جامعة مولاي إسماعيل
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵎⵓⵍⴰⵢ ⵉⵙⵎⴰⵄⵉⵍ
UNIVERSITÉ MOULAY ISMAÏL

كلية العلوم
ⵜⴰⵖⵡⵍⵍⵉⵜ ⵏ ⵜⵣⵡⵔⵉ
FACULTÉ DES SCIENCES

Filière M.I.P – S2 : Tronc Commun
Année Universitaire : 2023 – 2024

# Informatique 2 : Algorithmique 2 / Python

**Pr. Badraddine AGHOUTANE**
b.aghoutane@umi.ac.ma

# Plan

- Rappel
- Les fonctions et les procédures en python.
- La récursivité et son application dans des algorithmes.
- Les enregistrements et les fichiers en Python.
- La complexité des algorithmes et ses principaux types.
- Les preuves de correction et de terminaison d'un algorithme.

# Algorithme et programmation

- **Programme**

  **A chaque tâche, que notre PC réalise, correspond un programme.**

**Mais c'est quoi un programme?**

# Algorithme et programmation

- **Programme**

  **Un programme est constitué d'une suite logique d'instructions compréhensible par l'ordinateur.**

  **Une instruction spécifie :**

  - **Les opérations à exécuter**
  - **La façon dont elles s'enchaînent**

# Algorithme et programmation

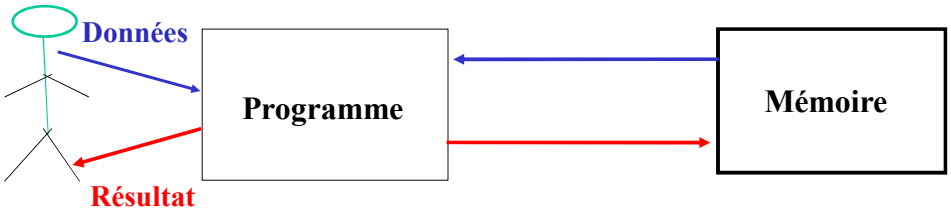
- **Données d'un programme et résultats**

**Un programme peut nécessité des données. Il peut aussi retourner un résultat.**

**Exemple :**

**On dispose d'un programme qui calcule la moyenne des notes :**

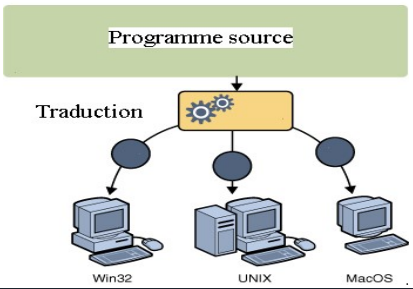
- **Celui-ci a besoin qu'on lui fournisse les notes (données).**
- **Pour qu'il nous retourne la moyenne (résultat).**




# Algorithme et programmation

**Programmation?**

- **L'ordinateur ne comprend que le binaire (0 et 1), est-ce pour autant qu'on doivent écrire des programmes en binaire ?**
- **Il existe des langages de programmation dits « évolués » (proches du langage courant)**
- **Pour chaque langage de programmation, il existe un programme « qui le traduit » en binaire (langage machine)**



# Algorithme et programmation

- **Traduction des programmes**


**Il existe essentiellement deux modes de traduction :**

- **Compilation : la traduction se fait une fois pour toute**
- **Interprétation : a chaque fois qu'on veut exécuter le programme, l'interprète traduit une instruction à la fois. Une fois que celle-ci est exécutée, il passe à l'instruction suivante.**

# Algorithme et programmation

- **Programmation**

  **Programmation = A priori, écriture de programmes dans un langage de programmation (C, Python, C++, Delphi, Java, …)**

  **Or il y a plusieurs langages, est-ce que ça veut dire qu'il existe plusieurs sortes de programmation ?**

  **En fait, la plupart des langages utilisent les mêmes concepts (*variables, structures conditionnelles, structures répétitives, tableaux, fonctions, …*)**

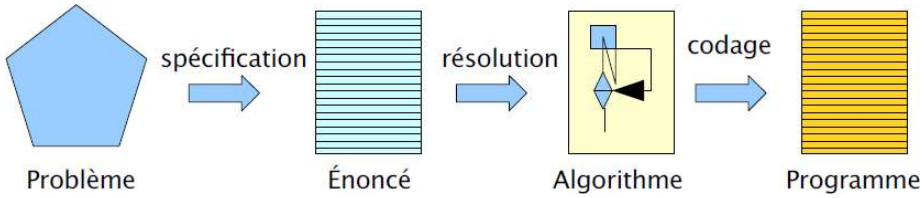
  **➔ dans le cours, on utilisera une notation particulière : notation algorithmique.**

# Bonnes pratiques de la programmation

**La réalisation d'un programme se fait en deux étapes :**

**Etape1 : Analyse du problème et recherche du moyen d'aboutir au résultat à partir des données dont on dispose.**
**➔ écriture d'un algorithme.**

**Etape2 : Traduction de l'algorithme dans un langage de programmation**




# Algorithmique

**Définition d'un algorithme**

**Une description des différentes étapes permettant de résoudre un problème quelconque**

**Exemple :**

**Résolution d'une équation du 2ème degré :**

$$ax^2 + bx + c = 0$$

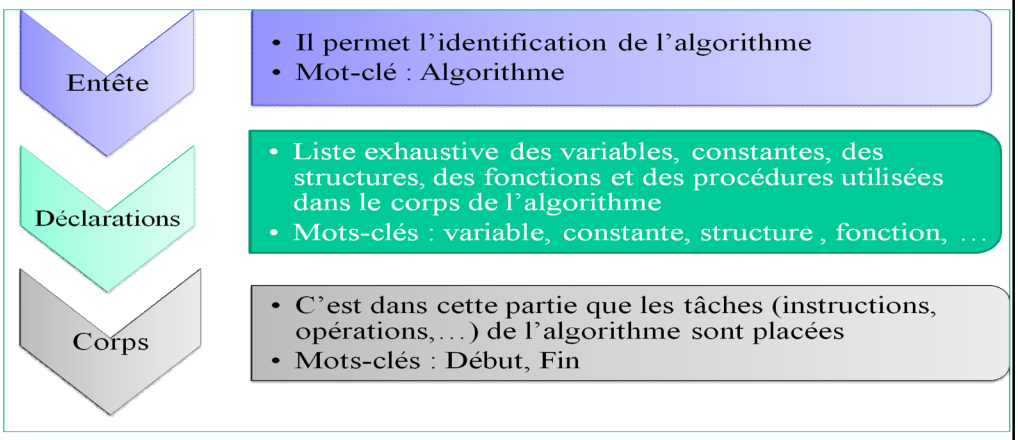
**Etapes de résolution :**

1. **Connaître les valeurs de *a*, *b* et *c***
2. **Calculer le discriminant $\Delta = b^2 - 4ac$**
3. **Si $\Delta < 0$ alors pas de solution**
4. **Si $\Delta = 0$ alors solution double = -b/2a**
5. **Si $\Delta > 0$ alors deux solutions**

# Algorithmique

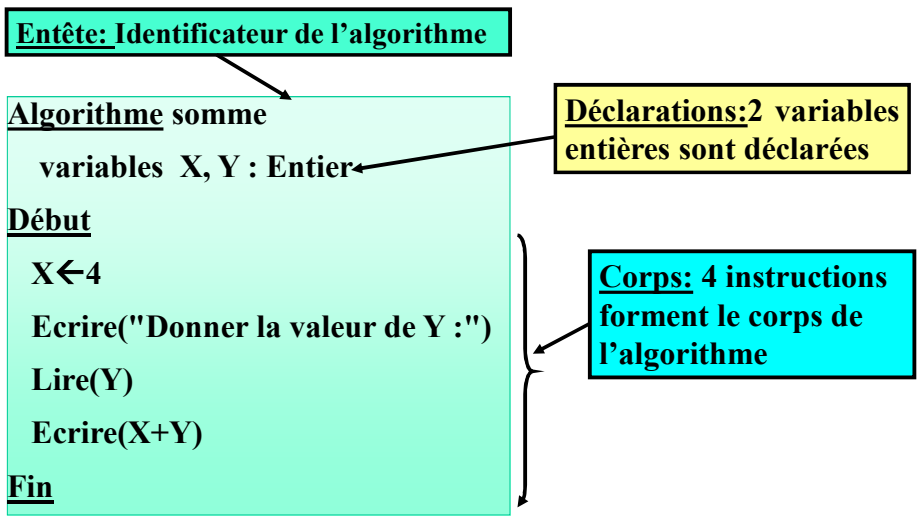
**Définition d'un algorithme**

- **Pour écrire un algorithme, on utilise le langage de description d'algorithme ou le pseudo-code.**
- **Un algorithme est composé de trois parties :**




# Structure d'un algorithme

**Exemple d'un algorithme :**



# Structure d'un algorithme

**Exercice 1 (*Examen de rattrapage*) : Ecrire un algorithme qui permet de lire un entier N et de calculer la valeur de l'expression E, telle que :**

**E = (1+2)*(1+2+3)*(1+2+3+4)*…*(1+2+3+…+(N-2)+(N-1)+N) et (N≥2)**

**Corrigé :**

**Entête: Identificateur de l'algorithme**

```
Algorithme Exercice1
     Variables   N, S, E, i :  entiers;
Début
    Ecrire ("Entrer un entier N") ;
    Lire (N) ;
    S ← 0 ;
    E← 1 ;
    Pour i ← 1 à i ← N faire
        S ← S + i ;
        E ← E * S ;
    FinPour
    Ecrire ("La valeur de E est", E);
Fin
```

**Déclarations:4 variables entières sont déclarées**

**Corps: instructions formant le corps de l'algorithme**

# Structure d'un algorithme

**Exercice 2 (*Examen de rattrapage*) : Ecrire un algorithme qui calcule et affiche le n$^{ième}$ terme (*n est donné par l'utilisateur*) de la suite définie par :**

$$U_n = \sqrt{1+\sqrt{2+\sqrt{3+\sqrt{\ldots+\sqrt{n}}}}}$$

**Rappel :** ***X est exprimée en algorithmique par sqrt(X).***

**Corrigé :**

```
Algorithme Exercice2
    Variables   n,i :  entiers;
                U :   réel;
    Début
        Ecrire ("Entrer un entier") ;
        Lire (n) ;
        U ← 0 ;
        Pour  i ← n  à  i ← 1 (pas de -1) faire
            U ← sqrt(U+i) ;
            Ecrire ("Un= ", U) ;
        finPour
    Fin
```

# Algorithme et programmation

**La programmaion : Ensemble d'activités qui permettent de produire un programme en informatique.**

**Programme : séquence d'instrucions qui spécifie, étape par étape, les opérations à effectuer pour obtenir un résultat.**

- **Un programme est écrit dans un langage de programmaion et exprimé sous une forme interprétable par une machine.**
- **Un programme représente une suite finie et non ambigüe d'instrucions pour résoudre un problème**
- **Une instrucion permet de lire, manipuler, afficher ou sauvegarder des données.**
- **Les instrucions sont traduites en langage machine**

# Présentation de Python

**Python** est un langage portable, extensible, gratuit, qui permet une approche modulaire et orientée objet (sans l'imposer).

**Python** est développé depuis 1989 par **Guido van Rossum** et de nombreux contributeurs bénévoles**.**

- Python un langage interprété ;
- Très largement utilisé dans de nombreux domaines ;
- Syntaxe claire et très simple ;
- Les types python sont nombreux et puissants.
- S'interface facilement avec un très grand nombre de langages ;
- Permet d'écrire des programmes concis avec un haut niveau d'abstraction ;
- Disponible sur Unix, Windows, Mac OS ;
- Possède peu de limites et un champ d'applications très vaste ;

# Bonnes pratiques de la programmation

**Exercice1 :**

**E = (1+2)*(1+2+3)*(1+2+3+4)*…*(1+2+3+…+(N-2)+(N-1)+N) et (N≥2)**

```
Algorithme Exercice1
    Variables  N, S, E, i :  entiers;
Début
    Ecrire ("Entrer un entier N") ;
    Lire (N) ;
    S ← 0 ;
    E← 1 ;
    Pour i ← 1 à i ← N faire
        S ← S + i ;
        E ← E * S ;
    FinPour
    Ecrire ("La valeur de E est", E);
Fin
```

⇒ Code python

```
1  # Write Python 3 code in this editor and run it.
2  print("Entrer un entier N")
3  N=int(input())
4  S=0
5  E=1
6  for i in range(1, N+1) :
7      S=S+i
8      E=E*S
9  print("La valeur de E est: ", E)
```

⇒ Résultat d'exécution (écran)

```
Entrer un entier N
3
La valeur de E est:  18
```

```
Entrer un entier N
5
La valeur de E est:  2700
```

# Bonnes pratiques de la programmation

**Exercice 2 (*Examen de rattrapage*) :** **Ecrire un algorithme qui calcule et affiche le n$^{ième}$ terme (*n est donné par l'utilisateur*) de la suite définie par :**

$$U_n = \sqrt{1+\sqrt{2+\sqrt{3+\sqrt{\ldots+\sqrt{n}}}}}$$

**Rappel :** ***X est exprimée en algorithmique par sqrt(X).***

```
Algorithme Exercice2
    Variables  n,i :  entiers;
               U :  réel;
    Début
        Ecrire ("Entrer un entier") ;
        Lire (n) ;
        U ← 0 ;
        Pour  i ← n à i ← 1 (pas de -1) faire
            U ← sqrt(U+i) ;
            Ecrire ("Un= ", U) ;
        finPour
    Fin
```

Fonction prédéfinie

⇒Code python

```
from math import sqrt
print("Entrer un entier")
N=int(input())
U=0
for i in range(N, 0, -1) :
    U=sqrt(U+i)
print("Un=", U)
```

⇒Résultat d'exécution (écran)

```
Entrer un entier
4
Un= 1.7487627132551438
```

```
Entrer un entier
20
```

# Les fonctions prédéfinies

Certains **traitements sont complexes** à effectuer par un algorithme. Par exemple, le calcul ***de la racine carrée d'un nombre*** nécessite une formule complexe.

- **En algorithmique,** il existe des **fonctions prédéfinies comme :** *Sqrt, Cos, Len, Find, Chr, Asc, Mod, …etc.*

Egalement, tous les **langages de programmation** ont un certain nombre de **fonctions prédéfinies** qui permettent de connaître ce genre de résultat.

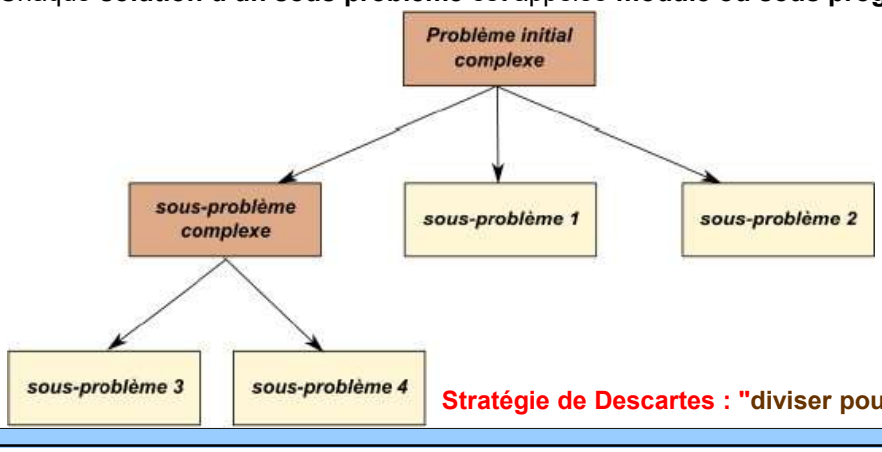
- **En python,** il existe déjà des **fonctions prédéfinies comme :** *len, sqrt, pow, abs, randint, etc.*

➔ **Le concept de la programmation modulaire**

# Le concept de la programmation modulaire

- La **programmation modulaire** consiste à **décomposer un problème** donné en un ensemble fini des **sous-problèmes.**
- Ensuite, pour chaque sous-problème *i* , on cherche de trouver la solution $s_i$ .
- Enfin on regroupe l'ensemble des sous solutions pour aboutir à la résolution du problème de départ.

Chaque **solution d'un sous problème** est appelée **module ou sous programme**.




**Stratégie de Descartes : "diviser pour régner"**

# Le concept de la programmation modulaire

## Qu'est-ce qu'un sous programme ?

Un **sous programme** est une entité qui contient des **données et des instructions**, qui porte un nom donné et qui peut être appelé selon les besoins pour exécuter une tâche donnée.

En **programmation** et en **algorithmique**, on distingue deux types de **sous programme :**

- **Les fonctions :** sous-programme qui retourne un résultat.
  *Exemple : abs(), sqrt()*
- **Les procedures :** sous-programmes qui n'ont pas de résultats.
  *Exemple* : *print(),*

**⇒ Nos objectifs consiste à :**

**1.Utiliser les fonctions prédéfinies :** Ne pas réinventer la roue

**2.Développer nos propres fonctions**

En Python **tout est fonction** : Une procédure est une fonction qui renvoie **None**.

# Algorithmique: Fonctions Prédéfinies

**Textes ou chaîne de caractères :**

- **Len(chaine) :** renvoie le nombre de caractères d'une chaine
- **Mid(chaine,pos,lg) :** renvoie un extrait de la chaine, commençant au caractère pos et faisant lg caractères de long. (commence à 0)
- **Left(chaine,n) :** renvoie les n caractères les plus à gauche dans chaine.
- **Right(chaine,n) :** renvoie les n caractères les plus à droite dans chaine
- **Find(chaine1,chaine2) :**
  - Renvoie la position de la **chaine2** dans **chaine1**,
  - Commence par 0,
  - Si **chaine2** n'est pas comprise dans **chaine1**, la fonction renvoie **-1**

# Algorithmique: Fonctions Prédéfinies

**Textes ou chaîne de caractères :**

- **Exemples :**

| | | |
|---|---|---|
| **Len("Bonjour")** | vaut | **7** |
| **Len("")** | vaut | **0** |
| **Mid("Ahmed is back", 3, 7)** | vaut | **"ed is b"** |
| **Mid("Ahmed is back", 11, 1)** | vaut | **"c"** |
| **Left("Et pourtant…", 8)** | vaut | **"Et pourt"** |
| **Right("Et pourtant…", 4)** | vaut | **"t…"** |
| **Find(" Dans la vie ", "bonheur ")** | vaut | **-1** |
| **Find("Un pur bonheur", "pur")** | vaut | **3** |

# Algorithmique: Fonctions Prédéfinies

**Textes ou chaîne de caractères :**

**Fonction Chr :** fonction qui renvoie le caractère correspondant à un code Ascii donné (),

**fonction Asc :** rôle inverse de Asc()

**Exemples**

| | | |
|---|---|---|
| **Chr(64)** | **vaut** | **'@'** |
| **Asc('N')** | **vaut** | **78** |

# Algorithmique: Fonctions Prédéfinies

## Numériques :

**Partie Entière: Ent()**

Fonction qui permet de récupérer la partie entière d'un nombre :

**A ← Ent(5,28)** **A vaut 5**

**Modulo: Mod()**

Cette fonction permet de récupérer le reste de la division d'un nombre par un deuxième nombre. Par exemple :

**A ← Mod(17,3)** **A vaut 2 car 17 = 5*3 + 2**
**B ← Mod(12,2)** **B vaut 0 car 12 = 6*2**

# Algorithmique: Fonctions Prédéfinies

## Numériques :

**Racine carée d'un nombre : Sqrt()**

Fonction qui permet de récupérer la racine carée d'un nombre :

**A ← Sqrt(25)** **A vaut 5**

**Valeur absolue : Abs()**

Cette fonction permet de le renvoie de la valeur absolue du nombre. Par exemple :

**A ← Abs(17)** **A vaut 17**
**B ← Abs (-12)** **B vaut 12**