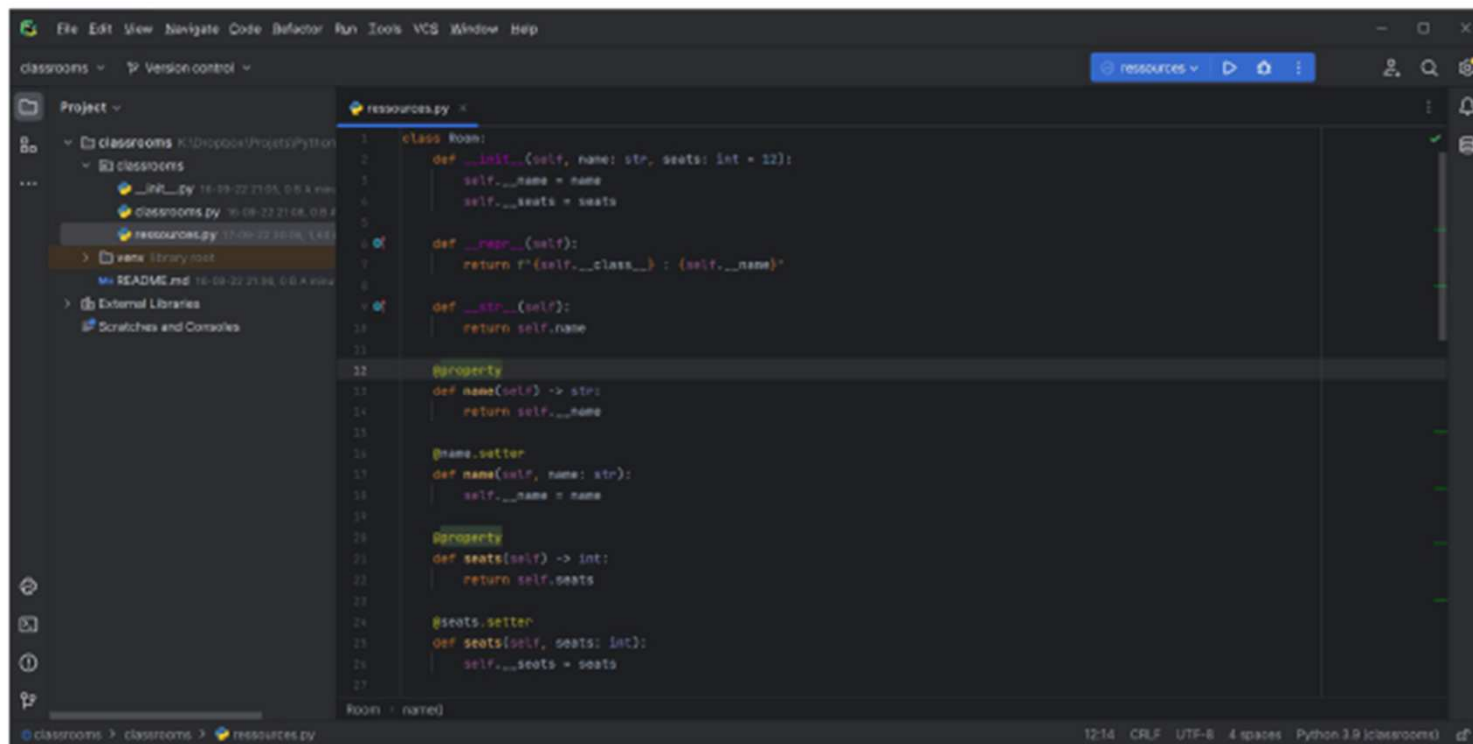
- **Environnement de développement intégré (IDE)**
 - Ensemble d'outils permettant d'améliorer la productivité du développeur informatique
 - Principaux outils:
 - Éditeur de texte brut
 - Fonctions de démarrage de la compilation/interprétation/debugging/exécution
 - Aides à la création du code (complétion automatique, suggestions, ...)
 - Contrôle de version (ex: Git)

Environnement de développement intégré (IDE)

- **Pour programmer en Python : PyCharm**



- Développé par JetBrains
- Disponible sous Windows, Linux et MacOS
- Version « Community » gratuite
- Principalement pour le développement en langage Python

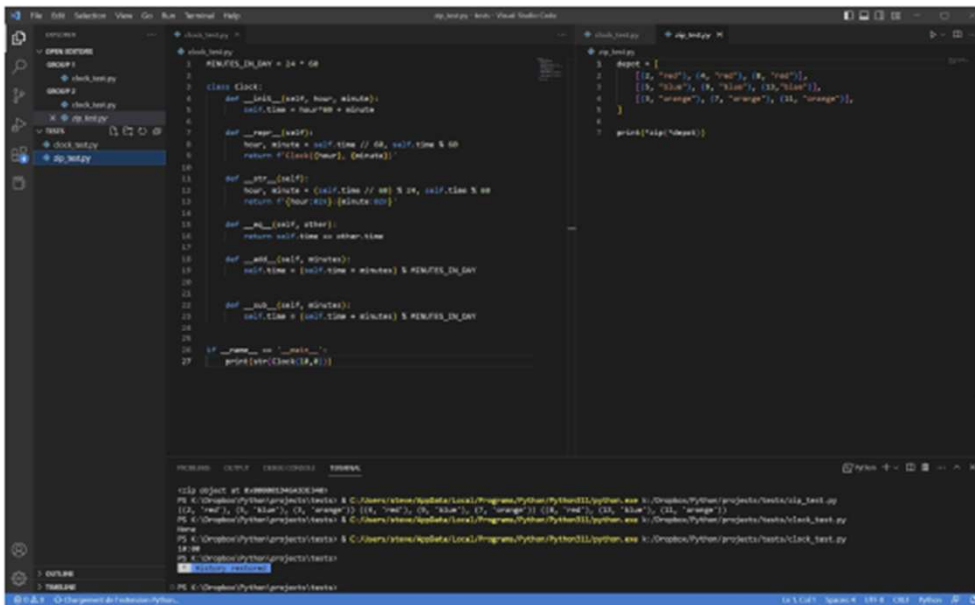


Environnement de développement intégré (IDE)

- Pour programmer en Python, C, C#, C++ ...



Exemple d'IDE: Visual code Studio

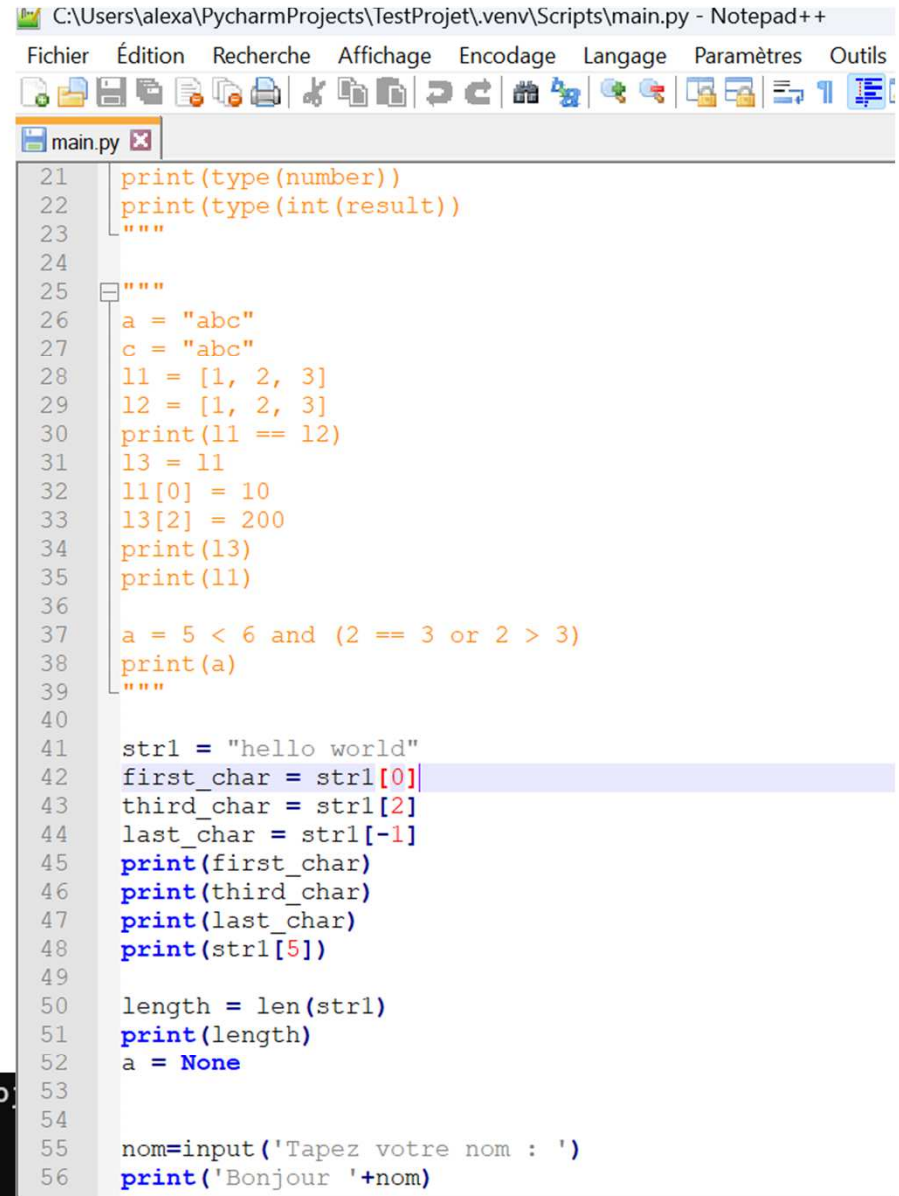


- Développé par Microsoft
- Disponible sous Windows, Linux et MacOS
- Supporte une multitude de langages (Python, C, C++, CSS, C#, PHP, Java, ...)

Sans IDE

- **Python**

- **Editeur de texte** pour coder le fichier .py
- **Terminal « Konsole »** pour l'exécuter



```
C:\Users\alexa\PycharmProjects\TestProjet\venv\Scripts\main.py - Notepad++
Fichier  Édition  Recherche  Affichage  Encodage  Langage  Paramètres  Outils

main.py x
21 print(type(number))
22 print(type(int(result)))
23 """
24
25 """
26 a = "abc"
27 c = "abc"
28 l1 = [1, 2, 3]
29 l2 = [1, 2, 3]
30 print(l1 == l2)
31 l3 = l1
32 l1[0] = 10
33 l3[2] = 200
34 print(l3)
35 print(l1)
36
37 a = 5 < 6 and (2 == 3 or 2 > 3)
38 print(a)
39 """
40
41 str1 = "hello world"
42 first_char = str1[0]
43 third_char = str1[2]
44 last_char = str1[-1]
45 print(first_char)
46 print(third_char)
47 print(last_char)
48 print(str1[5])
49
50 length = len(str1)
51 print(length)
52 a = None
53
54
55 nom=input('Tapez votre nom : ')
56 print('Bonjour '+nom)
```

```
PS C:\Users\alexa> python C:\Users\alexa\PycharmPro
h
l
d

11
Tapez votre nom : Alex
Bonjour Alex
PS C:\Users\alexa> |
```

Langages

Langages

- **Langages courants**

- C, C++, C#
- Java
- PHP
- Python

- **Vieux langages**

- Fortran
- Cobol

Langages

- Importance de la syntaxe
 - « Hello World! » à travers les langages

Langage C

```
#include <stdio.h>

main()
{
    printf("Hello World!\n");
}
```

Langage BASIC

```
10 PRINT "Hello World!"
```

Langage JavaScript

```
console.log("Hello World");
```

Langage Go

```
package main

import "fmt"

func main() {
    fmt.Printf("Hello World\n")
}
```

Langages

- Importance de la syntaxe
 - « Hello World! » à travers les langages

Langage Assembleur Intel

```
mov ax,cs
mov ds,ax
mov ah,9
mov dx, offset Hello
int 21h
xor ax,ax
int 21h

Hello:
    db "Hello World!",13,10,"$"
```

Langage Python 3

```
print("Hello World")
```

Langage Java

```
class HelloWorld {
    static public void main( String args[] ) {
        System.out.println( "Hello World!" );
    }
}
```

Langage PHP

```
<?php echo 'Hello World!'; ?>
```

Langages

- **Langages compilés vs langages interprétés**

Langage compilé:

- Utilise un compilateur
- Transforme code écrit en langage machine
- Réorganise et optimise le code
- Code doit être compilé pour être exécuté

Exemples: C, C++, Go

Langage interprété:

- Utilise un interpréteur
- Interprète et exécute le code à la volée
- Pas de compilation

Exemples: Python, PHP, JavaScript

Langages

- **Pour la compilation : un compilateur**
 - La **compilation** se passe entièrement sur l'ordinateur du développeur
 - **Compilateur**
 - fourni par les créateurs du langage
 - reçoit des fichiers texte dans lequel le programmeur a listé les ordres à donner dans un langage de programmation compatible avec ce compilateur
 - traduit ces fichiers contenant du texte en un seul exécutable (fichier.exe sous Windows) qui ne contient que du code binaire
 - Le programmeur envoie le fichier exécutable à ses utilisateurs qui peuvent l'utiliser immédiatement

Langages

- **Compilation**

- **Avantages de la compilation**

- Performances élevées de l'app' chez l'utilisateur
 - Confidentialité des sources : les utilisateurs finaux ne verront que le code binaire

- **Inconvénients de la compilation**

- Chaque modification de code doit entraîner une recompilation de l'app et un envoi de nouvel exécutable aux utilisateurs
 - L'utilisateur ne pourra jamais modifier l'app, même s'il est programmeur. Il lui faut les codes sources.

Langages

- **Pour l'interprétation : un interprète/interpréteur**
 - Traduit en temps réel le code source, en instructions à suivre
 - Chaque utilisateur reçoit le **code source** de l'app (et non un exécutable) et il doit installer l'interprète sur sa machine

➔ **Compilation <> Interprétation :**

- La compilation traduit une seule fois en amont chez le développeur
- <> l'interprétation va traduire en temps réel à chaque fois que l'app sera lancée chez l'utilisateur

Langages

- **Interprétation**

- **Avantages de l'interprétation**

- L'utilisateur peut modifier les codes source de l'app
 - Légèrement plus simples à prendre en main que des langages compilés

- **Inconvénients de l'interprétation :**

- Les performances seront moins bonnes qu'une app compilée
 - Les codes sources originaux sont visibles et modifiables par les utilisateurs

Langages

- **Doit-on apprendre tous les langages ?**
 - **Non**
 - **Principes similaires**
 - Opérations mathématiques,
 - Variables,
 - Boucles d'itération (for/while),
 - Conditions, Fonctions
 - **Différences : La syntaxe**
- ➔ Lorsque l'on connaît un langage, les autres s'apprennent beaucoup plus vite

Langages

- **Importance de la syntaxe**
 - Permet au compilateur ou à l'interpréteur de transformer votre texte en code binaire pour la machine
 - Une erreur de syntaxe \Leftrightarrow La machine ne comprendra pas quoi faire
 - ➔ Importance du débogueur qui nous permet de détecter nos erreurs de code

Métiers en programmation

Métiers orientés développement informatique

- **Analyste fonctionnel**
 - Concevoir le fonctionnement d'une application
 - Analyser les besoins des utilisateurs
 - Formaliser les besoins et formaliser une solution adaptée

Métiers orientés développement informatique

- **Développeur-euse informatique**
 - Responsable de la programmation, de la production du code informatique
 - **Différentes catégories**
 - Développement d'applications
 - Développement d'applications mobiles
 - Développement Web
 - Développement de jeux vidéos
 - Développement Front-/Back-End/Full Stack

Métiers orientés développement informatique

- **Front-End / Back-End d'un site web**
 - **Partie Front-End : « Interface utilisateur » ⇔ ce que l'on voit**
 - Lorsque l'on atterrit sur la page d'un site web, on peut interagir avec ce que l'on appelle "l'interface utilisateur"
 - ➔ cliquer sur des liens, scroller la page de haut en bas, remplir des formulaires, naviguer entre les onglets...
 - ➔ toute cette partie visuelle, c'est le front-end
 - **Partie Back-End ⇔ Coulisses de l'application**
 - Une fois que l'on a cliqué quelque part, cette action de "clic" est perçue comme une demande par le site : la "demande" envoyée par l'utilisateur est reçue par le site, qui va chercher l'information demandée, et va la renvoyer à l'utilisateur : c'est le back-end.
 - ➔ Le back-end, c'est toute la partie que l'utilisateur ne voit pas, mais qui lui permet de réaliser des actions sur un site ou une application.

Vocabulaire

- Code (ou script)
- Compiler
- Interpréter
- Débugger
- IDE
- Console
- Prompt