جامعة مولاي إسماعيل
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵎⵓⵍⴰⵢ ⵉⵙⵎⴰⵄⵉⵍ
UNIVERSITÉ MOULAY ISMAÏL

كلية العلوم
ⵜⴰⵖⵣⵓⵜ ⵏ ⵜⵣⵔⴼⵜ
FACULTÉ DES SCIENCES

Filière M.I.P – S2
Année Universitaire : 2024 – 2025

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 ces opérations s'enchaînent (*séquence, structure alternative, répetivies, …)***

Algorithme et programmation

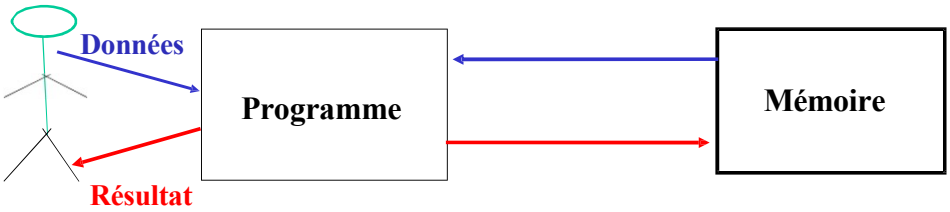
- **Données d'un programme et résultats**

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

Exemple :

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

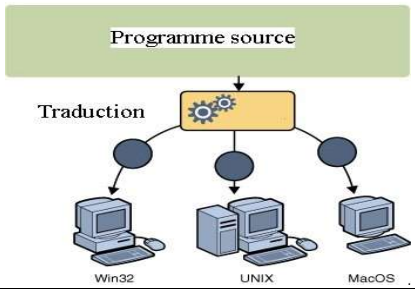
- **Celui-ci a besoin qu'on lui fournisse les notes (données).**
- **Pour qu'il nous retourne la moyenne (résultat).**



Algorithme et programmation

Programmation?

- **L'ordinateur ne comprend que le binaire (0 et 1), est-ce pour autant qu'on doit 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 (*ou 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++, 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, …*)

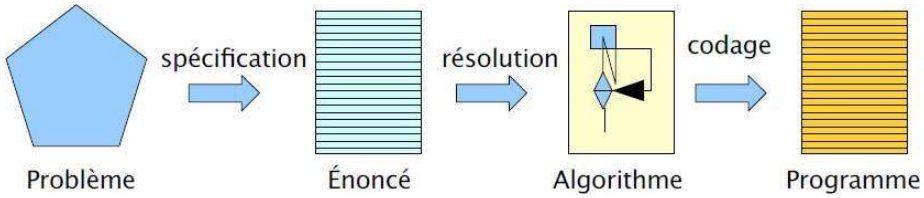
➔ Dans ce cours, on utilisera une notation intermédiaire : notation algorithmique.

Bonnes pratiques de la programmation

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

Etape1 : Analyser le problème et rechercher un 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 étapes conduisant à la résolution d'un problème

Exemple :

Résolution d'une équation du 2ème degré :

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

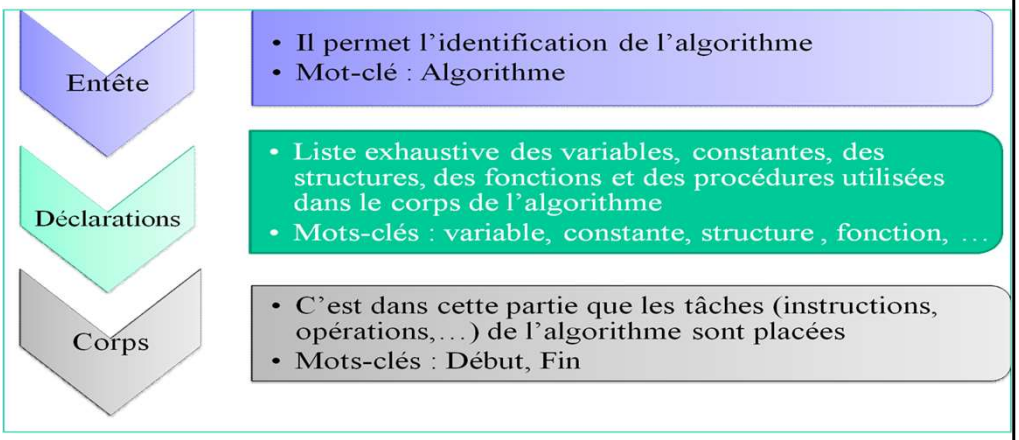
Etapes de résolution :

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 :

Entête: Identificateur de l'algorithme

```
Algorithme Somme
   variables  X, Y : Entier
Début
   X← 4
   Ecrire("Donner la valeur de Y :")
   Lire(Y)
   Ecrire(X+Y)
Fin
```

Déclarations:2 variables entières sont déclarées

Corps: 4 instructions forment le corps de l'algorithme

Structure d'un algorithme

Exercice 1 (*Examen de rattrapage*) :

Parmi tous les entiers supérieurs à 1, seuls quatre peuvent être exprimés comme la somme des cubes de leurs chiffres.

Par exemple : $153 = 1^3 + 5^3 + 3^3$, ce qui en fait un nombre cubique selon cette propriété.

Écrire un algorithme pour trouver les nombres cubiques sachant qu'ils sont dans l'intervalle [150, 410].

Structure d'un algorithme

Corrigé :

Entête: Identificateur de l'algorithme

```
Algorithme nombre_cubiques ;
Var   i, k1, k2, k3: Entier;
Début
      Pour i de 150 à 410  faire
            k1← i Div100;
            k2← (i Mod 100) div10;
            k3← i Mod 10;
           Si (k1*k1*k1 + k2*k2*k2 + k3*k3*k3 = i)
alors
                  Ecrire(i," est un nombre cubique ");
          Finsi
      FinPour
Fin
```

Déclarations: 4 variables entières

Corps: instructions formant le corps de l'algorithme

Algorithme et programmation

La programmaion : Ensemble d'activités (*Spécification, Résolution, Codage*) qui permettent de produire un programme informatique.

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

- **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, traiter, afficher ou sauvegarder des données.**
- **Le programme est traduit en langage machine**

Présentation de Python

Python est un langage portable, 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 ;
- Syntaxe claire et très simple ;
- Les types python sont nombreux et puissants.
- Très largement utilisé dans de nombreux domaines ;
- 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

Exercice 1 (Traduction en langage Python) :

```
Algorithme nombre_cubiques ;
Var   i, k1, k2, k3: Entier;
Début
 Pour i de 150 à 410  faire
            k1← i Div100;
            k2← (i Mod 100) div10;
            k3← i Mod 10;
   Si (k1*k1*k1 + k2*k2*k2 +
k3*k3*k3 = i) alors
    Ecrire(i," est un nombre cubique ");
    Finsi
FinPour
Fin
```

⇒ **Code python**

```
Exercice1_Rat.py
for i in range (150, 411) :
    K1= i//100
    K2=(i%100)//10
    K3=i%10
    if (K1**3 +K2**3+K3**3==i):
        print("Le nombre ", i, "est CUBIQUE")
```

⇒ **Résultat d'exécution (écran)**

```
Le nombre  153 est CUBIQUE
Le nombre  370 est CUBIQUE
Le nombre  371 est CUBIQUE
Le nombre  407 est CUBIQUE
```

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 (avec, U_0=0):**

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

Rappel : ***La racine carrée de 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
```

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 :**

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

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** pour être 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 calculer ce genre de résultat.

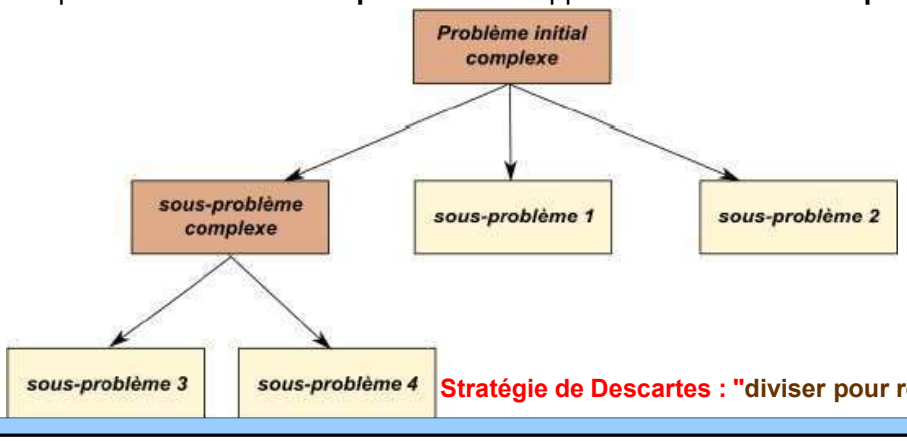
- **En python,** il existe déjà des **fonctions prédéfinies comme :** *len,* ***sqrt****, pow, abs, randint, etc.*

➔ Principe 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.**
- Pour chaque sous-problème *i* , on cherche une solution s_i .
- On regroupe l'ensemble des sous solutions pour aboutir à la résolution du problème initial complexe.

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(),*

Notre objectif 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")
Len("")
Mid("Ahmed is back", 3, 7)
Mid("Ahmed is back", 11, 1)
Left("Et pourtant…", 8)
Right("Et pourtant…", 4)
Find(" Dans la vie ", "bonheur ")
Find("Un pur bonheur", "pur")
```

Algorithmique: Fonctions Prédéfinies

Textes ou chaîne de caractères :

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

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

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 = 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)
B ← Mod(12,2)

A = 2 car 17 = 5*3 + 2
B = 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 = 5**

Valeur absolue : Abs()

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

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

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

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

```python
# Write Python 3 code in this editor and run it.
print("Entrer un entier N")
N=int(input())
S=0
E=1
for i in range(1, N+1) :
    S=S+i
    E=E*S
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
```