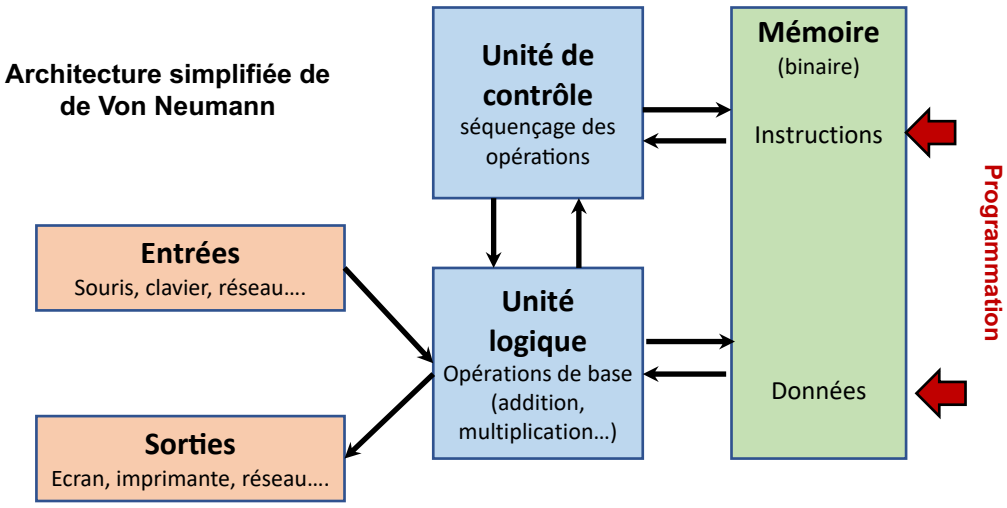
Le fonctionnement d'un ordinateur

1

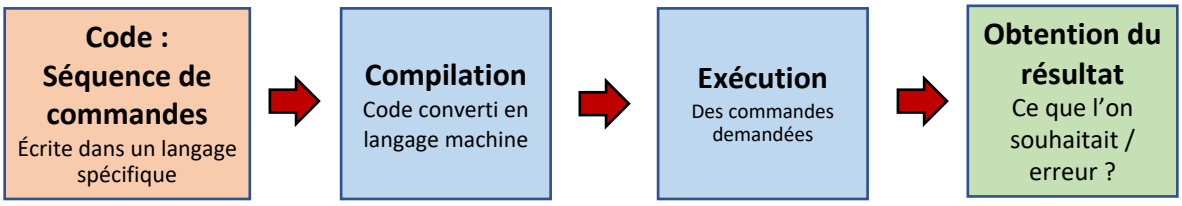


Langage de programmation

2

- Suite d'instructions (opérations de base: addition, soustraction, multiplication…) que le programmeur demande à l'ordinateur d'exécuter
- Les instructions sont lues de façon séquentielle (les unes après les autres)

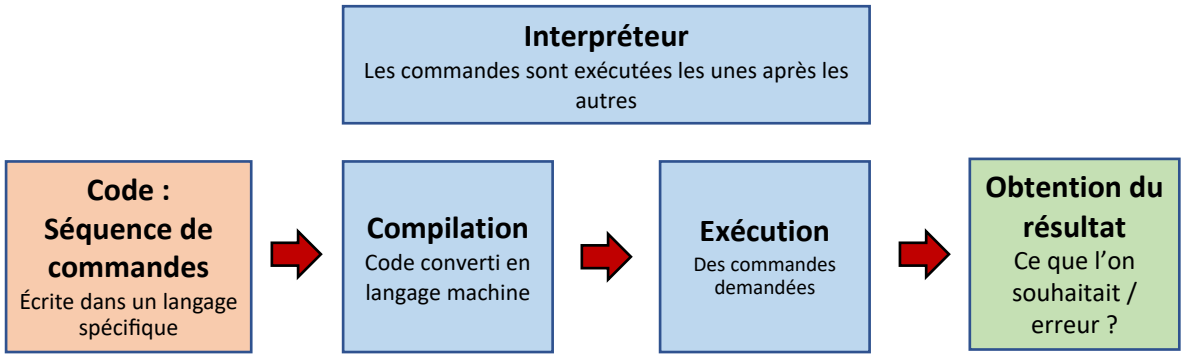
⇒ **Programmeur doit :**
- **Apprendre les commandes de base**
- **Savoir assembler ces commandes**



Langage de programmation

3

- Différents langages (C, C++, python, Java, html, php…)



Python, c'est quoi ?

4

- **Inventé en 1991**
- **un langage interprété (pouvant être compilé)**
- **Très utilisé, facile, puissant…**
- **Beaucoup de librairies (fonctions déjà existantes)**

Premier pas, l'interpréteur une super calculette!

1 Opérations élémentaires

1.1 Addition, soustraction

```
In [97]: # Addition
         1 + 3
```

```
Out[97]: 4
```

```
In [98]: # Soustraction
         2 - 4
```

```
Out[98]: -2
```

1.2 Multiplications

```
In [26]: # Multiplication
         10 * 5
```

```
Out[26]: 50
```

```
In [27]: #Puissance
         10 ** 5
```

```
Out[27]: 100000
```

```
In [28]: # 10^5
         1e5
```

```
Out[28]: 100000.0
```

1.3 Divisions

```
In [29]: # Division
         10 / 3
```

```
Out[29]: 3.3333333333333335
```

```
In [30]: # Division entière
         10 // 3
```

```
Out[30]: 3
```

```
In [31]: # Reste de la division entière
         10 % 3
```

```
Out[31]: 1
```

1.4 Comparaisons

```
In [32]: 10 > 5

Out[32]: True

In [33]: 10 < 5

Out[33]: False

In [34]: 10 <= 5

Out[34]: False

In [35]: 10 >= 5

Out[35]: True

In [36]: 10 == 5

Out[36]: False

In [37]: 10 != 5

Out[37]: True
```

2 Affectation

On accède à une donnée dans la mémoire grâce à un NOM que l'on choisit (ex. age). Pour stocker une données, on utilise l'opérateur =

Attention le = ne correspond pas au = (égalité) des mathématiques. Pour tester si deux variables sont égales, il faut utiliser ==

```
In [38]: # Affectation
         age = 20

In [39]: # Vérification
         age

Out[39]: 20
```

Il est possible de modifier la valeur de la variable **(de façon irréversible)**

```
In [40]: # Réaffactation
         age = 40

In [41]: # Vérification
         age

Out[41]: 40
```

3 Opérateurs pratiques sur les variables

3.1 Permutation

```
In [42]: a = 5
         b = 10
         a,b = b,a

In [43]: a

Out[43]: 10

In [44]: b

Out[44]: 5
```

3.2 Incrémentations

```
In [59]: a = 1

In [60]: # Incrémentation simple
         a = a + 1
         a

Out[60]: 2

In [61]: # Incrémentation condensée
         a += 1
         a

Out[61]: 3

In [62]: # Rentracher une valeur
         a-=1
         a

Out[62]: 2

In [63]: # Multiplier par une valeur
         a *= 10
         a

Out[63]: 20

In [64]: # Diviser par une valeur
         a /= 2
         a

Out[64]: 10.0
```

4 Les types de variables

4.1 Les nombres entiers (int)

```
In [65]: a = 10
         b = -15
         c = 3e8
```

4.2 Les nombres réels (float)

```
In [66]: a = 3.14159
         b = -12.4e-5
         c = 3.
```

4.3 Les chaînes de caractères (str)

```
In [68]: phrase = "Il fait beau !"
         phrase2 = """Il fait beau !"""
```

4.4 Les bouléens (bool)

```
In [69]: test = True
         test2 = False
         test3 = 3 > 4
```

Attention lorsque l'on fait des opérations sur des variables de type différent. Ex : integer + string ?

```
In [70]: a = 10
         b = "toto"
         a + b
```

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-70-04ab7bb24f21> in <module>
  1 a = 10
  2 b = "toto"
----> 3 a + b


TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

5 Les fonctions standards

Fonction : *suite d'instruction déjà enregistrées. pour l'exécuter, il faut connaitre son nom et lui donner les arguments (informations) nécessaires*

```
nom_de_la_fonction(argument1, argument2,...
```

Librairie : *ensemble de fonctions prêtes à être utilisées (déjà compilées)*

5.1 La fonction `print(variable)`

```
affiche la valeur d'une variable!
```

```
In [71]: phrase = "Il fait beau"
         print(phrase)
```

```
Il fait beau
```

```
In [72]: suite = "aujourd'hui !!!"
         print(phrase,suite)
```

```
Il fait beau aujourd'hui !!!
```

5.2 La fonction `type(variable)`

```
affiche le type d'une variable
```

```
In [73]: a = 10
         type(a)
```

```
Out[73]: int
```

5.3 La fonction `help(nom_de_la_fonction)`

```
affiche l'aide (en anglais) de la fonction!
```

Avec jupyter-notebook, nous pouvez aussi utiliser ?

```
In [81]: help(print)
```

```
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
```

```
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

```
In [82]: ?print
```

5.4 La fonction `exit()`

pour quitter python

On ne va pas le faire dans jupyter-notebook

5.5 Les fonctions pour changer le type d'une variables `float(variable)`, `int(variable)`, `str(variable)`

```
In [84]: a = 10.5
         b = int(a)
         print(b)
         type(b)
```

```
10
```

```
Out[84]: int
```

```
In [85]: a = 1000.5
         b = str(a)
         print(b)
         type(b)
```

```
1000.5
```

```
Out[85]: str
```

5.6 La fonction `input()`

Demande à l'utilisateur d'entrer une valeur

```
In [86]: phrase = input()
```

```
Bonjour à tous.
```

```
In [87]: print(phrase)
```

```
Bonjour à tous.
```

```
In [88]: note = input()

10


In [89]: type(note)

Out[89]: str

In [90]: note = int(input())

10


In [91]: type(note)

Out[91]: int

In [92]: note = input("Entrer notre note :")

Entrer notre note :10
```

5.7 Fonction mathématiques usuelles

Pour avoir les fonctions mathématiques usuelles dans python, il est nécessaire de charger une bibliothèque externe. Nous allons utiliser la librairie numpy déjà installée.

Pour charger la librairie

```
In [93]: from numpy import *
```

On peut alors appeler les fonctions mathématiques classiques (sqrt, cos, sin, tan, log...)

```
In [94]: tan(1.23)

Out[94]: 2.8198157342681518

In [95]: exp(10)

Out[95]: 22026.465794806718

In [96]: arcsin(0.2)

Out[96]: 0.2013579207903308
```

Attention pour les fonctions trigonométriques, il faut utiliser les radians