

# Informatique 1

## ST: semestre 1

Dr. Amir DJOUAMA

[djouama.amir@gmail.com](mailto:djouama.amir@gmail.com)

# Volume horaire

- 1 heure 30 minutes de cours
- 1 heure 30 minutes de TP

# Évaluation

- Deux contrôles continus
  - Des interrogations (courte durée)
  - Examen final
  - Rattrapage
- 
- $MoyGle = 0,6 \times Examen + 0,4 \times AutreMoy$
  - $AutreMoy = MoyCC(/15) + MoyenneInterro(/5)$

➔ Notes non négociables

# Règles à suivre

- Horaires à respecter
- Pas un mot dès que le cours commence
  - Sauf pour des questions
  - Le respect de chacun
- Présence obligatoire en TP, émargement pendant le cours
- TP
  - Réaliser en salle de lecture
  - Respect de la machine
  - Attention aux mot de passe

# Programme

- Chapitre 1
  - Introduction à l'informatique
- Chapitre 2
  - Notions de base de la programmation

# Chapitre 1

Introduction à l'informatique

# Chapitre 1

## Introduction à l'informatique

- Définitions de base : Informatique, Ordinateur, Information.
- Système informatique
  - Le Hardware
    - Structure et fonctionnement d'un ordinateur
      - Architecture d'un ordinateur
      - Les catégories d'ordinateurs
      - Le fonctionnement général d'un ordinateur
      - Le codage des informations
  - Le software
    - Définitions de base : Instruction, Programme, Langage, Logiciel, Progiciel.
    - Apprentissage des systèmes d'exploitation
      - MS-DOS
      - Windows
      - Linux... etc.

# Informatique

- *INFOR*mation auto*MATIQUE*

# Informatique

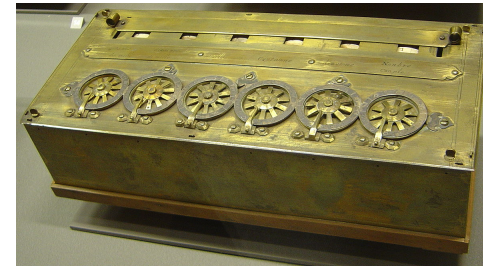
- La science de traitement automatique de l'information
  - (automatiser l'information que nous manipulons).
- Cette informatisation permettra de réaliser un gain considérable en temps et en effort.
- Informatique
  - La conception et la construction des ordinateurs,
  - Le fonctionnement et la maintenance des ordinateurs,
  - Leur exploitation (utilisation des ordinateurs dans les différents domaines d'activités).

# Ordinateur

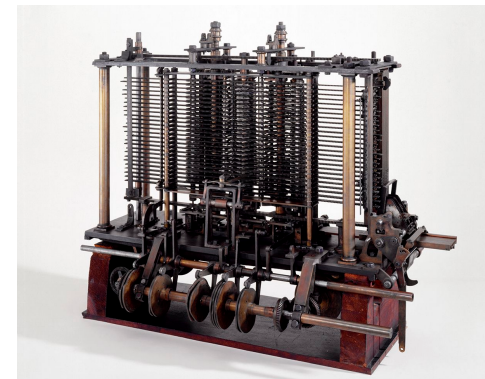
- Est une machine automatique de traitement de l'information.
- Il peut recevoir des données en entrée,
  - « fonction d'entrée »,
- Effectuer sur ces données des opérations en fonction d'un programme, « fonction de traitement »
- Et enfin fournir des résultats en sortie,
  - « fonction de sortie ».

# Évolution de l'informatique et des ordinateurs

- Pascaline 1643 (Blaise pascal) - machine mécanique capable de réaliser des additions et soustractions.

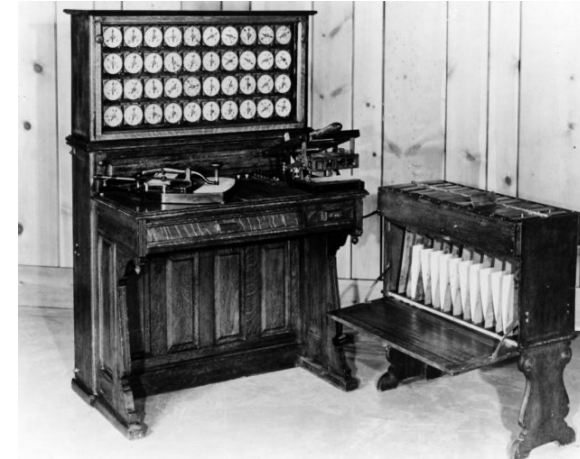


- 1812, "CHARLES BABBAGE". Une machine mécanique pour effectuer des calculs numériques compliqués (c'est une machine à base de cartes perforées).

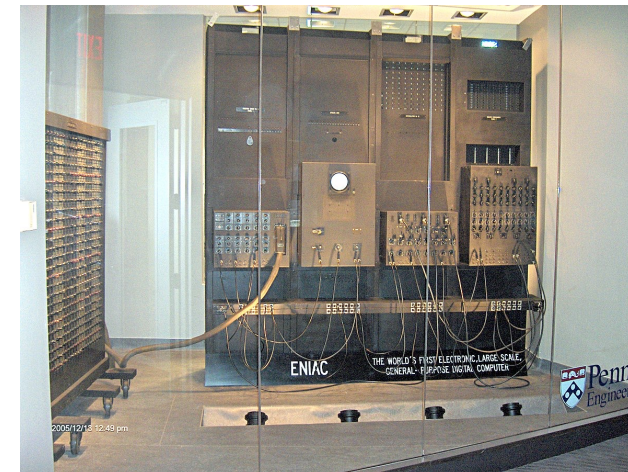


# Évolution de l'informatique et des ordinateurs

- 1885, "HERMAN HOLLERITH" (inventeur des cartes perforées) construit la première machine à cartes perforées et qui a servit dans l'opération de recensement de la population d'Amérique en 1890.



- 1946, ENIAC (Electronic Numerical Integrator and Computer). Construit sur la base du binaire



# Évolution de l'informatique et des ordinateurs

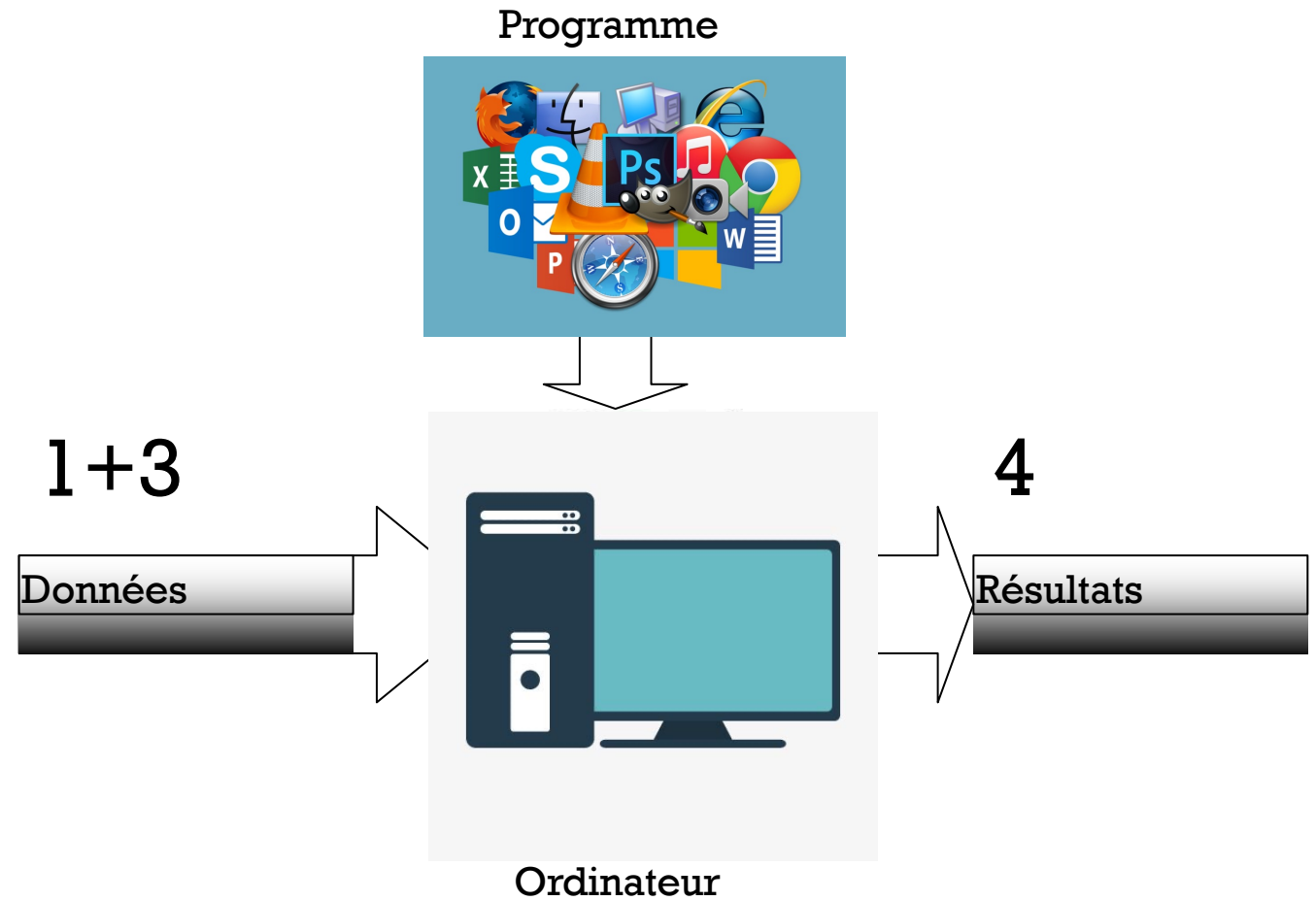
- IBM 701 (1952)
- PDP-1 (1960)
- IBM 7030 (1961)
- Altair 8008 (1973)
- Apple 1 (1976)
- IBM PC (1981)
- Pentium



# Schéma d'une machine



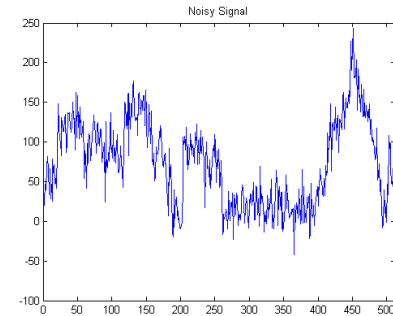
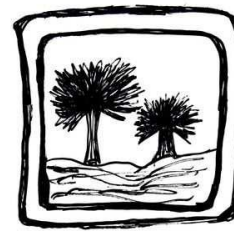
# Schéma d'une machine



# Information

- Est un ensemble d'événements qui peuvent être **communiqué** à l'ordinateur

TEXT



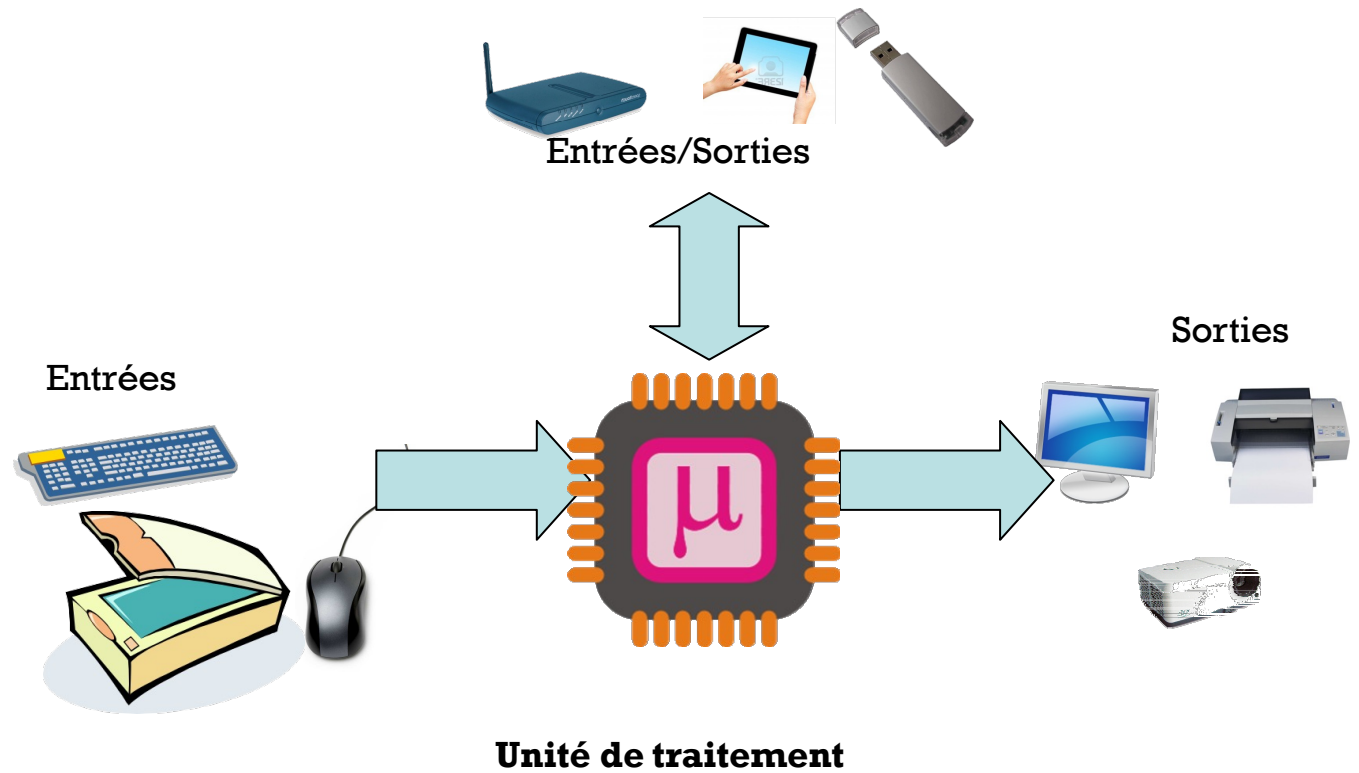
# Systeme d'information

# Systeme d'information

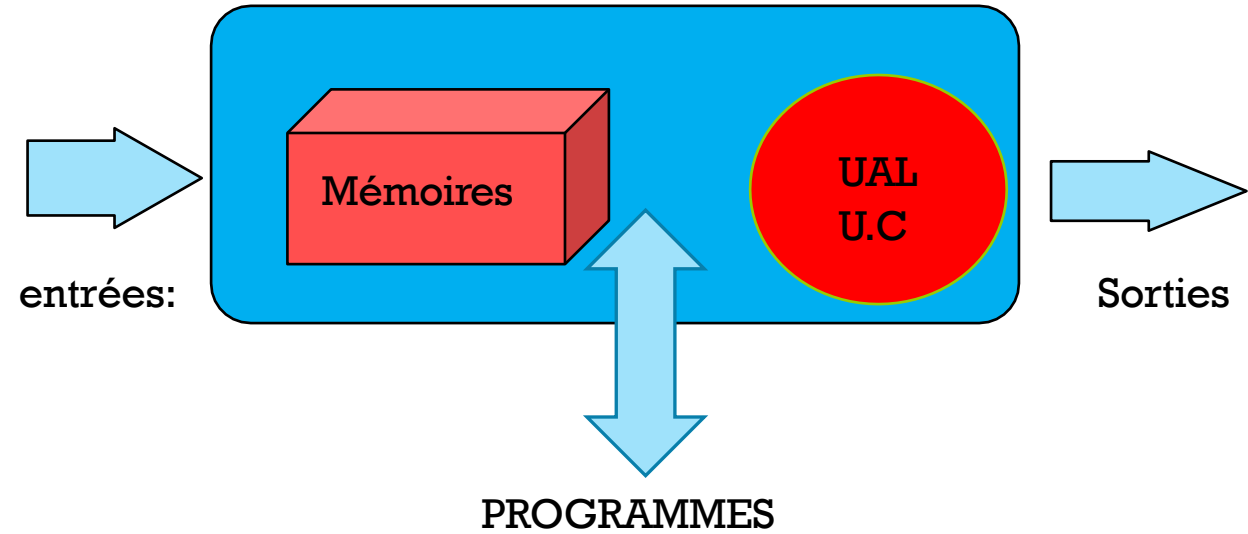
- Un est composé de deux parties :
- Le Hardware (la quincaillerie) :
  - tout ce qui concerne les circuits électriques, électroniques ainsi que le mécanisme.
- Le Software (le programme) :
  - tout ce qui concerne les programmes nécessaires pour le bon démarrage et l'utilisation du micro-ordinateur.

# Hardware

# Entrées/sorties

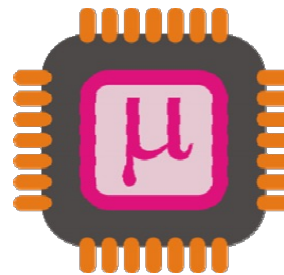


# Unité de traitement



# Unité de traitement

- C'est un organe principal ou le cerveau de l'ordinateur (microprocesseur).
- Il traite les informations introduites dans la mémoire. Il contient:
  - Une unité de commande – U.C
  - Une unité arithmétique et logique – U.A.L



UAL  
U.C

# Unité de commande U.C

- C'est la partie **intelligente** du microprocesseur
- Elle permet de chercher les instructions d'un programme se trouvant dans la mémoire
- De **l'interpréter** pour la suite
- Acheminer les données vers l'UAL pour les traiter

# Unité arithmétique et logique

- Est composée d'un ensemble de circuits (registres mémoires) chargés d'exécuter les opérations arithmétiques/
  - Addition
  - Soustraction
  - Multiplication
  - Division
  - Opérations logiques.



# Mémoire

## Auxiliaires

### Externes



**Flash Disque**

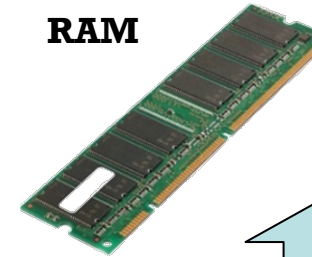
**CD-ROM**

**Carte mémoire**

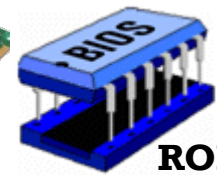
**Disque dure**

## Principales

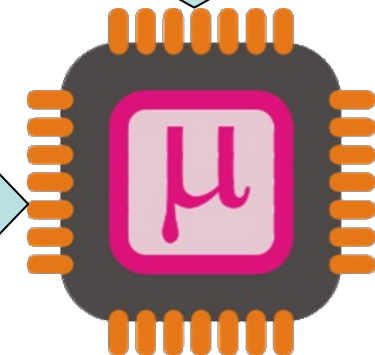
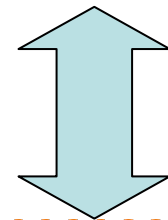
**RAM**



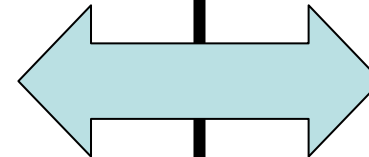
**internes**



**ROM**

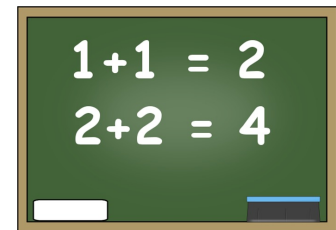
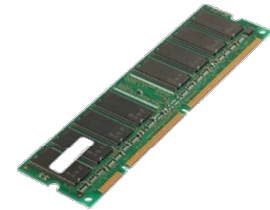


**Unité de traitement**



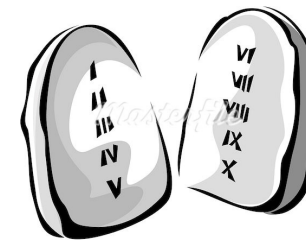
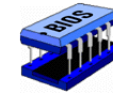
# Mémoire vive RAM

- Random Access Memory (RAM)
- La lecture/écriture de données
- Volatile



# Mémoire morte ROM

- Read only Memory (ROM)
- La lecture seule des données
- Permanente

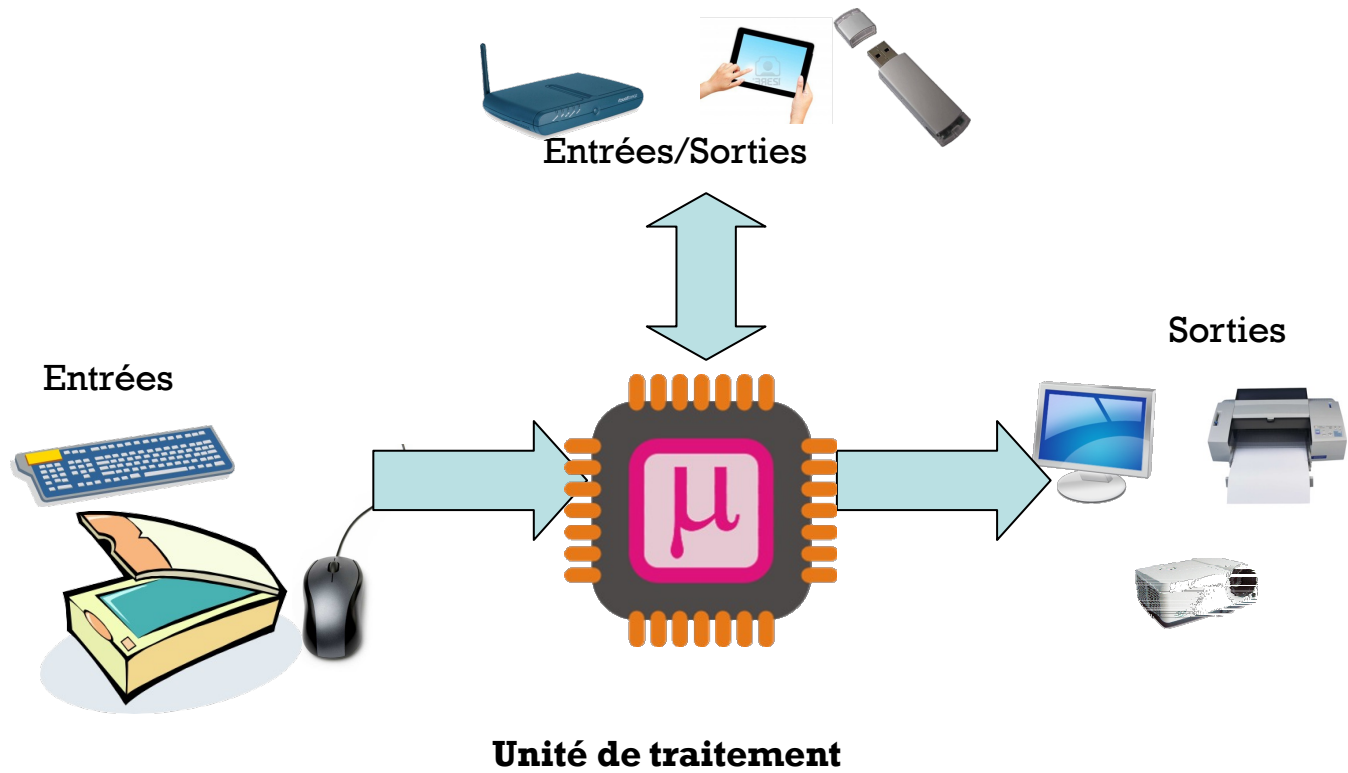


# Mémoires auxiliaires



# Les périphériques

# Entrées/sorties



Entrées



# Sorties

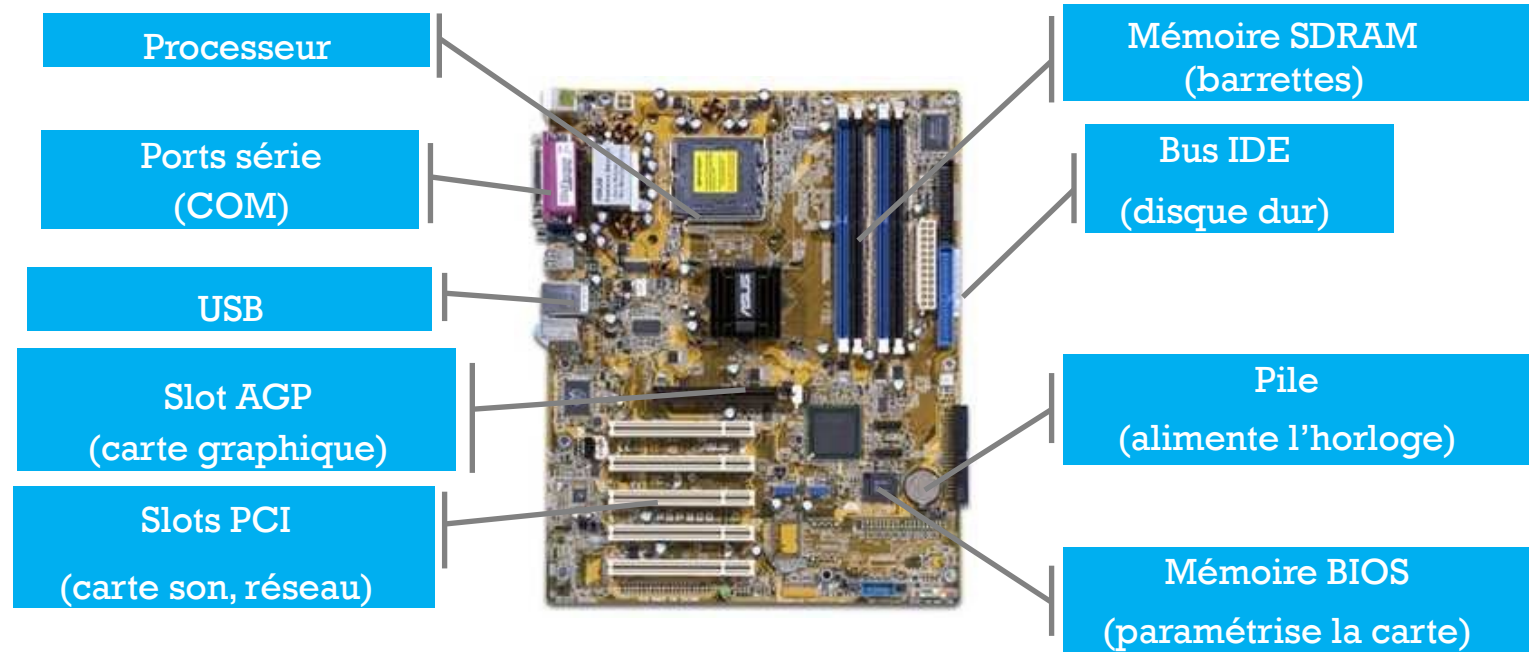


# Entées/Sorties



# La carte mère

# Vue d'ensemble de l'ordinateur Carte mère



# Carte mère

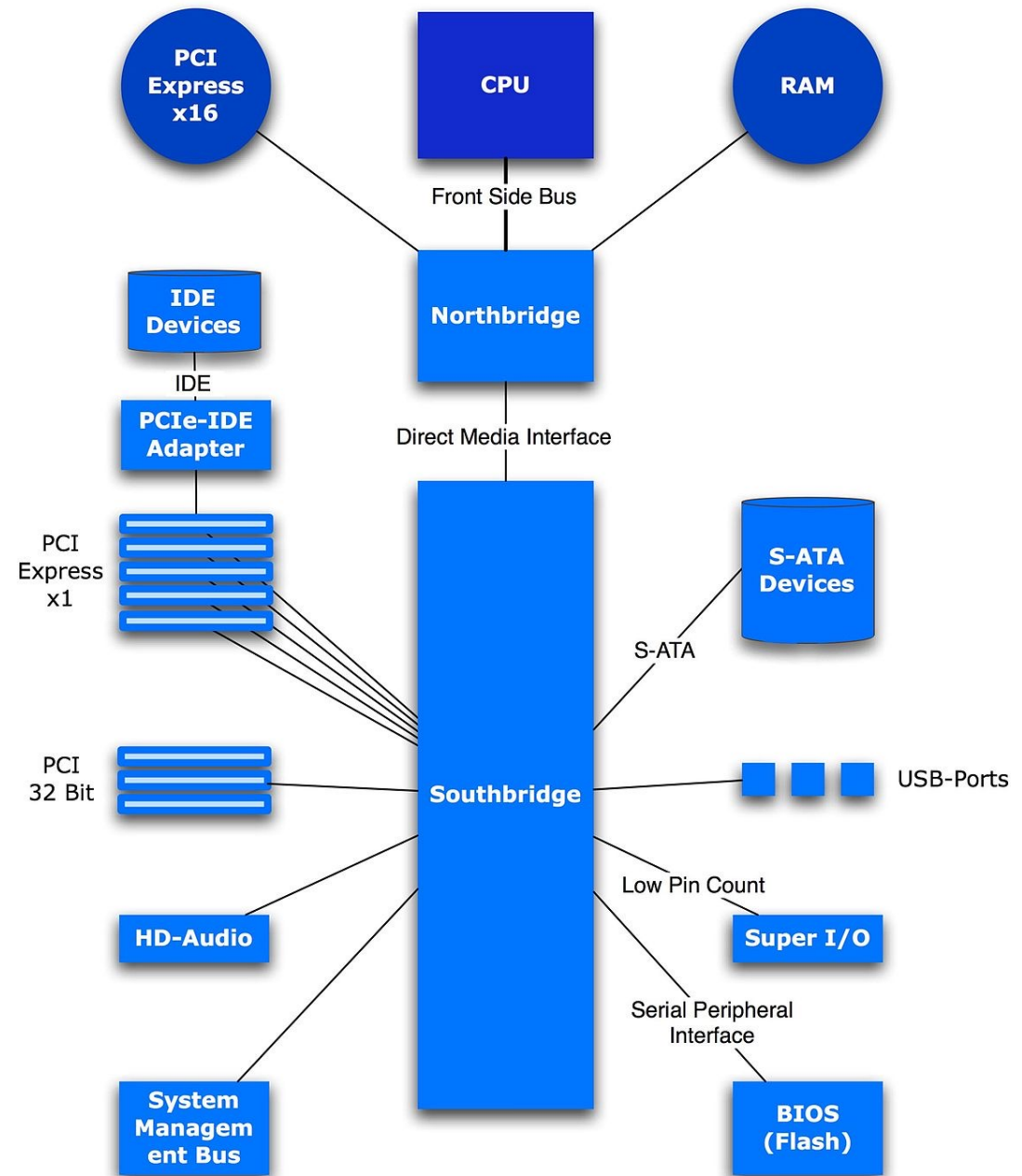
- Le système nerveux du pc.
- C'est sur elle que sont assemblés tous les éléments (ou presque) de l'ordinateur.
- Donnant quelques éléments importants:
  - Le chipset.
  - Le BIOS
  - La pile du CMOS
  - Bus
  - Connecteurs

# Chipset

- Le chipset (aussi appelé jeu de composants)
- est la plateforme centrale de la carte.
- Son rôle est de coordonner les échanges de données entre le processeur, la RAM voir la carte graphique.



# Chipset

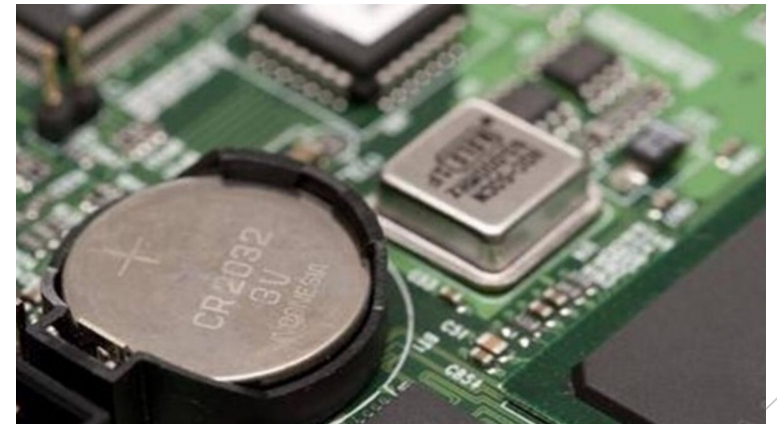


# BIOS

- Le BIOS (Basic Input Output System)
- est présent sur toutes les cartes-mères.
- Il permet au PC de booter (démarrer) et d'initialiser les périphériques
- avant de passer le relais au système d'exploitation (Windows, Linux...).

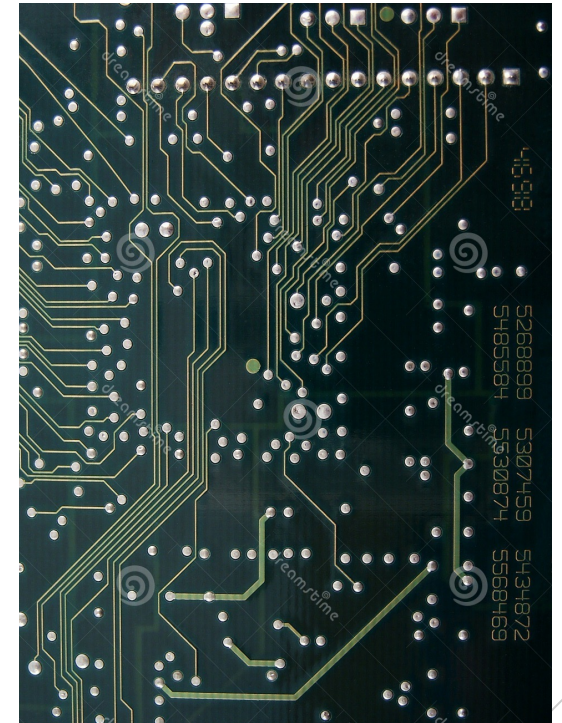
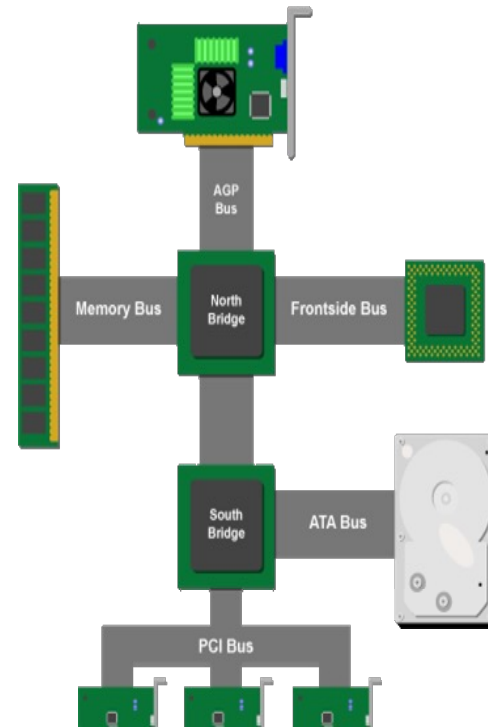
# Pile CMOS

- Complementary Metal Oxide Semiconductor – CMOS
- Lorsqu'on éteint l'ordinateur,
- Il conserve l'heure et tous les paramètres qui lui permettent de démarrer correctement.
- Le CMOS est une mémoire lente mais qui consomme peu d'énergie



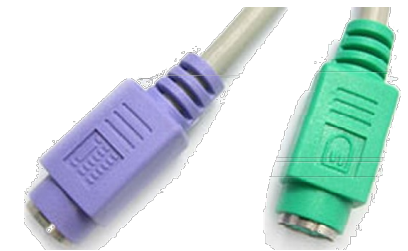
# Les bus

- Un bus est un circuit intégré à la carte-mère qui assure la circulation des données entre les différents éléments du PC (mémoire vive, carte graphique, USB, etc...).



# Les bus

- Le bus système (Front Side Bus) :
  - assure le transport de données entre le processeur et la mémoire vive.
- Le bus série :
  - débouche sur le port servant à brancher une souris ou un modem, de jeux.



# Les bus

- Le bus parallèle :
  - communique avec le port parallèle, qui sert à brancher l'imprimante, le scanner, graveurs externes, etc...
- Le bus USB (Universal Serial Bus) :
  - relié au port USB qui sert à brancher presque tous les périphériques du marché :
    - webcams,
    - modems, imprimantes,
    - scanners, manettes de jeu...



USB



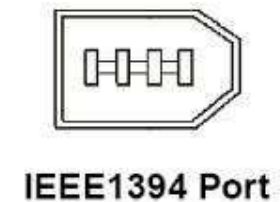
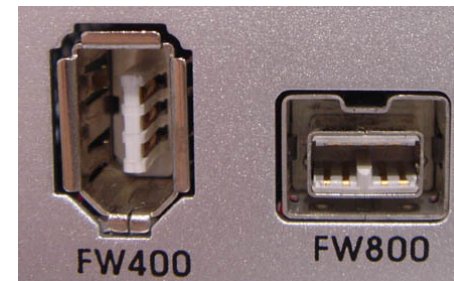
# Les bus

- Le bus P-ATA (Integrates Drive Elect. IDE)
  - Connecter disque dur, lecteur DVD
- Le bus SATA :
  - Serial Advanced Technology Attachment
  - connecter une mémoire de masse à une carte mère.



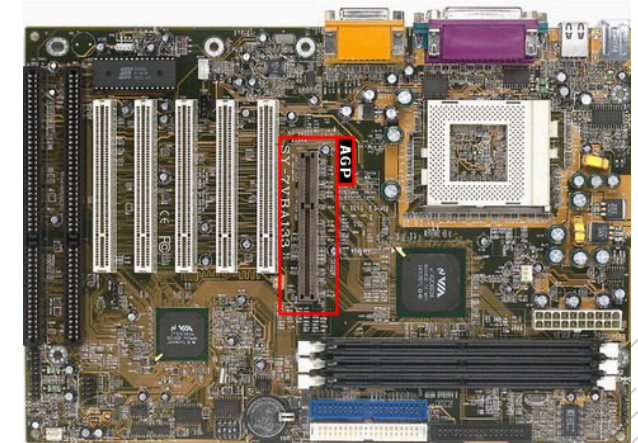
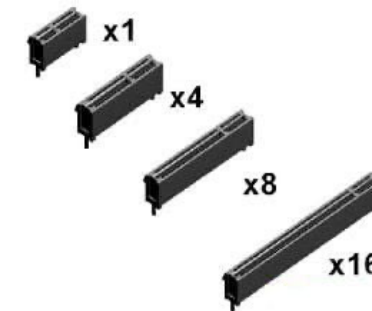
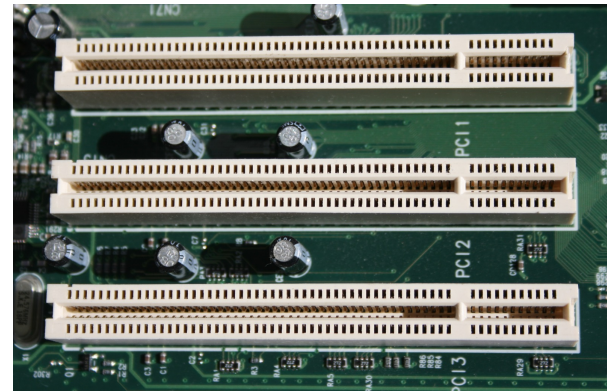
# Les bus

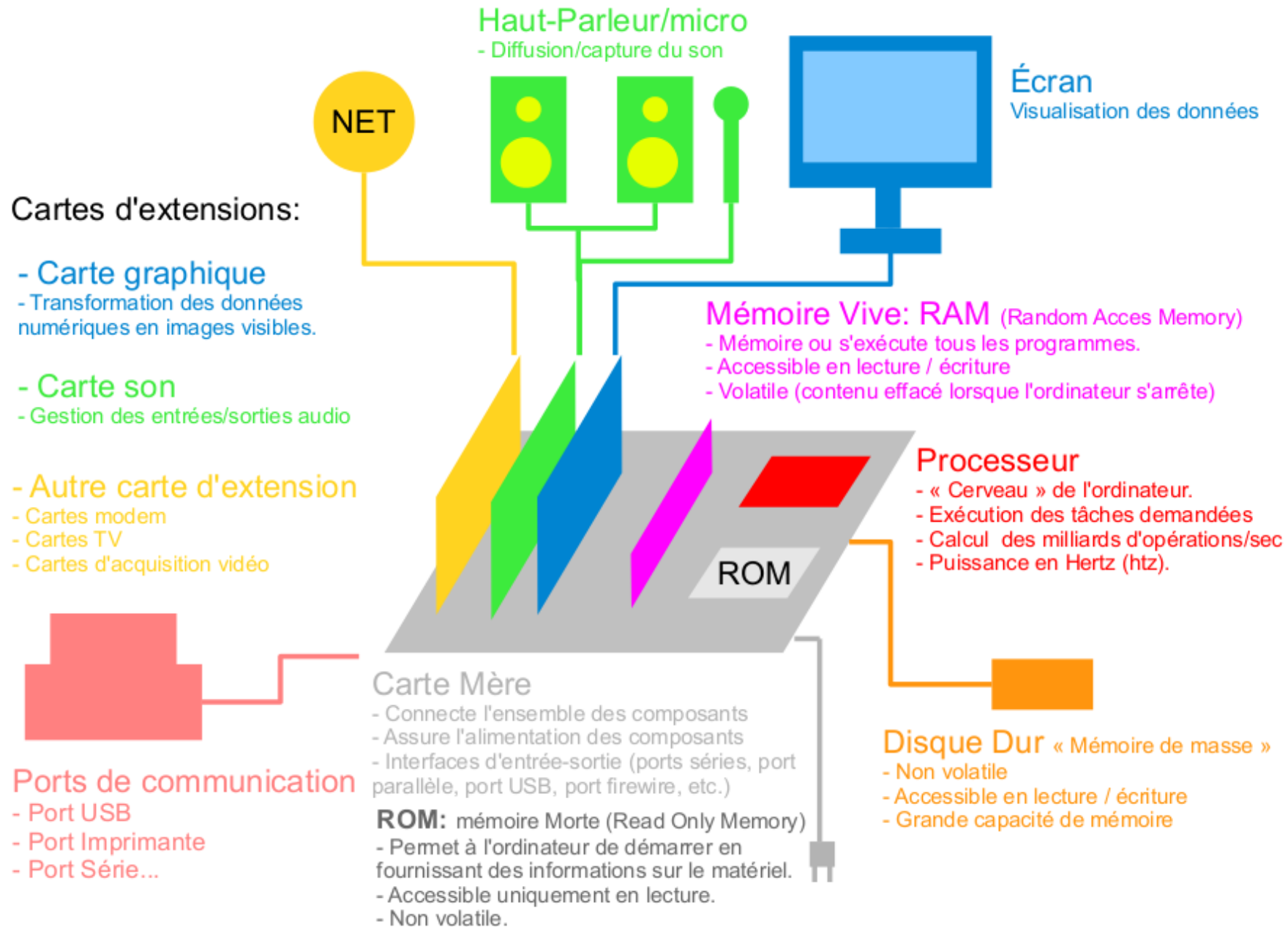
- Le bus FIREWIRE :
  - il permet de brancher 63 périphériques
  - offre des caractéristiques semblables à l'USB, en beaucoup plus performant
  - périphériques qui se branchent sur ce type de port sont rares (et chers).

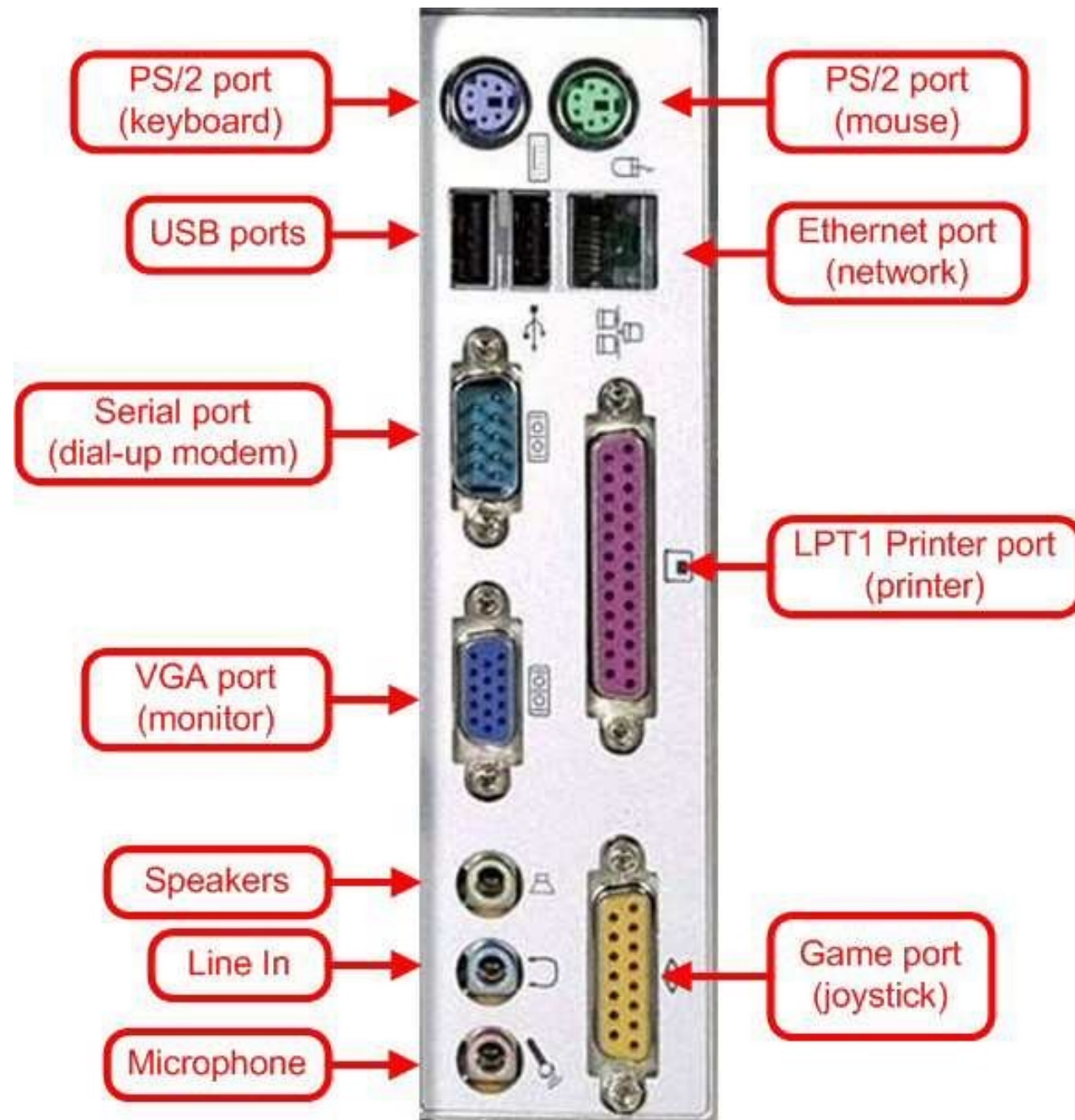


# Les bus

- Le bus PCI (Peripheral Component Interconnect)
- Le bus PCI Express : allant de 250 Mo/s à 4 Go/s via ses nombreuses déclinaisons (1X, 2X, 4X, 8X)
- Le bus AGP (Accelerated Graphic Port) : Il communique avec le port AGP.







# Systeme de codage

# Codage

- Toutes informations manipulées par un ordinateurs
  - (numériques, textuelles, images, sons, vidéos, etc.)
  - est représentée par des séquences de deux chiffres : 0 et 1.
- BIT (BInary digiT)
- Un bit
  - 0 ou 1
  - Représenté par deux états électroniques

# Codage binaire

- **Système décimal**

- Basé sur 10 chiffres (0 – 9)
- $(378)_{10} = 8 \times 10^0 + 7 \times 10^1 + 3 \times 10^2$

- **Système binaire**

- Basé sur deux chiffres 0 ou 1
- $(101101)_2 = 1 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 + 1 \times 2^3 + 0 \times 2^4 + 1 \times 2^5$
- $= 1 + 4 + 8 + 32 = 45$
- $= 1 + 4 + 8 + 32 = 45 = (45)_{10}$

# Autres systèmes de numérotation

- **Octale**

- Basé sur 8 chiffres (0 – 7)

- **Héxadécimal**

- Basé sur 16 chiffres/lettres (0 - 9, A, B, C, D, E, F)

- **Avec**

- $A = (10)_{10} = (1010)_2$
- $B = (11)_{10} = (1011)_2$
- $C = (12)_{10} = (1100)_2$
- $D = (13)_{10} = (1101)_2$
- $E = (14)_{10} = (1110)_2$
- $F = (15)_{10} = (1111)_2$

# Conversion

- Décimale  $\rightarrow$  binaire

- $(378)_{10} = (?)_2$

|       |         |                                    |         |
|-------|---------|------------------------------------|---------|
|       |         | Poids faible<br>(le plus à droite) |         |
| 387:2 | Reste 0 | 189:2                              | Reste 1 |
| 94:2  | Reste 0 | 47:2                               | Reste 1 |
| 23:2  | Reste 1 | 11:2                               | Reste 1 |
| 5:2   | Reste 1 | 2:2                                | Reste 0 |
| 1:2   | Reste 1 |                                    |         |
|       |         | Poids fort<br>(le plus à gauche)   |         |

# Conversion

- Octal → binaire

- On code par groupe de 3 chiffres binaires
- Chaque chiffre octal est remplacé par un chiffre binaire

- $(356)_8 = (11\ 101\ 110)_2$

| Octal | Binaire |
|-------|---------|
| 0     | 000     |
| 1     | 001     |
| 2     | 010     |
| 3     | 011     |
| 4     | 100     |
| 5     | 101     |
| 6     | 110     |
| 7     | 111     |

# Conversion

- Hédadécimal → binaire

- On code par groupe de 4 chiffres binaires
- Chaque chiffre hexa est remplacé par un chiffre binaire

- $(1A5)_{16} = (1\ 1010\ 0101)_2$

| Hexa | Binaire |  | Hexa | Binaire |
|------|---------|--|------|---------|
| 0    | 0000    |  | 8    | 1000    |
| 1    | 0001    |  | 9    | 1001    |
| 2    | 0010    |  | A    | 1010    |
| 3    | 0011    |  | B    | 1011    |
| 4    | 0100    |  | C    | 1100    |
| 5    | 0101    |  | D    | 1101    |
| 6    | 0110    |  | E    | 1110    |
| 7    | 0111    |  | F    | 1111    |

# Conversion

- Binaire → octal
    - On code par groupe de 3 binaires
    - De droite jusqu'à la gauche
- $$(11\ 101\ 110)_2 = (?)_8$$
- $$= (356)_8$$

# Conversion

- Binaire → hexadécimal

- On code par groupe de 4 binaires
- De droite jusqu'à la gauche

$$\begin{aligned}(101\ 1101\ 1110)_2 &= (?)_8 \\ &= (5DE)_8\end{aligned}$$

# Conversion

- **Conversion alphanumérique**

- Chiffres, lettres, signe de ponctuation, les symboles ...
- Sont représentés en utilisant un code normalisé à 8 bits
- On utilise :
  - EBCDIC – Extended Binary Coded Decimal International Code
  - ASCII – American Standard Code Information Interchange

| Lettre/sign/symbole | EBCDIC   | ASCII   |
|---------------------|----------|---------|
| A                   | 11000001 | 1000001 |
| d                   | 10000100 | 1100100 |
| =                   | 01111110 | 0111100 |
| 3                   | 11110011 | 0110011 |

# Conversion

| Nom          | Symbole | Préfixe<br>Système<br>Int. | Préfixe<br>binaire |
|--------------|---------|----------------------------|--------------------|
| Kilo octets  | Ko      | $10^3$                     | $2^{10}$           |
| Méga octets  | Mo      | $10^6$                     | $2^{20}$           |
| Giga octets  | Go      | $10^9$                     | $2^{30}$           |
| Téra octets  | To      | $10^{12}$                  | $2^{40}$           |
| Péta octets  | Po      | $10^{15}$                  | $2^{50}$           |
| Exa octets   | Eo      | $10^{18}$                  | $2^{60}$           |
| Zetta octets | Zo      | $10^{21}$                  | $2^{70}$           |
| Yotta octets | Yo      | $10^{24}$                  | $2^{80}$           |

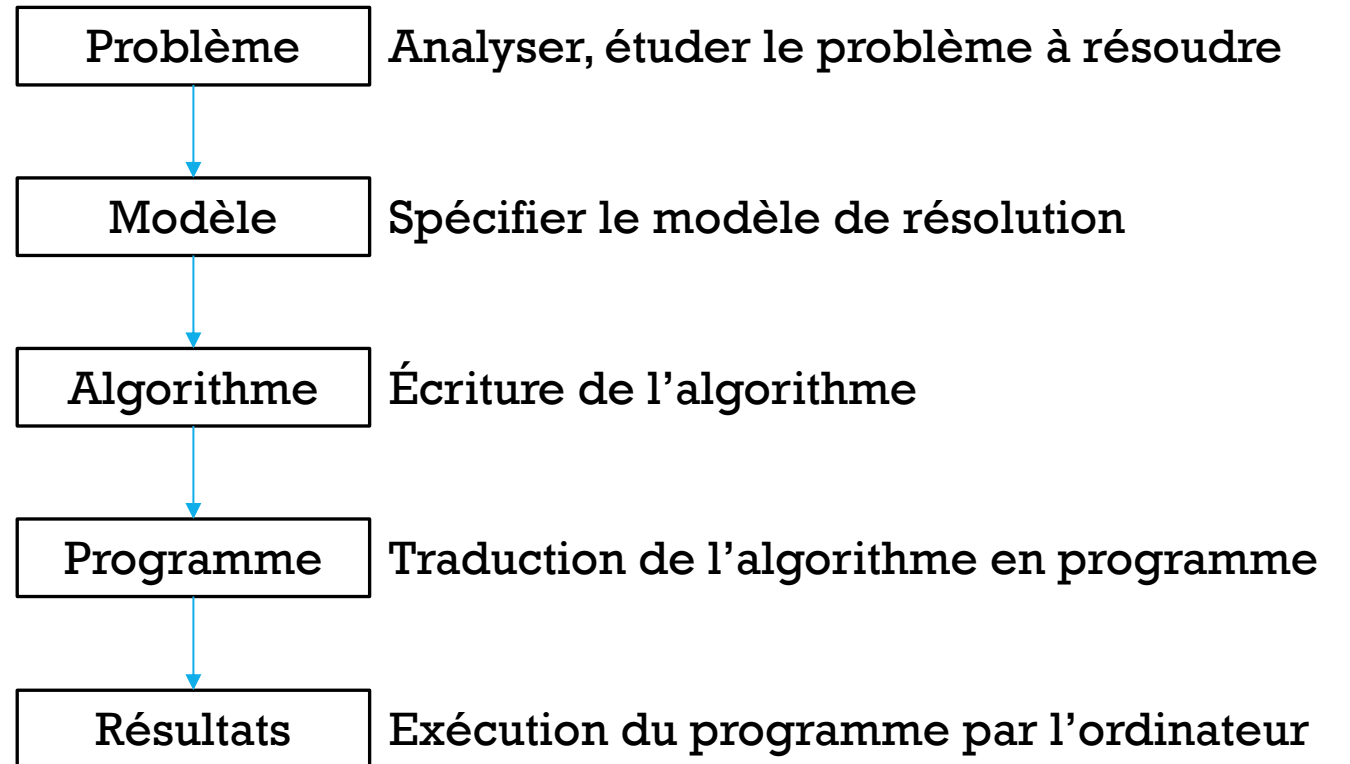
# Chapitre 2

Notions de base de la programmation

# Algorithme

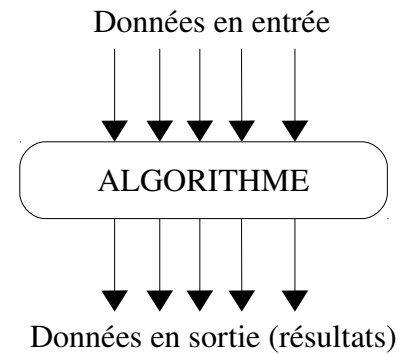
- Algorithme est inventé par Al Khawarizmi
- Est l'énoncé d'une séquence d'actions primitives réalisant un traitement
- Est un ensemble d'actions séquentielles et logiquement ordonnées
- Transformant une (des) entrée(s) en données de sortie(s) pour résoudre un problème.
- Un algorithme représente une solution à un problème donné
  - Spécifiée à travers un ensemble d'instructions qui manipulent des données
  - Écrites en n'importe quelle langue, puis transformé en code source

# Algorithme



# Structure de données

- Un algorithme permet de réaliser un traitement sur un ensemble de données en entrées pour produire des données en sorties



# Structure de données

- Identificateur
- Chaîne de caractère qui contient:
  - Caractères alphanumériques ([a-z][A-Z][0-9])
  - Le tiret 8 ( \_ )
  - Doit commencer par un caractère alphabétique ou \_.
  - Permet d'identifier de manière unique un algorithme, une variable, une constante, une procédure ou une fonction.
- Dans un langage, on doit utiliser un ensemble de mots réservés
- Exemple (Python):
  - and, if, else, for, or, print, return, while, input ...

# Structure de données

- **Constantes et variables**
- **Les données manipulées sont soit des constantes ou des variables**
  - **Constante** : contenant une valeur qui ne change jamais
  - **Variable** : contient une valeur qui peut être changée

# Structure de données

- Types de données
- Cinq type de base:
  - Entiers : l'ensemble {... -4 -3 -2 -1 0 1 2 3 4 5 6 ...}
  - Réels : représente les valeurs fractionnels (avec virgules)
  - Caractères : tous les caractères imprimables
  - Chaînes de caractères : séquence de plusieurs caractères
  - Booléens : vrai ou faux

| Algorithme          | Python |
|---------------------|--------|
| Entier              | int    |
| Réels               | float  |
| Caractère           | char   |
| Chaîne de caractère | str    |
| Booléen             | bool   |

# Structure d'un programme

- Déclaration en algorithme
  - Identifier les données et les déclarer avant de les utiliser
  - En utilisant un identificateur
- Est constitué de trois parties :

Algorithme <identificateur de l'algo>

<Déclarations>

Début

<Corps>

Fin

# Structure d'un programme

- Déclarations
- On déclare toutes les données en entrées et en sorties sous forme de variables et de constantes
  - Constantes : des objets non modifiables
    - Déclarées : <identificateur> = <valeur>;
    - Exp.    PI = 3.14;  
          MAX = 10;  
          cc = 'c';
  - Variables : des objets modifiables
    - Déclarées : <identificateur> = <type>
    - Exp.    x : réel;  
          a : entier;  
          s : caractère

# Structure d'un programme

- Corps
- Constitué d'un ensemble d'actions et instructions séquentiellement ordonnées
- Quatre types d'instructions
  - Lecture : entrer des données à l'algorithme
  - Écriture : affichage des données
  - Affectation : modifier les valeurs des variables
  - Structure de contrôle :
    - Structure de test
    - Structure répétitive

# Types d'instruction

- Toutes les instructions sont écrites dans le corps
- Les regrouper en trois groupes :
  - Les entrées/sorties (saisi de valeurs et affichage du résultats)
  - Les affectations
  - Les structures de contrôles (tests et boucles)

# Types d'instruction

## Les entrées / sorties

### ■ Les entrées

- Permet de donner une valeur quelconque à une variable
- L'instruction concerne uniquement les variables (pas de valeurs ni de constantes)
- Lors de la lecture dans Python, le programme se bloque en attendant la saisie d'une valeur
- Une fois la valeur saisie, on valide par la touche **entrée**

| Algorithme     | Python                  | Signification                                                           |
|----------------|-------------------------|-------------------------------------------------------------------------|
| Lire(<id_var>) | <id_var> = Input('...') | Donner une valeur quelconque à la variable dont l'identifiant <id_var>. |

### ■ Exemples:

- Lire(a)

a = input('Saisir une valeur:')

# Types d'instruction

## Les entrées / sorties

### ■ Les sorties

- Permet de faire afficher un résultat ou un message

| Algorithme                                                    | Python                                                       | Signification                                                                                                           |
|---------------------------------------------------------------|--------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|
| Écrire(<id_var>  <br><const>  <br><valeur>  <br><expression>) | Print(<id_var>  <br><const>  <br><valeur>  <br><expression>) | Afficher une valeur d'une<br>variable, d'une constante,<br>valeur immédiate ou<br>calculée à travers une<br>expression. |

### ■ Exemple

- |                               |                             |
|-------------------------------|-----------------------------|
| ■ écrire('Bonjour');          | print("Bonjour");           |
| ■ écrire(a,b,c);              | print(a,b,c);               |
| ■ écrire(5+2);                | print(5+2);                 |
| ■ écrire('la valeur de x ',x) | print("la valeur de x ",x); |

# Types d'instruction

- Consiste à donner une valeur à une variable

- Possède deux parties :
  - Gauche: représente la variable
  - Droite: une valeur, une variable ou une expression

- Condition de fonctionnement
  - Les deux parties du même type

- $a \leftarrow 5;$                        $a = 5;$
- $b \leftarrow a+5;$                  $b = a+5;$
- $c \leftarrow b;$                      $c = b;$

# Types d'instruction

## Les structures de contrôles

- Structures de contrôles conditionnelles
  - Utilisées pour décider de l'exécution d'un bloc d'instruction
- Deux types d'instructions conditionnelles :
  - Test alternatif simple
  - Test alternatif double

# Types d'instruction

## Test alternatif simple

- Contient un seul bloc d'instructions
- Selon une condition (expression logique)
  - On décide si le bloc d'instructions est exécuté ou non.
  - Si la condition est vraie, on exécute le bloc, sinon on l'exécute pas.

```
si <conditions> alors  
    <instruction(s)>
```

```
finsi
```

### ■ Exemple

```
lire(x)
```

```
si x > 2 alors
```

```
    x ← x + 3;
```

```
finsi
```

```
écrire(x)
```

```
if <condition> :  
    <instruction(s)>
```

```
x = input("Saisir une valeur:");
```

```
if x>2 :
```

```
    x = x + 3;
```

```
print(x);
```

# Types d'instruction

- Test alternatif double
- Contient deux blocs d'instructions
  - On est amené à décider entre le premier et le second bloc
  - Si la condition est vraie, on exécute le premier bloc
  - Sinon, on exécute le second

```
si <conditions> alors  
    <instruction(s) 1>  
sinon  
    <instruction(s) 2>  
finsi
```

## ■ Exemple

```
lire(x)  
si x > 2 alors  
    x ← x + 3;  
Sinon  
    x ← x - 3;  
finsi  
écrire(x)
```

```
if condition :  
    <instruction(s)>  
else :  
    <instruction(s) 2>  
  
x = input("Saisir une valeur:");  
if x>2 :  
    x = x + 3;  
else:  
    x = x-3;  
  
print(x);
```

# Types d'instruction

- Test alternatif multiple
- Contient plusieurs bloc conditionnels

si <conditions> alors

<instruction(s) 1>

Sinon si <condition(s) 2>

<instruction(s) 2>

Sinon

<instruction(s) 3>

Finsi

if condition :

<instruction(s)>

elif condition :

<instruction(s) 2>

else

<instruction(s) 3>

# Types d'instruction

## Structures de contrôle répétitives

- Permet de répéter un traitement un nombre fini de fois
- Boucle pour (for)
- Boucle tant que (while)

# Types d'instruction

- La boucle pour
- Utilise un indice entier qui varie (avec une incrémentation)
  - D'une valeur initiale à une valeur finale
  - À la fin de chaque itération, l'indice est incrémenté de 1 (ou selon le pas)

pour <ind> ← <vi> à <vf> faire  
    <instruction(s)>

finpour;

## Exemple

pour i ← 1 à 6 faire  
    écrire(i);

finpour

for <ind> in <syntaxe> :  
    <instruction(s)>

for i in range(6):  
    print(i);

# For en python

```
for i in [0, 1, 2, 3]:  
    print("i a pour valeur", i)
```

```
for i in range(4):  
    print("i a pour valeur", i)
```

```
c = [ "INOOF:", "INstitut", "d'Optométrie",  
      "Ouled", "Fayet"]
```

```
for i in c:  
    print("i vaut", i)
```

# Types d'instruction

- La boucle tant que
- Utilise une expression logique ou booléenne comme condition d'accès à la boucle
  - Si la condition est vérifiée (résultat vrai), on entre à la boucle, sinon on la quitte.

```
tant-que <conditions> faire  
    <instruction(s)>
```

```
finTant-que
```

```
i ← 1;
```

```
tant-que (i < 6) faire
```

```
    écrire(i);
```

```
    i ← i + 1;
```

```
finTant-que
```

```
while <condition> :  
    <instruction(s)>
```

```
i = 1;
```

```
while i<6 :
```

```
    print(i);
```

```
    i=i+1;
```

- Toute boucle pour peut être remplacée par tant-que

# Types d'instruction

- L'instruction continue
- Permet de passer à l'itération suivante

```
i = 0
while i < 6:
    i += 1
    if i == 3:
        continue
    print(i)
```

# Types d'instruction

- L'instruction **break**
- Permet d'arrêter la boucle même si la condition est **vraie**

```
i = 1
while i < 6:
    print(i)
    if i == 3:
        break
    i += 1
```

# Types d'instruction

- L'instruction else
- Exécute un (des) instruction(s) quand la condition est fausse

```
i = 1
while i < 6:
    print(i)
    i += 1
else:
    print("i n'est plus grand que 6")
```

# Les tableaux

- Utilité des tableaux
- Besoin simultanément de 12 valeurs (par exemple, des notes pour calculer une moyenne).
- Besoin de déclarer douze variables, appelées par exemple Notea, Noteb, Notec, etc.
  - On peut opter pour une notation un peu simplifiée, par exemple N1, N2, N3, etc.
  - Mais ça ne change pas notre problème, car arrivé au calcul, et après une succession de douze instructions « Lire » distinctes, cela donnera :

$$\text{Moy} \leftarrow (N1+N2+N3+N4+N5+N6+N7+N8+N9+N10+N11+N12)/12$$

# Les tableaux

- Qu'est-ce qu'un tableau ?
- Un tableau unidimensionnel ou tableau linéaire est une variable indicée permettant de stocker valeurs de même type.
- Le nombre maximal d'éléments, qui est précisé à la déclaration, s'appelle la dimension (ou capacité ou taille) du tableau.
- Le type du tableau est le type de ses éléments.
- La position d'un élément s'appelle indice ou rang de l'élément.

# Les tableaux

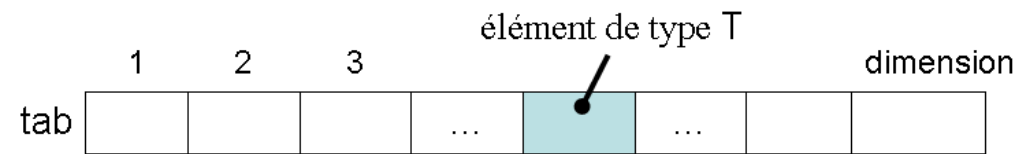
- Remarque
- Tous les éléments d'un tableau portent le même nom.
- Tous les éléments d'un tableau ont le même type ; on parlera d'un tableau d'entiers, d'un tableau de caractères, etc.
- Un tableau possède une taille fixe, connue dès le départ de l'algorithme.
  - Cette taille ne pourra plus changer au cours du déroulement de l'algorithme (pas en python).
- Un tableau peut ne pas être entièrement rempli mais il ne pourra jamais contenir plus d'éléments que le nombre prévu lors de la déclaration.

# Les tableaux

## ■ Déclaration d'un tableau

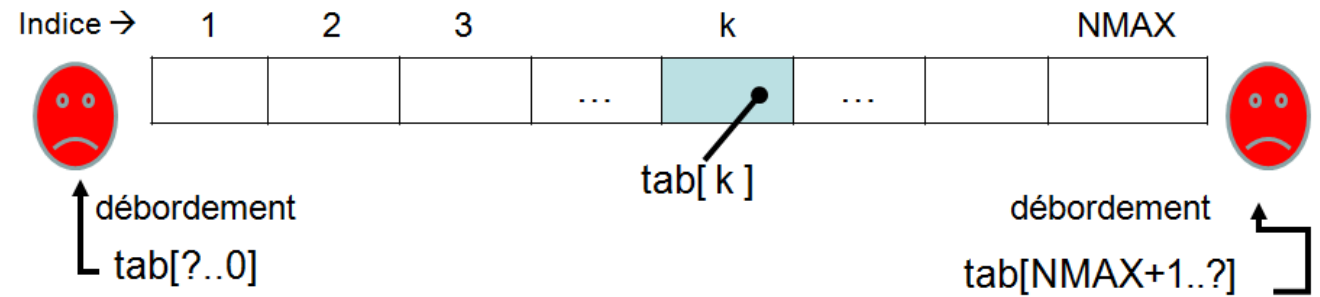
| Algorithme                                                                                                                                                                | Python                                                  |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------|
| <pre>nomVariable[ dimension<br/>] : TypeVariable</pre> <p>Exemple:<br/>Tab[8] : Entier</p> <p>Constante NB_MAX <math>\leftarrow</math> 366</p> <p>Temp[NB_MAX] : Réel</p> | <pre>mon_tab = [5, 8, 6, 9]<br/><br/>mon_tab = []</pre> |

Variable tab : T [ dimension ]



# Les tableaux

- Accès indiciel



- En python, on commence à 0

# Les tableaux

## ■ Exemple Algorithme

Algorithme Tableau

Déclarations

Constante NMAX  $\leftarrow$  24

tab [NMAX], x, k : Entier

Début

écrire(tab[2]);

k  $\leftarrow$  3;

écrire(tab[k]) {L'entier k doit être compris entre 1 et NMAX }

tab[2]  $\leftarrow$  5

x  $\leftarrow$  tab[k]

tab[2]  $\leftarrow$  'a' { erreur, type non compatible. C'est un tableau d'entiers et non un tableau de caractères }

Fin

# Les tableaux

## ■ Exemple Python

```
tab = [10,20,33,-5, 49.5]
```

```
print(tab[0]);
```

```
k = 3;
```

```
print(tab[k]) {L'entier k doit être compris  
entre 0 et NMAX-1 }
```

```
tab[2] = 5
```

```
x = tab[k]
```

```
tab[2] = 'a' { Pas d'erreur}
```

# Les tableaux

- **Fonction de tableau en python**
- **append()**
  - Ajoute un élément à la fin
  - Exemple `tab.append(20)`
- **insert()**
  - Insère un élément à une position souhaitée
  - Exemple: `tab.insert(4, 10)`
- **Extend()**
  - Ajoute plusieurs éléments à une position souhaitée
  - Exemple: `tab.extend(2, 'Bonjour', 'INOOF')`
- **del**
  - Supprime un élément du tableau
  - Exemple: `del tab[4]`
- **len()**
  - Donne la longueur du tableau
  - Exemple: `print(len(tab))`

# Représentation en logigramme

- Un logigramme est une représentation graphique de la résolution d'un problème
- Chaque type d'action dans l'algorithme possède une représentation dans l'organigramme.
- Il est préférable d'utiliser la représentation algorithmique que la représentation par logigramme lorsque **le problème est complexe**.

# Représentation en logigramme

## ■ Les symboles d'organigramme

|                                                                                      |                                                                    |
|--------------------------------------------------------------------------------------|--------------------------------------------------------------------|
|   | Représente le début et la Fin de l'organigramme                    |
|   | Entrées / Sorties : Lecture des données et écriture des résultats. |
|   | Calculs, Traitements                                               |
|   | Tests et décision : on écrit le test à l'intérieur du losange      |
|  | Ordre d'exécution des opérations<br>(Enchaînement)                 |