Cours et exercices

Version 2.0

4ème * Sciences de l'informatique

ALGORITHMIQUE & PROGRAMMATION

Nom et Prénom :

Classe : 4 ème Sciences de l'Informatique

Lycée OTHMAN CHATTI M'SAKEN

2022 - 2023

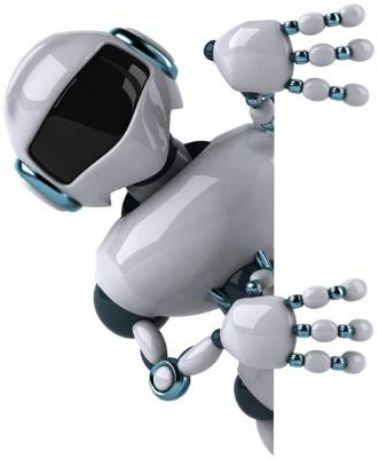
Présentation du manuel :

Ce manuel est destiné aux élèves du niveau 4ème année secondaire de la section "Sciences de l'informatique ", il est reparti sur les 7 parties suivantes :

Partie 1 : Révision générale du programme 3ème Année
Partie 2 : Les fichiers texte et binaires
Partie 3 : Les algorithmes de tri
Partie 4 : La récursivité
Partie 5 : Les algorithmes récurrents
Partie 6 : Les algorithmes arithmétiques
Partie 7 : Les algorithmes d'approximation

Chaque partie présente une explication simple, détaillée et conviviale des notions algorithmiques dont l'élève aura besoin pour résoudre n'importe quel problème algorithmique.
Le manuel contient également une série de 65 exercices pour bien consolider les acquis des élèves durant l'année scolaire.

Pour plus d'information
elbahi.anis@gmail.com



Partie 1 : RAPPEL

Révision Générale :
Programme 3 ème Année
(Algorithmique et Programmation)

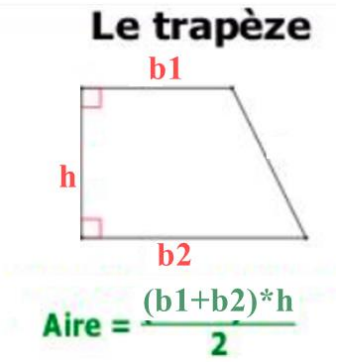
Série d'exercices 1 :

Objectif :

1- Elaboration d'un algorithme et sa traduction en Python
2- Structures de données (simples et complexes)
3- Structures de contrôle (conditionnelles et itératives)
4- Décomposition modulaire d'un programme.
5- Vecteurs et Matrices
6- Algorithmes de tri
7- Vecteurs d'enregistrements

Exercice 1 (Premier programme)

On désire faire le programme nommé « trapèze » qui calcul l'aire d'un trapèze en demandant à l'utilisateur de saisir les valeurs de la hauteur (h) , la base1 (b1) et la base 2 (b2) Sachant que toutes les valeurs saisies doivent être strictement positifs.



Travail demandé :

1- Faire l'algorithme du programme principal
2- Faire les algorithmes des modules envisagés.
3- Donner le code Python correspondant.

Exercice 2 (Fonctions prédéfinies, opérateurs et types de données)

Compléter le tableau suivant par l'instruction algorithmique, en python, la valeur et le type de x:

	Instruction algorithmique	Instruction En python	Valeur de x	Type de x
1	x← (15 + 6 + 1) * 2 + 5.	x = (15 +6 + 1) * 2 + 5.		float
2	x← 1-2+3.			
3	x← long("Python") div 2			
4	x← (4*5*6*7*0*3) + 15			
5	x← non (ord("A") > ord("a"))			
6	x← 21 - 7 * Ent(2.75)			
7	x← Aléa(1,100) > 120			
8	x← (10 ≠ (9+1)) ou (12 > -1)			
9	x← Ent (Abs(- 3.75)) / 2			
10	x← Chr(Ord("a")-32) +Majus("b")			

Exercice 3 (Bac Pratique 2011 - scientifique)

Soit la somme Sn suivante :

Faire un programme Python qui permet de :

1- Saisir un entier n avec **1≤n≤100**
2- Calculer et afficher la somme Sn.

$$S_n = \frac{1}{1^3} + \frac{1}{2^3} + \frac{1}{3^3} + \frac{1}{4^3} + \frac{1}{5^3} + \ldots + \frac{1}{n^3}$$

Exercice 4 (Tournage à la main)

Soit l'algorithme de la fonction "travail" suivant :

0) Fonction Travail (N : entier) : ……..
1) R ← 0
2) Répéter
R ← R + N mod 10
N ← N div 10
Jusqu'à (N=0)
3) Retourner (R)
4) Fin travail

Remarque

On remarque que l'algorithme peut avoir une structure différente à celle que nous utilisons, mais le principe reste le même

Travail à faire :

1- Compléter par le type de résultat retourné par la fonction travail.
2- Compléter le Tableau de Déclaration des Objets Locaux de la fonction Travail.

Objet	Type / Nature
……	……

3- Quelle est la valeur retournée par la fonction Travail pour N = 125
Pour N = 125 le résultat = ……

4- Déduire le rôle de la fonction travail.

Exercice 5 (Manipulation d'un tableau)

On désire faire un programme intitulé « **Armstrong** » qui permet de faire les tâches suivantes :

1- Saisir le nombre N de cases à remplir dans le tableau : $4 \leq N < 15$
2- Remplir un tableau T par N entiers positifs de 3 chiffres chacun.
3- Afficher tous les entiers d'Armstrong qui se trouvent dans le tableau T.

ATTENTION

Un entier est dit d'Armstrong s'il est égal à la somme des cubes de ses trois chiffres

EXEMPLE $153 = 1^3+5^3+3^3$ (153 est un entier d'Armstrong)

153
370
371
407

Travail demandé :

1- Donner l'algorithme du programme principal
2- Donner les algorithmes des modules envisagés
3- Donner le programme Python correspondant

Remarque

En python, pour implémenter un tableau, on peut utiliser la bibliothèque numpy

```
#déclaration d'un tableau
from numpy import *
T=array([int()]*5)
```

```
#déclaration d'un tableau
import numpy as np
T=np.array([int()]*5)
```

Exercice 6 (Vecteur + nombres premiers)

On désire faire un programme qui charge un tableau **T** par **N** entiers strictement positifs (**avec 5 ≤ N ≤ 50**) puis d'afficher tous les entiers premiers qui se trouvent dans le tableau.

REMARQUE *Un entier est dit premier s'il n'a que deux diviseurs distincts 1 et lui-même comme : 2,3,5,7,11,13,...*

Travail à faire :

Faire le programme python permettant de résoudre le problème.

Exercice 7 (Matrice + aléatoire)

On désire faire un programme nommé « **Matrice_1** » permettant de faire les tâches suivantes :

1- Remplir une matrice carrée **M** d'ordre **N** par des entiers de façon aléatoire sachant que **N∈ [2..10]** et **M[i,j] ∈ [1..100]**.
2- Afficher le contenu de la matrice **M**.
3- Calculer et afficher la somme de la matrice **M.**

Travail à faire :

1- Faire l'algorithme du programme principal.
2- Faire les algorithmes des modules envisagés.

Exercice 8 (Tri à bulles d'un vecteur)

On désire faire un programme nommé « **classement** » qui permet de :

- Saisir N représentant le nombre d'élèves d'une classe (**5 < N < 40**)
- Remplir un tableau T par N réels représentant chacun la moyenne générale de chaque élève (T[i] ∈ [0 .. 20]) .
- Trier le contenu du tableau T par ordre croissant en appliquant le tri à bulles.
- Afficher le contenu du tableau avant et après le tri.

Travail demandé :

1- Faire l'algorithme du programme principal
2- Faire les algorithmes des modules envisagés
3- Faire le programme python correspondant

Exercice 9 (Tri par sélection)

1- Donner l'algorithme de la procédure nommée « **Tri_sélection** » permettant de trier d'une manière croissante, un tableau **T** rempli par **N** réels positifs en appliquant le tri par sélection.

2- Soit l'algorithme du programme principal suivant :

```
ALGORITHME classement
BEDUT
   Saisir(N)
   Remplir(T,N)
   Afficher(T,N)
   Tri_sélection(T,N)
   Afficher(T,N)
FIN
```

a) Compléter le TDNT sachant que T est un tableau pouvant contenir 40 réels représentants les moyennes générales des élèves d'une classe.

Tableau de Déclaration de nouveaux types :

Nouveaux Types

b) Compléter le TDOG sachant que les modules utilisés sont tous des procédures et N représente la taille du tableau (le nombre d'élèves de la classe).

Tableau de Déclaration des Objets Globaux :

Objet	Type / Nature

c) Développer les algorithmes des modules :

* **Saisir :** permettant de lire la valeur N (**avec 3≤N≤40**)
* **Remplir :** permettant de remplir un tableau par les moyennes de N élèves.
* **Tri_sélection :** permettant de trier le tableau T par **ordre croissant** en appliquant le tri par sélection.
* **Afficher :** permettant d'afficher le contenu d'un tableau T de N éléments.

3 – Donner le code Python de la procédure **Tri_sélection** développée dans la question précédente.

Exercice 10 (Manipulation d'une matrice)

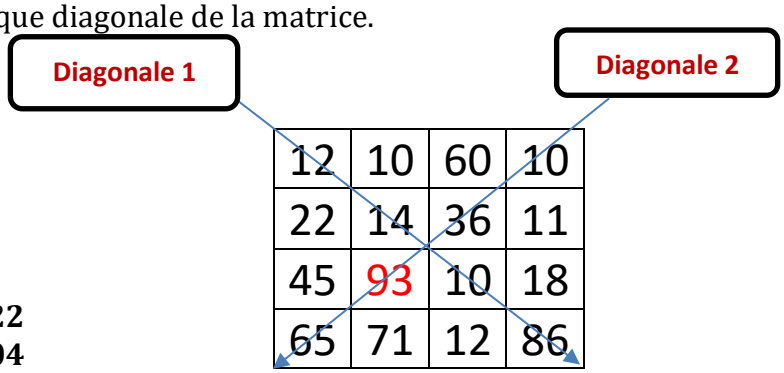
Ecrire un programme qui permet de :

1- Remplir une matrice carrée M d'ordre N par des entiers positifs de deux chiffres chacun sachant que **2<N≤10** .
2- Chercher puis afficher le maximum de la matrice.
3- Calculer puis afficher la somme des éléments de la matrice.
4- Calculer et afficher la somme de chaque diagonale de la matrice.

EXEMPLE

Pour N=4 et la matrice M suivante :

Le programme doit afficher :
Le maximum de la matrice = 93
La somme de la matrice = 575
La somme de la diagonale 1 = 122
La somme de la diagonale 2 = 204



Travail à faire :

1- Donner l'algorithme du programme principal.
2- Faire les algorithmes des modules proposés.
3- Donner le script Python de votre programme

Exercice 11 (Enregistrement)

Partie1 :

Donner deux exemples d'objets du monde réel ainsi que certaines de leurs caractéristiques comme le montre l'exemple suivant :

EXEMPLE

OBJET	CARACTERISTIQUES (PROPRIETES)
VOITURE	Matricule, Puissance, Couleur, marque, ...

Les objets présentés dans le tableau peuvent être présentés en programmation sous forme .....

Partie 2 :

Soit l'enregistrement eleve suivant :

ELEVE

Champ	Signification	Type	Valeur
c	Code	Entier	1234
n	Nom	Chaine	"Elbahi"
p	Prénom	Chaine	"Anis"
mg	Moyenne Générale	Réel	13.25

1- Implémenter en python une variable nommée E de type eleve en remplissant ses champs par les valeurs correspondantes.
2- Afficher tous les champs de l'enregistrement E
3- Ajouter 1.5 à la moyenne générale de l'élève E
4- Afficher la nouvelle valeur du champ mg.

Exercice 12 (Enregistrement ayant un champ de type enregistrement)

Soit l'enregistrement développeur suivant :

Développeur

Champ	Signification	Type
c	Code	Entier
n	Nom	Chaine
p	Prénom	Chaine
prog	Les langages de programmation maitrisés	Tableau de 5 chaines
dt	Date de naissance	Date
adrs	Adresse postale	Chaine

NB : Date est un enregistrement de 3 champs

- J : entier #jour de naissance
- M : entier #mois de naissance
- A : entier #année de naissance

1- Déclarer en algorithmique l'enregistrement Date.
2- Déclarer en algorithmique l'enregistrement développeur.
3- Implémenter votre solution (1) et (2) en python en donnant des valeurs de votre choix aux champs des deux enregistrements

Exercice 13 (Tri d'un vecteur d'enregistrements)

RAPPEL

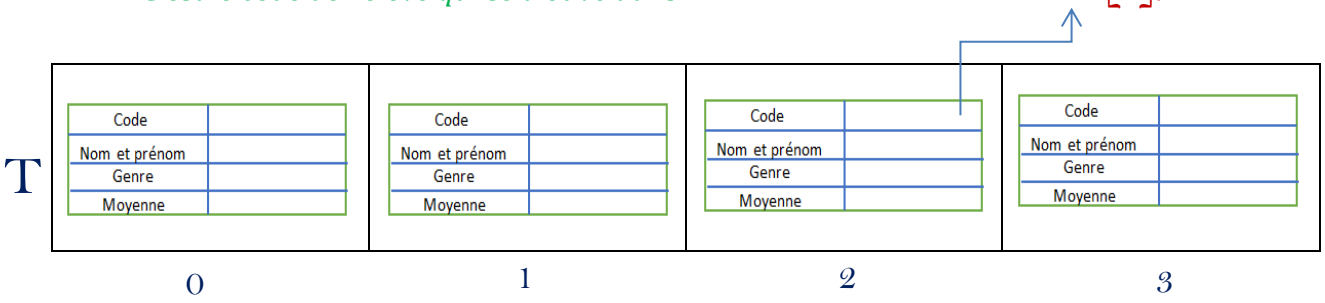
EXEMPLE

Soit l'enregistrement **ELEVE** suivant formé de 4 champs :

Code	
Nom et prénom	
Genre	
Moyenne	

Le tableau T suivant est un vecteur formé par 4 enregistrements ELEVE.

C'est le code de l'élève qui se trouve dans la case d'indice 2 du tableau T : **T[2].code**



Enoncé de l'exercice :

Soit l'enregistrement élève suivant :

ELEVE		
Champ	**Signification**	**Type**
c	Code	Entier
np	Nom	Chaine
mg	Moyenne Générale	Réel *(compris entre 0 et 20)*

On désire faire un programme qui permet de remplir un tableau T par les informations de N élèves (**avec 3≤N≤39**) puis d'afficher la liste des élèves admis par ordre de mérite (c'est-à-dire ordre décroissant basé sur la moyenne).
NB : pour faire le tri du tableau, utiliser le tri à bulles

Travail à faire :

1- Faire l'algorithme du programme principal.
2- Faire les algorithmes des modules envisagés.

Partie 2 :

Les Fichiers :
Texte et Binaires
(Algorithmique et Programmation)

Les Fichiers :

1- Présentation du problème

Les structures de données utilisées jusqu'à présent (entier, réel, chaine de caractères, tableau , matrice, enregistrement, …) ne peuvent pas garder les données de façon **permanente** c'est-à-dire une fois on redémarre l'ordinateur les données seront …….. Pour garder les données de façon permanente, il faut utiliser une nouvelle structure qui s'appelle ……..

2- Définition :

Un Fichier :

……
……
……

1. En programmation, on distingue deux types de fichiers :
   - Les fichiers ……
   - Les fichiers ……
2. Un fichier possède deux noms : un nom …… (interne) et un nom …… (externe)
3. Chaque fichier doit être enregistré sur un support de stockage physique (disque dur, flash, …) et peut contenir une grande quantité d'informations.

3- Les fichiers texte :

3.1 / Déclaration algorithmique d'un fichier texte:

En algorithmique

Tableau de Déclaration des Objets

Objet	Type / nature
nom_logique	Fichier Texte

Exemple

TDO

Objet	Type / nature
F	Fichier Texte

un fichier est comme un livre . il faut **l'ouvrir** pour **lire** , on peut prendre des notes (**écrire**) et il faut le **fermer** à la fin.

3.2 / Commandes de manipulation d'un fichier texte:

- **Ouverture :**

Mode d'ouverture :
- "r" : (read) .....
- "w" : (write) .....
- "a" : (append) .....

En algorithmique	En python
Ouvrir("chemin\nom_physique", nom_logique,"mode")	**nom_logique=open(" chemin\nom_physique ", "mode")**
Exemple : **Ouvrir("travail.txt", F ,"w")**	*Exemple :* .....

- **Fermeture :**

En algorithmique	En python
Fermer(nom_logique) *Exemple :* Fermer(F)	**nom_logique.close()** *Exemple :* .....

- **Lecture :**

En algorithmique	En python
Lire (nom_logique, ch) ***Exemple :*** **Lire(F, ch)**	**ch=nom_logique.read()** *Exemple :* .....

Lecture de la totalité d'un fichier

En algorithmique	En python
Lire_ligne (nom_logique, ch) ***Exemple :*** **Lire_ligne(F, ch)**	**ch=nom_logique.readline()** *Exemple :* .....

Lecture d'une seule ligne du fichier et placer le curseur sur la ligne suivante

- **Ecriture :**

En algorithmique	En python
Ecrire (nom_logique, ch) *Exemple :* Ecrire(F, ch)	**nom_logique.write(ch)** *Exemple :* .....

Ecriture de la chaine ch sans retour à la ligne

En algorithmique	En python
Ecrire_nl (nom_logique, ch) *Exemple :* Ecrire_nl(F, ch)	**nom_logique.write(ch +"\n")** *Exemple :* .....

Ecriture de la chaine ch puis retour à la ligne

- **est de Fin de fichier :** *retourne Vrai si le pointeur est à la fin du fichier sinon elle retourne Faux.*

En algorithmique	En python
Fin_fichier (nom_logique,) ***Exemple :*** **Fin_fichier(F)**	Pas de correspondance

Exercice 14 (Création, remplissage et affichage du fichier texte)

Le responsable du « Parc du Belvédère » de Tunis veut sauvegarder la liste des animaux du parc dans un fichier, pour cela il demande votre aide pour lui faire un programme nommé Zoo qui permet de :

- Créer un fichier nommé physiquement **« animaux.txt »** et logiquement **F**
- Sauvegarder dans le fichier **F** les noms de N animal sachant que **2<N<30** à raison d'un animal par ligne.
- Afficher à l'écran la liste des animaux. (Le nom d'un animal par ligne)

Travail demandé:

1- Faire l'algorithme du programme principal.
2- Faire les algorithmes des modules envisagés.
3- Donner le script Python correspondant.

Remarque

1- Lorsque les opérations sur un fichier sont terminées, il faut **le fermer** en utilisant la commande : **Fermer(nom_logique)**

2- En python on peut utiliser la commande suivante qui ferme le fichier automatiquement après utilisation :
with open (" chemin\nom_physique ", "mode") as nom_logique

4- Les fichiers de données (binaires):

4.1 / Déclaration algorithmique d'un fichier de données:

Déclaration algorithmique du type d'un fichier de données

En algorithmique	Exemple
Tableau de Déclaration des Nouveaux Types **TYPES** Nom_type1 = **Enregistrement** Champ1 : type1 Champ2, Champ3 : type2 … Champn : typen **Fin** Nom_type2 = fichier de Nom_type1 Nom_type3 = fichier de Type_élément	TDNT **TYPES** eleve = **Enregistrement** C : entier N, P : chaine G : caractère MG : réel **Fin** F_eleves = Fichier d'eleves F_entiers = Fichier d'entiers

Déclaration algorithmique d'un fichier binaire

En algorithmique	Exemple
Tableau de Déclaration des Objets **Objet** | **Type / nature** nom_logique | Nom_type	TDO **Objet** | **Type / nature** F | F_eleves V1, V2 | F_entiers

4.2 / Commandes de manipulation d'un fichier de données:

Mode d'ouverture :
- "rb" : Lecture (pointer au début)
- "wb" Ecriture (création)
- "ab" : Ecriture à la fin (ajout)

- **Ouverture :**

En algorithmique	En python
Ouvrir("chemin\nom_physique", nom_logique,"mode")	**nom_logique=open(" chemin\nom_physique ", "mode")**
Exemple : **Ouvrir("travail.dat", F ,"wb")**	*Exemple :* ……

- **Fermeture :**

En algorithmique	En python
Fermer(nom_logique)	**nom_logique.close()**
Exemple : **Fermer(F)**	*Exemple :* ……

- **Lecture :**

En algorithmique	En python
Lire (nom_logique, objet) ***Exemple :*** **Lire(f, x)**	**from pickle import *** **objet=load(nom_logique)** *Exemple :* …… ……

- **Ecriture :**

En algorithmique	En python
Ecrire (nom_logique, objet) ***Exemple :*** **Ecrire(F, x)**	**from pickle import *** **dump(objet, nom_logique)** *Exemple :* …… ……

- **Test de fin de fichier :**

En algorithmique	En python
Fin_fichier (nom_logique) ***Exemple :*** **Fin_fichier(F)**	Pas de correspondance

Remarque

Mode d'ouverture d'un fichier :

"r"	Valeur par défaut. Ouvrir le fichier en lecture. Erreur si le fichier n'existe pas.
"w"	Ouvrir le fichier en écriture. Si le fichier existe son contenu sera effacé sinon (le fichier n'existe pas) python le crée.
"a"	Ouvrir le fichier en mode ajout à la fin (append). Si le fichier n'existe pas python le crée.
"x"	Créer un nouveau fichier et l'ouvrir pour écriture. Erreur si le fichier existe.
On peut ajouter une deuxième lettre dans le mode d'ouverture pour spécifier le type de fichier.	
"t"	Par défaut, fichier Texte
"b"	Fichier binaire

Exercice 15 (Création, remplissage et affichage du fichier de données)

On vous demande de faire le programme python permettant de :
1- Remplir un fichier de données nommé physiquement « **nombres.dat** » et logiquement « **F** » par tous les nombres de 1 à 1000.
2- Afficher à l'écran tous les nombres du fichier F en mentionnant devant chaque nombre parfait le message « **est parfait** »

NB : un nombre est dit parait s'il est égal à la somme de tous ses diviseurs autre que lui-même.
Exemples de nombres parfaits : 6,28,496 en effet 6 est parfait car 6=1+2+3

Exercice 16 (Création, remplissage, tri et affichage d'un fichier d'enregistrements)

Soit l'algorithme « Tri_insertion » suivant permettant de trier par ordre croissant un tableau T rempli par N entiers.

```
Procédure Tri_insertion (@T : TAB , N : entier)
Début
    Pour i de 1 à N-1 faire
    |   V ← T[i]
    |   j←i
    |   Tant que (T[j-1]>V) et (j>0) faire
    |   |   T[j]← T[j-1]
    |   |   j← j-1
    |   Fin tant que
    |   T[j]← V
    Fin pour
Fin
```

Essayez de comprendre le principe de la procédure tri_insertion

On vous demande de faire le programme permettant de :
1 - Remplir un fichier F « **classe.dat** » par les informations de N élèves sachant que **2<N<10** et que chaque élève est un enregistrement présentant les données suivantes :

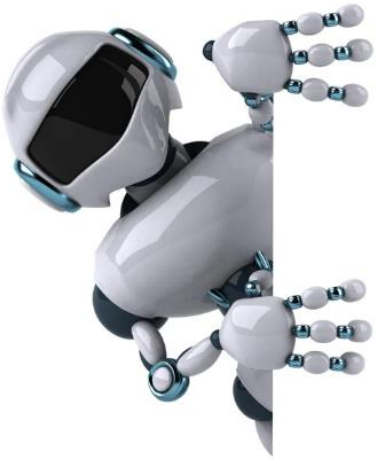
ELEVE

Champ	Signification	Type
id	Identifiant	Entier
no	Nom	Chaine formée par des lettres alphabétiques
pr	Prénom	Chaine formée par des lettres alphabétiques
mg	Moyenne Générale	Réel compris entre 0 et 20

2- Trier le contenu du fichier par ordre décroissant en utilisant le tri par insertion.
3- Afficher les élites de la classe (les élèves ayant une moyenne générale ≥15) par ordre de mérite.

NB : pour faire le tri du fichier il faut suivre les étapes suivantes :
1- Transférer le contenu du fichier vers un tableau d'enregistrements.
2- Trier le contenu du tableau T par ordre décroissant
3- Transférer le contenu du tableau trié vers le fichier.

Travail à faire :
1- Donner l'algorithme du programme principal
2- Donner les algorithmes des modules envisagés



Partie 3 :

Les algorithmes de tri : (Algorithmique et Programmation)

Les algorithmes de tri :

http://lwh.free.fr/index.html

1- Définition :

Un algorithme de tri est une suite d'instructions servant à ..... (ranger) une séquence d'éléments de façon ..... ou .....

Exemples :

- Séquence triée : 12 – 15 – 88 – 121 – 122 – 714 – 901 – 2510
- Séquence non triée : 15.5 – 19 – 10 - 201 - 3 – 155

Nombreux algorithmes de tri existent voici quelques exemples :
Tri à bulles, Tri par sélection, Tri par insertion, Tri Shell, Tri rapide, Tri à peigne, Tri fusion, Tri comptage, Tri Gnome, Tri stupide, ...

2- Tri à bulles :

2.1/ Principe (tri croissant d'un tableau T de N cases) :

Cette méthode consiste à faire remonter le plus grand élément du tableau en comparant les éléments successifs elle fonctionne comme suit :

1- On compare le premier **pair d'éléments T[0] et T[1]**.
2- Si T[0] > T[1] alors on permute T[0] et T[1], aller au pair suivant et répéter les étapes 1 et 2 jusqu'à comparer le dernier pair T[N-2] et T[N-1]. A la fin de ce premier parcours on aura passé le plus grand élément du tableau vers sa place finale qui est le (N-1)ième case du tableau.
3- On recommence cette opération en parcourant de 0 à N-2 puis de 0 à N-3 et ainsi de suite.
4- On arrête le traitement si on arrive au dernier élément du tableau ou le tableau devient trié.

2.2/ Algorithme de tri à bulles :

```
Procédure Tri_bulles (@ T :TAB , N : entier)
DEBUT
   Répéter
      test ← Faux
      Pour i de 0 à N-2 faire
      |     Si T[i] >T[i+1] alors
      |     |    aux← T[i]
      |     |    T[i] ← T[i+1]
      |     |    T[i+1]← aux
      |     |     test ← Vari
      |     Finsi
      Fin pour
      N ← N-1
   Jusqu'a (N =1) ou (test = Faux)
FIN
```

TDOL :

Objet	Type / Nature
aux, i	Entier
test	Booléen

2.3/ Autre version de tri à bulles :

```
Procédure Bubble_sort (@T: TAB , N : entier)
DEBUT
    Pour i de 0 à N-1 Faire
    |   Pour j de 0 à N-i-2 Faire
    |   |   Si T[j]>T[j+1] Alors
    |   |   |       aux ←T[j]
    |   |   |       T[j] ← T[j+1]
    |   |   |       T[j+1] ← aux
    |   |   Fin Si
    |   Fin Pour
    Fin Pour
FIN
```

TDOL

Objet	Type / nature
i,j, aux	Entier

Exercice 17 (Tri à bulles – Autre version)

Donner le script python du module Bubble_sort:

3- Tri par sélection :

3.1/ Principe (tri croissant d'un tableau T de N cases) :

Cette méthode consiste à :

1- Trouver l'indice ou la position (pm) du plus petit élément du tableau.
2- **Placer le plus petit élément (T[pm]) à sa position finale** (la première position)
3- Rechercher l'indice du second plus petit élément
4- Le placer à sa position finale (deuxième position)
5- Répéter le traitement précédent (3 et 4) jusqu'à ce que le tableau soit trié.

3.2/ Algorithme de tri par sélection :

```
Procédure Tri_sélection (@T: TAB , N : entier)
DEBUT
  Pour i de 0 à N-2 faire
  |     #on commence par chercher la position du minimum
  |      pm ← i
  |      Pour j de i+1 à N-1 faire
  |      |     Si (T[j] < T[pm]) alors
  |      |     |     pm ← j
  |      |     Finsi
  |      Fin pour
  |
  |     Si (pm ≠ i) Alors
  |     |     aux ← T[i]
  |     |      T[i] ← T[pm]
  |     |      T[pm]←aux
  |     Fin si
  Fin Pour
FIN
```

TDOL

Objet	Type / nature
i,j,pm, aux	entier

4- Tri par insertion :

4.1/ Principe (tri croissant d'un tableau T de N cases) :

1- On commence par le deuxième élément du tableau c'est-à-dire T[1].
2- On compare l'élément choisi (v) avec tous ses précédents dans la liste (la partie gauche du tableau) afin de **l'insérer dans la bonne position après avoir décalé tous les éléments** qui sont supérieurs à (v) vers la droite.
3- Répéter l'étape 2 pour l'élément suivant jusqu'à arriver au dernier élément du tableau.

4.2/ Algorithme de tri par insertion :

```
Procédure Tri_insertion (@T : TAB, N : entier)
  Début
    Pour i de 1 à N-1 faire
      v ← T[i]
      j←i

      Tant que (T[j-1]>v) et (j>0) faire
          T[j]← T[j-1]
          j← j-1
      Fin tant que
      T[j]← v
    Fin pour
  Fin
```

TDOL

Objet	Type / Nature
i,j,v	entier

Exercice 18 (Tri insertion en python)

Donner le script python du module Tri_insertion:

5- Tri shell :

5.1/ Présentation :

Le tri par shell est une amélioration de ..... proposée par **Donald L.Shell** en 1959.
Le tri par shell se base sur le calcul de **pas** (gap).

Question : Mais qu'est-ce que le pas et comment le calculer ?
Réponse : Donald Shell a proposé la formule (suite) suivante pour calculer le pas : (avec n est la taille du tableau)

$$\begin{cases} U_0 = 1 \\ U_{n+1} = 3*U_n + 1 \end{cases}$$

5.2/ Exemple de calcul de pas pour un tableau de 100 éléments :

Pour calculer le pas d'un tableau de taille N=100 on procède comme suit :

Etape1 : déterminer la valeur maximale de pas

Etape2 : Les valeurs du pas pour trier le tableau sont :

5.3/ Algorithme de tri shell :

```
Procédure Tri_shell (@ T :tab, N :entier)
 Début
    p←0
    Tant que (p<N) faire
    |    p← (3*p+1)
    Fin tant que

   Tant que (p≠0) faire
    |   p← p div 3
    |
    |   Pour i de p à N-1 faire
    |   |   v← T[i]
    |   |   j← i
    |   |
    |   |    Tant que (j>p-1) et (T[j-p]>v) faire
    |   |   |  T[j] ← T[j-p]
    |   |   |   j← j-p
    |   |   Fin tant que
    |   |
    |   |   T[j] ← v
    |   Fin pour
   Fin tant que
Fin
```

TDOL

Objet	Type / Nature
p,i,j,v	entier

Remarque

Le tri par insertion est un cas particulier du tri shell où le pas = 1

Exercice 19 (Tri shell)

Donner le script python permettant de trier (croissant) un tableau T déjà rempli pat N entiers en utilisant le tri par shell.

Exercice 20 (Tri par comptage)

Soit le principe suivant de tri par comptage qu'on peut appliquer sur un tableau T rempli par des entiers **distincts** afin de le trier par ordre croissant dans un deuxième tableau V:

Principe de tri comptage :

Pour chaque élément du tableau T :

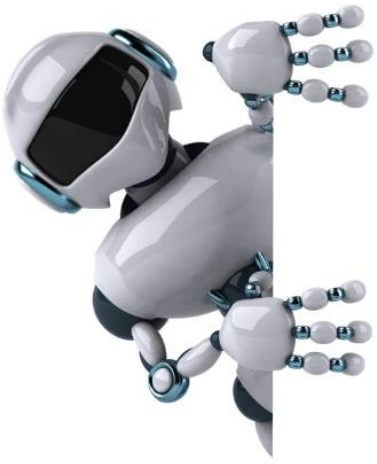
*1- Calculer le nombre **nb** d'éléments qui lui sont strictement inférieurs*

*2- Placer cet élément dans la position **nb** du tableau V*

1- Appliquer le principe précédent sur le tableau T suivant et remplir le tableau V.

T	**14**	**2**	**21**	**5**	**7**	**36**
V						

2- Donner l'algorithme d'un module Tri_comptage qui permet d'ordonner par ordre croissant les éléments d'un vecteur **T** de type **TAB** dans un deuxième tableau **V** de même type sachant que le vecteur **T** peut contenir **N** entiers.



Partie 4 :

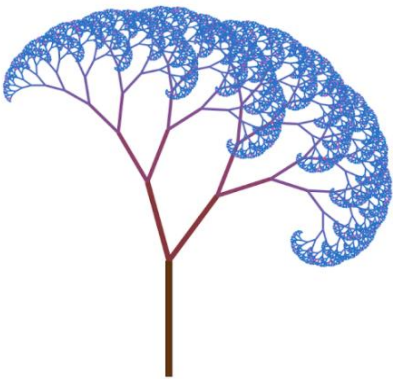
La récursivité :
(Algorithmique et Programmation)

La récursivité :

1- Définition :

La récursivité est une méthode algorithmique qui consiste à …… un sous-programme dans son propre corps.

Un sous-programme récursif est un module qui fait appelle à lui-même (jusqu'à ce qu'une condition d'arrêt soit vérifiée), à chaque appel, il y aura …… d'une nouvelle valeur d'un même paramètre formel.



Un programme récursif doit avoir :

- un (ou plusieurs) point d'arrêt (condition(s) de sortie)
- un (ou plusieurs) appel récursif.

2- Itératif vs récursif :

EXEMPLE

La factorielle d'un entier positif N notée N!=1*2*…*N
Pour calculer la factorielle d'un nombre N on peut utiliser une méthode itérative ou récursive.

Algorithme itératif de la fonction factoriel :

Algorithme récursif de la fonction factoriel :

3- Application de la récursivité:

Exercice 21 (Somme récursive)

1- Donner l'algorithme de la fonction récursive nommée « som » qui prend en paramètre un entier positif N pour calculer et renvoyer la somme des N premiers entiers .
2- Donner le script python correspondant à la fonction développée.

EXEMPLE

Pour N=6 , la somme des 6 premiers entiers est 1+2+3+4+5+6 = 21

Exercice 22 (Somme des chiffres d'une chaine)

1- Donner l'algorithme de la fonction récursive nommée « som_chif » qui prend en paramètre une chaine de caractères ch pour calculer et renvoyer la somme de ses chiffres .
2- Traduire votre fonction en python.

EXEMPLE

Pour ch= « Bac2023 » , la fonction doit retourner la valeur 7

Exercice 23 (Puissance de deux nombres)

On désire faire l'algorithme de la fonction nommée « **puis** » qui prend en paramètre deux entiers positifs x et y pour calculer et renvoyer la valeur de x à la puissance y (x^y)

1- Donner une version itérative de la fonction
2- Donner une version récursive de la fonction.

EXEMPLE

Pour x=2 et y=4 , la fonction doit retourner la valeur $2^4 = 16$

Exercice 24 (Somme d'un tableau)

Donner l'algorithme de la fonction récursive nommée « **som_tab** » qui prend en paramètre un tableau T et sa taille N pour calculer et renvoyer la somme des éléments qui se trouvent dans le tableau.

EXEMPLE

Pour le tableau T suivant de taille N= 6, la fonction doit retourner 72

12	10	11	3	4	32
0	**1**	**2**	**3**	**4**	**5**

Exercice 25 (Suite de Fibonacci)

La suite de Fibonacci est une suite très connue définie comme suit : $\begin{cases} F_0 = 0 \\ F_1 = 1 \\ F_n = F_{n-2} + F_{n-1} \quad \text{pour } n \geq 2. \end{cases}$

Les premiers termes de la suite de Fibonacci :

F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}	...	F_n
0	1	1	2	3	5	8	13	21	34	55	89	144	233	377	610	987	...	$F_{n-1} + F_{n-2}$

On vous demande de donner la fonction récursive en python nommée « **fibo** » qui prend en paramètre un entier N pour renvoyer le Nième terme de la suite de Fibonacci.

Exercice 26 (Espaces superflus)

Donner l'algorithme de la fonction récursive nommée « **nettoyage** » qui prend en paramètre une chaine et supprime tous les espaces superflus qui se trouvent dans la chaine.

EXEMPLE

Chaque point désigne un espace

Si ch= "Anis . . . ELBAHI . is . . . a . Python . . . Tutor"

La fonction retourne : "Anis . ELBAHI . is . a . Python . Tutor"

Exercice 27 (Chaine palindrome)

Faire le programme python permettant de :

1- Saisir une chaine formée au moins par 3 caractères
2- Vérifier et afficher si la chaine saisie est palindrome ou non.

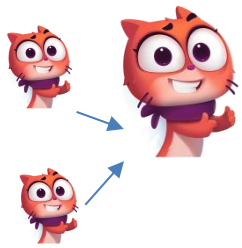
NB : pour vérifier si la chaine est palindrome ou non , développer une fonction récursive.

Partie 5 :

Les algorithmes récurrents : (Algorithmique et Programmation)

Les algorithmes récurrents:

1- Définition :



Un algorithme ou un traitement est dit récurrent s'il utilise un procédé itératif ou récursif pour donner un résultat qui peut dépendre de **un ou plusieurs résultats précédents**.

Remarque

Un algorithme récurrent d'ordre n est un algorithme donnant un résultat qui dépend de n résultats précédents.

1- Exemples de traitements récurrents :

Le calcul de la somme , de la factorielle, de la puissance, du nombre, d'un terme d'une suite, la recherche du maximum, le triangle de Pascal, la suite de Fibonacci, le nombre d'or, … sont des exemples de traitements récurrents.

2.1/ Calcul de la somme :

Exercice 28 (Somme matrice)

1- Soit **M** une Matrice de type **Mat** formée par **L** lignes et de **C** colonnes déjà remplie par des entiers. On demande de faire l'algorithme de la fonction qui calcule et retourne la somme de la matrice **M**.
2- Quel est l'ordre de récurrence du traitement développé dans la question précédente ?

2.2/ Traitement récurrent sur les chaines :

Exercice 29 (Suite de Thue-Morse)

C'est une suite **binaire** définie par : $U_0 = 0$ ou $U_0 = 1$ et par la récurrence suivante : pour passer de U_n à U_{n+1} on remplace tous les 0 de U_n par 01 et tous les 1 par 10.

EXEMPLE

U_0=0 (ou U_0=1)
U_1=01
U_2=01 10
U_3=01 10 10 01

Exemple d'exécution :

```
>>> %Run thue_morse.py
donner un chiffre binaire : 0
donner le nombre de termes : 4
0
01
0110
01101001
```

Travail demandé :

1- Sachant que U_0=0, Calculer le terme U4 de la suite Thue-morse

……

2- Quelle est l'ordre de récurrence de cette suite ?

……

3- Faire le programme Python qui permet de :

- Saisir le premier terme de la suite de Thue Morse (qui peut être 0 ou 1).
- Saisir N , le nombre de termes à calculer (avec N > 2).
- Calculer et afficher les N premiers termes de la suite Thue-morse

2.3/ Triangle de Pascal :

Le triangle de pascal est la matrice des coefficients (coefficients du binôme) qui sont utilisées pour développer des expressions comme $(a+b)^0$, $(a+b)^1$, … , $(a+b)^n$

Exemple :

$(a+b)^0 = 1$ **les coefficients : 1**

$(a+b)^1 = 1*a + 1*b$ **les coefficients : 1,1**

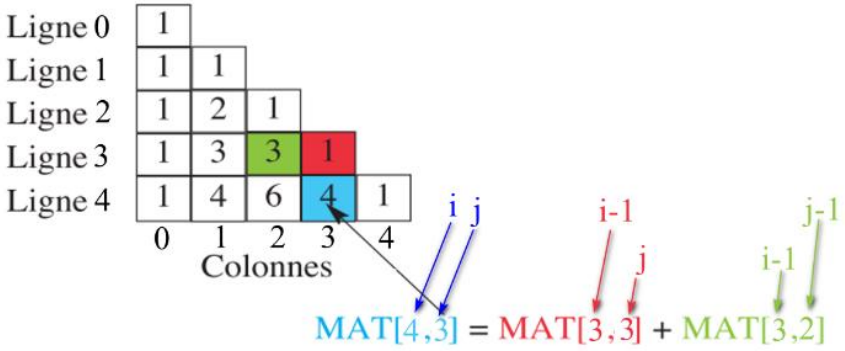
$(a+b)^2 = 1*a^2 + 2*a*b + 1*b^2$ **les coefficients : 1,2,1**

$(a+b)^3 = 1*a^3 + 3*a^2*b + 3*a*b^2 + 1*b^3$ **les coefficients : 1,3,3,1**

$(a+b)^4 = 1*a^4 + 4*a^3*b + 6*a^2*b^2 + 4*a*b^3 + 1*b^4$ **les coefficients : 1,4,6,4,1**

Calculer les coefficients pour $(a+b)^5$ ➔ ……

Pour N=5, le triangle de pascal affiché est le suivant:



On constate que le calcul de MAT[4,3] fait référence à deux résultats précédents : MAT[3,3] et MAT [3,2] donc il s'agit d'un traitement récurrent d'ordre 2.

Comment obtenir un triangle de pascal ?

De façon générale le triangle de pascal (d'ordre N) s'obtient de la façon suivante :

- Initialiser une matrice carré M (d'ordre N) par la valeur 0
- La première colonne de M doit contenir dans chaque cellule, la valeur 1.
- Les autres éléments sont déterminés en appliquant la formule suivante :

M[i,j] = M[i-1,j] + M[i-1,j-1] (avec i =numéro de la ligne et j =numéro de la colonne)

Exercice 30 (Triangle de Pascal)

On se propose d'afficher les N premières lignes du triangle de Pascal (avec 3 ≤ N <20). On vous demande de faire le programme python permettant de remplir et d'afficher les N premières lignes du triangle de Pascal comme le montre l'exemple suivant :

Exemple d'exécution :

```
>>> %Run 'triangle de pascal.py'
 donner N : 5
1  0  0  0  0
1  1  0  0  0
1  2  1  0  0
1  3  3  1  0
1  4  6  4  1
```

2.4/ Suite de Fibonacci :

Leonardo Pisano, connu sous le nom de Fibonacci, étudia la reproduction des lapins (du point de vue numérique). Il a remarqué qu'un couple de jeunes lapins met une saison pour devenir adulte, attend une autre saison puis met au monde un couple de jeunes lapins à chaque saison suivante. Et il a conclu que la reproduction des lapins peut être représentée par la suite suivante :

$$\begin{cases} U_1 = 1 \\ U_2 = 1 \\ U_n = U_{n-1} + U_{n-2} \end{cases}$$

Exercice 31 (Suite de Fibonacci)

1- Quelle est l'ordre de récurrence de la suite de Fibonacci ? Réponse : ……

2- Donner les valeurs correspondantes aux termes suivants :

U_1	U_2	U_3	U_4	U_5	U_6	U_7
1	1					

2- Pour un entier N donné, donner l'algorithme de la fonction « fibo » qui calcule et renvoie le Nième terme de la suite de Fibonacci :

a- en utilisant un traitement itératif

b- en utilisant un traitement récursif

2.5/ Nombre d'Or :

Exercice 32 (Nombre d'or)

1- Compléter le tableau suivant :

n	Fib (n)	Fib (n+1) / Fib (n)	Valeur approchée de Fib (n+1) / Fib (n)
1	1	1	1
2	1	2	2
3	2	3/2	1,5
4	3	5/3	1,666
5	5	8/5	1,6
6	8	13/8	1,625
7	13	21/13	1,615
8	21		
9	34		
10	55		
11	89		
12	144		
13	233		

constatation

Il semble que **Vn = Fib(n+1) / Fib(n)** converge vers **1,618**. Donc la suite de Fibonacci et la valeur **1,618** sont fortement liées. La valeur à laquelle converge **Vn** s'appelle **le nombre d'or**

Travail demandé :

Soient deux suites U et V définies comme suit :

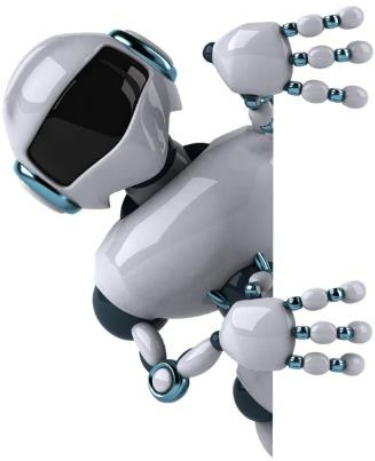
$$\begin{cases} U_1 = 1 \\ U_2 = 1 \\ U_n = U_{n-1} + U_{n-2} \text{ pour } n \geq 3 \end{cases}$$

et

$$\begin{cases} V_1 = 1 \\ V_n = U_n / U_{n-1} \text{ pour tout } n \geq 2 \end{cases}$$

La suite V_n tend vers une limite appelée nombre d'or.
On suppose que le nième terme de la suite V est égal à V_n.
V_n est la valeur approchée du nombre d'or avec une précision e dès que $|V_n - V_{n-1}| \leq e$.

On désire faire le programme python qui cherche V_n à 10^{-4} près et son rang.



Partie 6 :

Les algorithmes arithmétiques : (Algorithmique et Programmation)

Les algorithmes arithmétiques:

1- Définition :

L'arithmétique est une branche des mathématiques qui étudie…… …… et se définit par « la science des nombres ».

Remarque

Le test de parité, le test de primalité , l'étude des nombres parfaits, le calcul de factoriel, le calcul de PGCD, le calcul de PPCM, l'étude de l'arrangement et de la combinaison, les règles de divisibilité, la conversion entre les bases de numération, … sont des exemples d'études arithmétiques.

2- Exemple 1 : Test de primalité

Remarque

2,3,5,7,11,13,17, … sont premiers
1 est non premier

Un nombre est dit premier s'il n'a que deux diviseurs distincts 1 et lui-même.

Exercice 33 (Nombre premier)

Donner l'algorithme d'une fonction booléenne nommée « **Premier** » qui prend en paramètre un entier x, et vérifie s'il est premier ou non.

3- Exemple 2 : Calcul de factoriel

La factorielle d'un entier positif N noté N! est calculée comme suit : N!=1*2*3*…* N

Remarque

0! = 1

Exercice 34 (Factorielle)

1- Calculer 5 ! puis 6 ! puis 0 !
2- Donner l'algorithme d'une fonction nommée « **Fact** » qui prend en paramètre un entier x et calcule et renvoie sa factorielle :
   - En utilisant un traitement itératif
   - En utilisant un traitement récursif .

4- Exemple 3 : Calcul de puissance

X à la puissance Y s'écrit X^Y par exemple $2^4 = 2*2*2*2 = 16$

Exercice 35 (Puissance)

1- Calculer 5^2 puis 3^5 puis 8^0
2- Donner l'algorithme d'une fonction nommée « **Puis** » qui prend en paramètre deux entiers positifs X et Y pour calculer et retourner la valeur de X à la puissance Y (X^Y)
   - En utilisant un traitement itératif
   - En utilisant un traitement récursif .

5- Exemple 4 : Nombre parfait

Un nombre est dit parfait s'il est égal à la somme de ses diviseurs autres que lui-même.
Par exemple 6 est un nombre parfait car les diviseurs de 6 sont 1,2,3 et 6 et puisque 6=1+2+3 donc 6 est un nombre parfait.

Exercice 36 (Nombre parfait)

1- Est-ce que 10 est un nombre parfait ?
2- Donner l'algorithme d'une procédure nommée « **parfait** » qui prend en paramètre un entier N et vérifie et affiche s'il est parfait ou non.

6- Exemple 5 : Calcul de PGCD

Pour calculer le PGCD (Plus Grand Commun Diviseur) de deux entiers positifs X et Y on peut utiliser la méthode de différence qui se base sur le principe suivant :

Principe :

Si X=Y alors leurs PGCD est égal à X ou Y

Sinon Si X>Y alors soustraire Y de X Sinon soustraire X de Y jusqu'à X devient égale à Y.

Exercice 37 (PGCD : méthode de différence)

1- Calculer le PGCD(120,50) en utilisant la méthode de différence
2- Donner l'algorithme d'une fonction nommée « **PGCD** » qui prend en paramètre deux entiers X et Y pour calculer et retourner leurs PGCD.

Exercice 38 (Tournage à la main)

Soit la fonction Quoi suivante :

```
def Quoi(x,y):
    if y==0:
        return x
    else:
        return Quoi(y, x%y)
```

1- Faire le tournage à la main de la fonction Quoi pour le couple suivant : X=120 et Y=50
2- Déduire le rôle de la fonction Quoi.

7- Exemple 6 : Calcul de PPCM

Pour calculer le PPCM (Plus Petit Commun Multiple) de deux entiers positifs X et Y on peut utiliser la méthode suivante :

Principe :
Si X=Y alors leurs PPCM est égal à X ou Y
Sinon Si X≠Y alors on calcule R= X * 1 et vérifie si le résultat obtenu est divisible par Y si oui alors R est le PPCM sinon on calcule de nouveau R = X *2 et on vérifie si le résultat obtenu est divisible par Y si oui R est le PPCM sinon on refait la même opération.

Exercice 39 (PPCM)

1- Calculer le PPCM(5,6) en utilisant le principe décrit précédemment.
2- Donner l'algorithme d'une fonction nommée « **PPCM** » qui prend en paramètre deux entiers X et Y pour calculer et retourner leurs PPCM.

8- Exemple 7 : Décomposition en facteurs premiers

Exercice 40 (Bac pratique 2009 # 21/05/2009 à 14h30)

REPUBLIQUE TUNISIENNE MINISTERE DE L'EDUCATION ET DE LA FORMATION *** Examen du baccalauréat SESSION 2009	Section : Sciences de l'informatique Prof : Mr. Anis Elbahi Epreuve pratique Matière : Algorithmique et programmation Le Jeudi , 21 mai 2009 à 14h 30 Durée : 1H 30 Coef. : 0.75

Important :

1) *Dans le dossier Bac2009 de votre poste, créez votre dossier de travail portant le numéro de votre inscription (6 chiffres) et dans lequel vous devez enregistrer au fur et à mesure tous les fichiers solutions au problème posé. Vérifiez à la fin de l'épreuve que tous les fichiers que vous avez créés sont dans votre dossier.*
2) *Une solution modulaire au problème posé est exigée.*

Soit p et q deux entiers naturels tels que $10 < p < q < 100000$. On se propose d'enregistrer dans un fichier texte **decomp.txt** placé sur la racine du lecteur C la décomposition en facteurs premiers de tous les entiers compris entre p et q. La première ligne du fichier **decomp.txt** contiendra le naturel p suivi d'une espace, suivie du naturel q. Chacune des autres lignes contiendra l'entier à décomposer suivi du signe égal suivi de sa décomposition.

Exemple :

Si p=34 et q=36 le fichier decomp.txt contiendra les informations suivantes :

34 36

34=2 .17

35=5.7

36=2 .2.3.3

Ecrire un programme **Decomp.py** qui permet d'afficher à l'écran la décomposition en facteurs premiers de tous les entiers compris entre p et q et de remplir le fichier decomp.txt comme décrit précédemment.

Exercice 41 (Somme)

On désire faire l'algorithme d'un module nommé « **Somme** » qui permet de calculer et d'afficher la somme suivante sachant que x est un réel strictement positif et N est un entier strictement positif déjà saisis au niveau du programme appelant.

$$S = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^N}{N!}$$

9- Exemple 8 : Calcul de l'arrangement A_n^p et de la combinaison C_n^p :

Exercice 42 (Combinaison et arrangement)

Soit l'ensemble S={a,b,c} , calculer :

- L'arrangement de 2 éléments parmi 3 : A(2,3)
- La combinaison de 2 éléments parmi 3 : C(2,3).

9.1 / Calcul de A_n^p :

Présentation de l'arrangement :

APPROCHE

Ensemble	Choix	Nom
{a, b, c, d}	abcd, bcad, …	Arrangement de 4 parmi 4
	abc, bca, cda, cdb …	Arrangement de 3 parmi 4
	ab, ba, bc, cd, db	Arrangement de 2 parmi 4

- L'ordre a son importance
- Un élément choisi, ne peut plus être re-choisi

C'est un tirage sans remise

Sachant que 1≤p≤n, pour calculer l'arrangement de p éléments par n éléments, on utilise les formules suivantes : $A_n^p = n(n-1)(n-2)\ldots(n-p+1)$ Ou encore $A_n^p = \frac{n!}{(n-p)!}$

Exercice 43 (Calcul d'Arrangement)

On vous demande de donner l'algorithme de la fonction qui prend en paramètre deux entiers n et p sachant que n et p sont déjà saisies au niveau du programme appelant et vérifiant la condition 1≤p≤n, pour calculer et retourner l'arrangement A_n^p .

9.2 / Calcul de C_n^p :

Présentation de la combinaison :

APPROCHE

Ensemble	Choix	Nom
{a, b, c, d}	abcd	1 combinaison de 4 parmi 4
	abc, abd, acd, bcd	4 combinaisons de 3 parmi 4
	ab, ac, ad, bc, bd, cd	6 combinaisons de 2 parmi 4

- L'ordre n'a pas importance
- Un élément choisi, ne peut plus être re-choisi
  C'est un tirage sans remise

Exemple avec des chiffres

2 parmi 2	[1, 2]
2 parmi 3	[1, 2], [1, 3], [2, 3]
2 parmi 4	[1, 2], [1, 3], [1, 4], [2, 3], [2, 4], [3, 4]
2 parmi 5	[1, 2], [1, 3], [1, 4], [1, 5], [2, 3], [2, 4], [2, 5], [3, 4], [3, 5], [4, 5]
3 parmi 3	[1, 2, 3]
3 parmi 4	[1, 2, 3], [1, 2, 4], [1, 3, 4], [2, 3, 4]
3 parmi 5	[1, 2, 3], [1, 2, 4], [1, 2, 5], [1, 3, 4], [1, 3, 5], [1, 4, 5], [2, 3, 4], [2, 3, 5], [2, 4, 5], [3, 4, 5]

Pour calculer C_n^p, il faut appliquer la formule suivante : $$C_n^p = \frac{A_n^p}{p!} = \frac{n!}{p!(n-p)!}$$

Exercice 44 (Calcul de Combinaison)

1- Calculer C_5^2 = et C_5^3 =
2- Faire la fonction python qui calcule la combinaison de p éléments parmi n éléments d'un ensemble donnée. (Avec **0 ≤ p ≤ n**).
3- Tester le programme pour les valeurs suivantes :

C(0,5)	C(2,5)	C(4,5)	C(5,5)

Exercice 45 (Problème de binôme)

1- Calculer : $(2+3)^2$ = , $(4+2)^3$ = , $(1+4)^3$ =
2- Soient a et b deux réels et n un entier naturel non nul.
Ecrire un programme python permettant de vérifier la formule de binôme exprimée par l'égalité suivante :

$$(a+b)^n = C_n^0 a^n + C_n^1 a^{n-1} b^1 + C_n^2 a^{n-2} b^2 + \ldots + C_n^{n-1} a^1 b^{n-1} + C_n^n b^n$$

https://fr.wikipedia.org/wiki/Liste_de_critères_de_divisibilité

10- Exemple 9 : Règles de divisibilité :

10.1 / Définition :

Un entier **N** est dit divisible par un entier **M** si le reste de la division euclidienne de N par M, est égal à zéro, autrement dit **N mod M = 0**.
Exemple : 10 est divisible par 5, car 10 mod 5 = 0.

Une regèle de divisibilité est une séquence d'opérations simples qui permet de reconnaitre rapidement si un entier est divisible par un autre sans effectuer la division directe.

10.2 / Divisibilité par 2 :

Un entier est divisible par 2 si **son chiffre d'unités est divisible par 2**.

10.3 / Divisibilité par 3 :

Un entier est divisible par 3 si **la somme de ses chiffres est divisible par 3**.

Exercice 46 (Divisibilité par 3)

Donner le script python de la fonction booléenne qui prend en paramètre un entier x strictement positif pour vérifier s'il est divisible ou non par 3 en appliquant le principe précèdent.

10.4 / Divisibilité par 4 :

Un entier est divisible par 4 si **le nombre formé par ses deux derniers chiffres est divisible par 4** .

Exercice 47 (Divisibilité par 4)

1- Répondre par oui / non :
1204 est divisible par 4 →
12351 est divisible par 4 →

2- Donner la fonction qui vérifie si un entier naturel est divisible par 4 ou non.

10.5 / Divisibilité par 5 :

Un entier est divisible par 5 si **son chiffre d'unités est égal à 0 ou 5**.

Exercice 48 (Divisibilité par 5)

1- Répondre par oui / non :
   504 est divisible par 5 →
   1235 est divisible par 5 →

2- Donner la fonction qui vérifie si un entier naturel est divisible par 4 ou non.

10.6 / Divisibilité par 6 :

Un nombre est divisible par 6 si et seulement **s'il est divisible à la fois par 2 et par 3**.

10.7 / Divisibilité par 7 :

Un nombre est divisible par 7 si et seulement si la différence entre son nombre de dizaines et le double de son chiffre des unités l'est (divisible par 7). Si cette différence est négative, on peut la remplacer par sa valeur absolue. En répétant cette transformation jusqu'à obtenir un résultat strictement inférieur à 14, le nombre de départ est divisible par 7 si et seulement si le résultat final est 0 ou 7.

17 381 est divisible par 7 car
1738 – 2 × 1 = 1736,
173 – 2 × 6 = 161,
16 – 2 × 1 = 14 et
|1 – 2 × 4| = 7.

Exercice 49 (Divisibilité par 7)

Faire la fonction python qui vérifie si un entier naturel est divisible par 7 ou non en appliquant le principe précèdent.

11- Exemple 10 : Conversion entre les bases de numération

11.1 / Définition :

Un système de numération est une méthode de …… fondée sur une base de numération.
Si N est une base de numération, le système contiendra N chiffres allant de 0 à N-1.
Les 4 systèmes de numérations les plus utilisés sont : le décimal, le binaire, l'octal et l'hexadécimal.

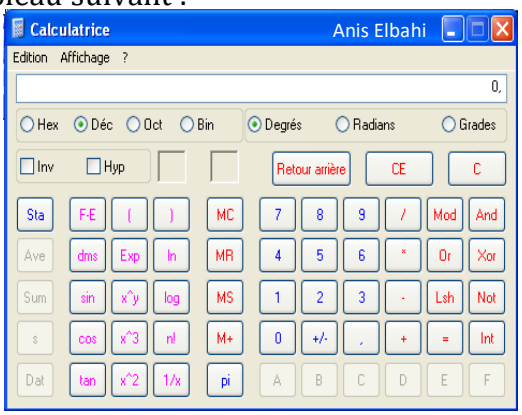
EXEMPLE

Base	Système de numération
Binaire (2)	0 et 1
Octale (8)	0,1,2,3,4,5,6 et 7
Décimal (10)	0,1,2,3,4,5,6,7,8 et 9
Hexadécimale (16)	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E et F

Exercice 50 (Conversion entre Bases de numération)

Lancer la calculatrice de votre ordinateur et compléter le tableau suivant :

Base 10	Base 2	Base 8	Base 16
10			
	100011		
		400	
		144	64



Remarque

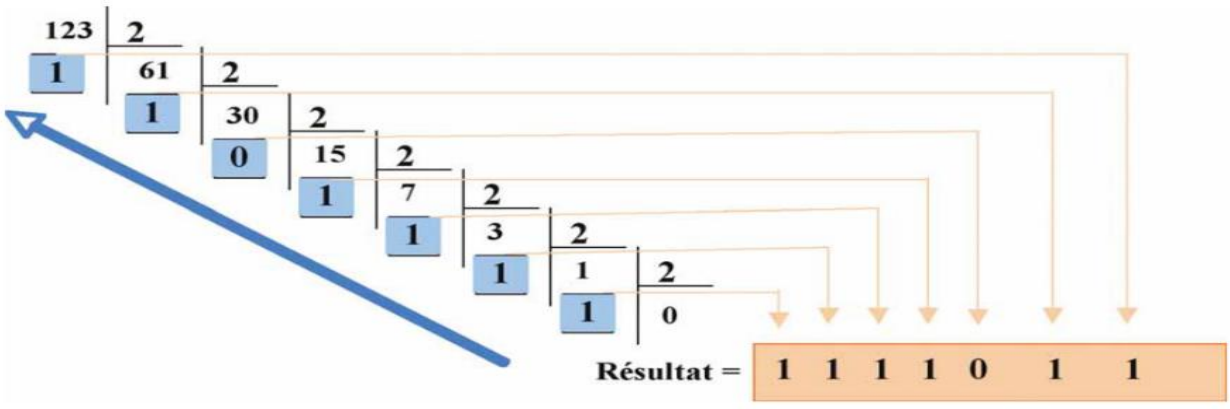
Il existe d'autres bases : base3, base5, base Shadok, …

11.2 / Conversion d'un nombre décimal vers une autre base:

a / Conversion d'un nombre décimal en nombre binaire :

Principe :

Pour convertir un nombre décimal X en nombre binaire, il suffit de diviser successivement X par 2 jusqu'à ce que le quotient obtenu soit égale à 0. Les restes de la division lus de droite à gauche représentent le nombre binaire.



Exercice 51 (B10→B2)

Donner l'algorithme de la fonction qui permet de convertir un nombre de la base 10 vers la base 2 en appliquant le principe précédent.

Remarque

Si vous voulez convertir un nombre décimal vers une autre base différente de la base 2, il faut suivre le même principe mais en faisant les divisions successives par le numéro de la base au quelle vous voulez faire la conversion.
Par exemple pour convertir un nombre décimale X vers la base octale (8), il faut faire la division successive de X par 8.

b / Conversion d'un nombre décimal en nombre hexadécimal :
pour convertir un nombre décimale X vers la base hexadécimale (16), il faut tenir compte des restes qui dépassent 9 et les replacer comme suit :

Reste de la division par 16	*Remplacer par*
10	**A**
11	**B**
12	**C**
13	**D**
14	**E**
15	**F**

Exercice 52 (B10→B16)

1- Convertir manuellement vers la base hexadécimale : $(5347)_{10} = (\quad)_{16}$
2- On se propose d'écrire l'algorithme d'une fonction qui permet de convertir un nombre décimal en son équivalent hexadécimal.

DONC : $(NOMBRE)_{10}$ → Division successive → $(NOMBRE)_{autre\ base}$

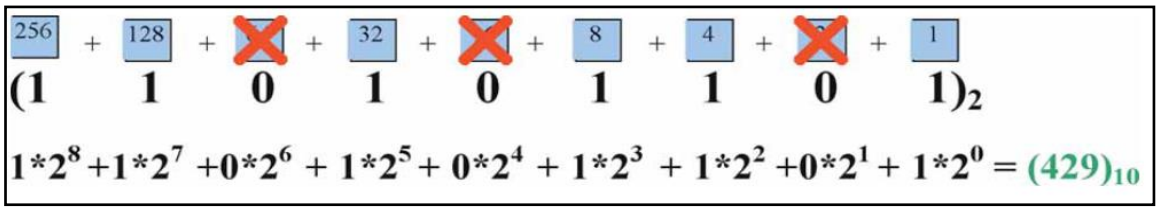
11.3 / Conversion d'un nombre d'une base vers la base décimale :

a/ Conversion d'un nombre binaire en nombre décimal :

Principe :

Pour convertir un nombre binaire à un nombre décimal, on peut appliquer le principe suivant :

1- Multiplier chaque chiffre binaire par 2 à la puissance de son poids.
2- Additionner les résultats trouvés.



Exercice 53 (B2→B10)

1- Convertir manuellement vers la base décimale : $(1001101)_2 = (\quad)_{10}$
2- Donner l'algorithme de la fonction qui permet de convertir un nombre binaire vers la base décimale.

b/ Conversion d'un nombre Hexadécimal en nombre décimal :

Exercice 54 (B16→B10)

1- Convertir manuellement vers la base décimale : $(1E5A)_{16} = (\quad)_{10}$
2- On se propose d'écrire un programme python qui permet de saisir un nombre hexadécimal puis d'afficher son équivalent en base décimale.

11.4 / Conversion d'un nombre de la base 16 vers la base 2 :

Principe

Chaque chiffre hexadécimal doit être remplacé par sa représentation binaire sous 4 chiffres.

Exercice 55 (B16→B2)

1- Convertir manuellement vers la base binaire : $(1E5A)_{16} = (\quad)_2$
2- On se propose d'écrire la fonction python permettant de convertir un nombre hexadécimal passé en paramètre en son équivalent binaire en utilisant le principe précèdent.

11.5 / Conversion d'un nombre de la base 2 vers la base 8 :

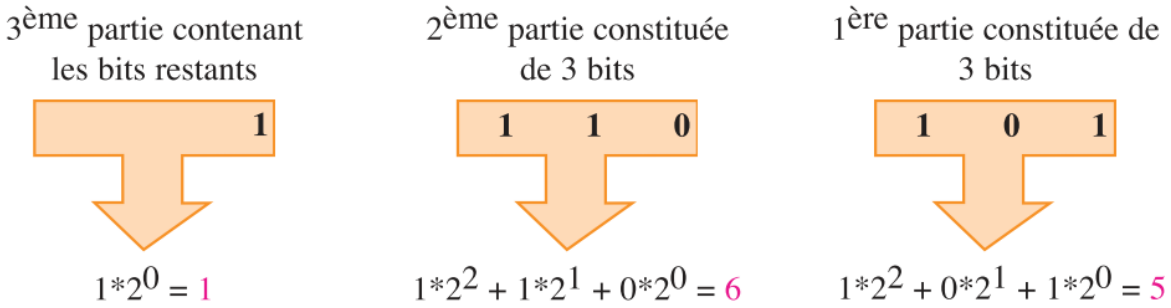
Principe

Nous proposons de convertir un nombre binaire en base 8.

Exemple :

Soit à convertir en octal le nombre binaire x = 1110101

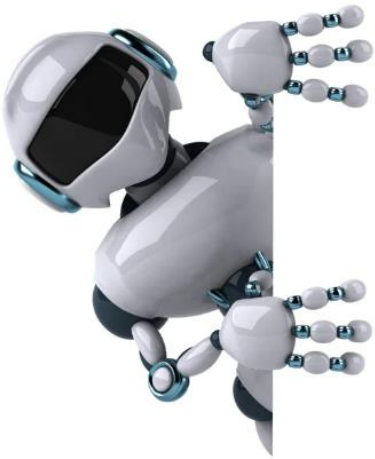
Nous allons procéder de la manière suivante :



Le nombre octal obtenu est : 165

Exercice 56 (B2→B8)

1- Convertir manuellement vers la base octale : $(101111011)_2 = (\ 573\ )_8$
2- On se propose d'écrire l'algorithme de la fonction qui prend en paramètre un nombre binaire pour le convertir en octal en utilisant le principe précèdent.



Partie 7 :

Les algorithmes d'approximation : (Algorithmique et Programmation)

Les algorithmes d'approximation:

1- Définition :

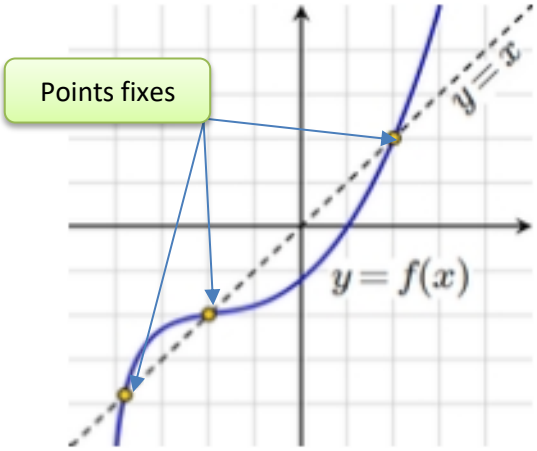
Un algorithme d'approximation est un algorithme qui consiste à déterminer une ..... dont on ne connait pas la solution exacte.

Approximation : (selon le dictionnaire)
Nom féminin :

- Sens1 = évaluation, estimation par à-peu-près.
- Sens2= valeur imprécise, valeur approchée.

2- Recherche d'un point fixe d'une fonction :

Soit E un ensemble et f :E→E une application.
On dit que x ∈ E est un point fixe de f si **f(x)=x**. graphiquement les points fixes d'une fonction f s'obtiennent en traçant la droite d'équation y=x. tous les points d'intersection de la courbe représentative de f avec cette droite sont alors les points fixes de f.



Certaines fonctions n'ont pas de points fixes : f(x)=x+1

Pour trouver le point fixe x0 d'une fonction f, il faut que la suite suivante : x, f(x), f(f(x)), f(f(f(x))), … converge vers x0.

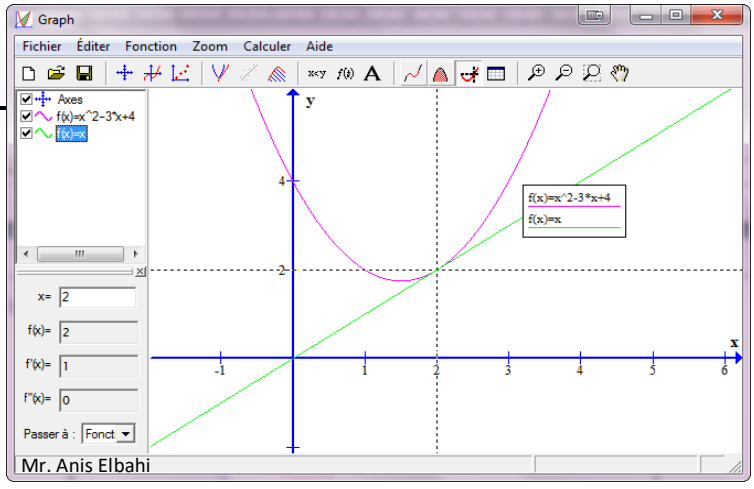
Exemple : f(x)= x^2-3x+4 ; 2 est un point fixe de f car f(2)=2

Exercice 57 (Point fixe d'une fonction)

1- Lancer le logiciel **graph** et vérifier le point fixe de la fonction précédente.

2- Donner le programme python qui permet de calculer et d'afficher le point fixe de la fonction **f(x)= x^2-3x+4** avec une précision epsilon = 10^{-5}

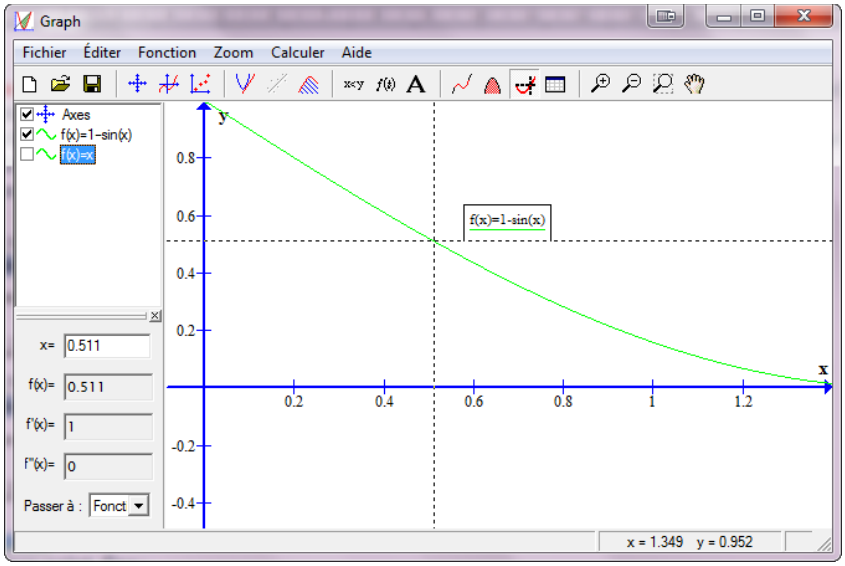
Remarque : x doit être ≥1 et <2



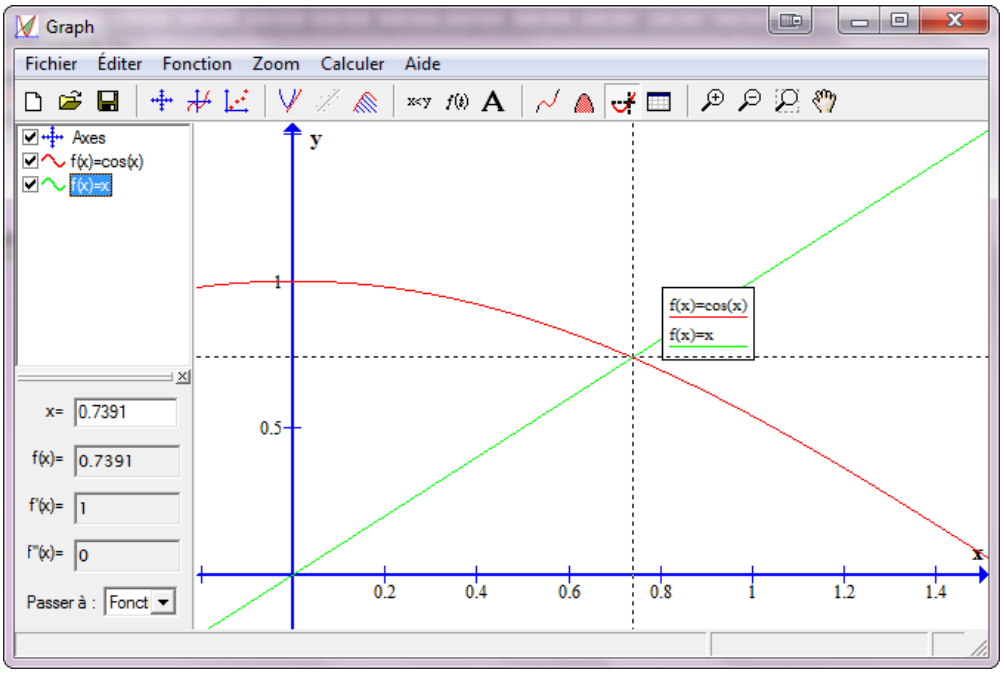
Exercice 58 (Point fixe « sin(x) »)

1- On vous demande de chercher graphiquement le point fixe des fonctions suivantes :

- **f(x) = 1 - sin(x)**



- **f(x) = cos(x)**



2- Donner le script python d'un module qui calcule et affiche une valeur approchée du point fixe de la fonction **f(x) = 1- sin(x)** à une précision epsilon (eps=10^{-5}).

3- Modifier votre programme python pour qu'il puisse afficher en plus de la valeur approchée, le nombre d'itérations nécessaires pour trouver cette valeur approchée à 10^{-5} près.

3- Calcul de valeurs approchées de constantes connues

Exercice 59 (Constantes célèbres)

1- Est-ce que la valeur exacte de Pi=3.14 ? →
2- Quelle est la valeur exacte de π ? →
3- Voici quelques exemples de constantes connues ainsi que leurs valeurs:

e	2,718...	Le nombre de neper
π	3,1416...	Le nombre π
g	9, 80665	L'accélération normale de la pesanteur

2.1 / valeur approchée du nombre de Neper (e) :

Le nombre e (donné par Euler), appelé aussi nombre exponentiel ou nombre de Neper est une constante représentant la base du logarithme népérien. C'est un nombre irrationnel (tronqué à 50 décimaux = 2,71828182845904523536028747135266249775724709369995...)

Exercice 60 (Nombre de Neper)

On désire calculer une valeur approchée du nombre de Neper à 10^{-4}, en utilisant la formule suivante :

$$e = 1 + \frac{1}{1} + \frac{1}{1\times 2} + \frac{1}{1\times 2\times 3} + \frac{1}{1\times 2\times 3\times 4} + \cdots = \sum_{n=0}^{+\infty} \frac{1}{n!}.$$

Donner l'algorithme d'un module permettant de résoudre ce problème.

2.2 / valeur approchée du nombre de Pi (π) :

Pour trouver une valeur approchée de, plusieurs techniques existent (Archimède est le 1er qui a calculé une valeur approchée de π).

Exercice 61 (Valeur approchée de π par la formule d'Euler)

Nous proposons de calculer une valeur approchée de π en utilisant la formule d'Euler :

$$\pi^2/6 = 1 + 1/2^2 + 1/3^2 + \ldots + 1/n^2 + \ldots$$

1- Faire le programme Python qui calcule et affiche la valeur de π en utilisant la formule précédente avec n une valeur donnée (n>1).
2- Tester votre programme pour les différentes valeurs de n suivantes.

Valeur de n	Valeur de π
2	
10	
100	
698	
699	
700	
948	
949	
950	

3- Nous proposons calculer une valeur approchée de π en utilisant la formule d'Euler à 10^{-7} près (c'est-à-dire lorsque la différence entre deux termes consécutifs devient inférieure ou égale à 10^{-7}. Le dernier terme calculé est la valeur de π recherchée.

Travail A Faire :
Faire le programme Python correspondant

Exercice 62 (Valeur approchée de e^x)

On demande de faire l'algorithme puis sa traduction en Python d'un module qui permet de calculer une valeur approchée de la valeur e^x en utilisant la formule suivante :

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

NB :

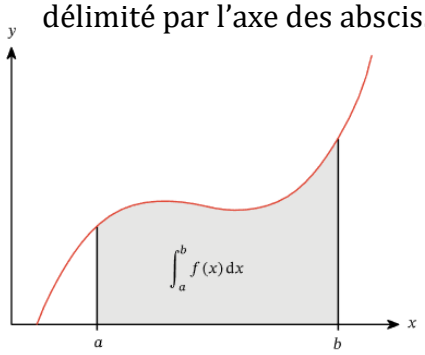
- Le calcul s'arrête si la différence entre deux termes devient ≤ epsilon donnée.
- x est un entier naturel non nul déjà saisi au niveau du programme appelant.
- Epsilon est une valeur saisie au niveau du programme appelant comprise entre 10^{-2} et 10^{-5}
- la valeur approchée recherchée est égale à la valeur du dernier terme calculé.

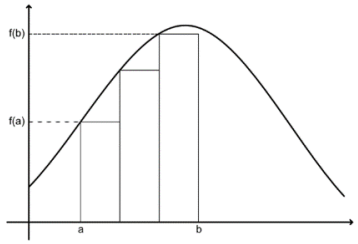
4- Calcul d'aires :

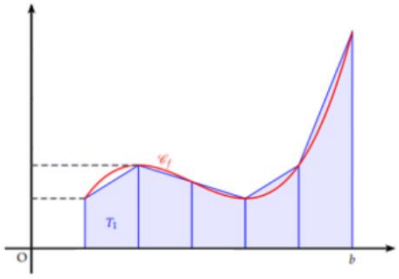
3.1 / Présentation :

Soit f une fonction continue et croissante dans l'intervalle [a,b].Si nous ne connaissons pas la primitive de la fonction f, nous ne pouvons pas calculer $\int_a^b f(t)dt.$
Mais nous pouvons chercher une valeur approchée.

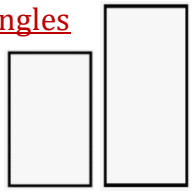
En mathématiques, l'intégrale d'une fonction réelle positive, est la valeur de l'aire du domaine délimité par l'axe des abscisses et la courbe représentative de la fonction.





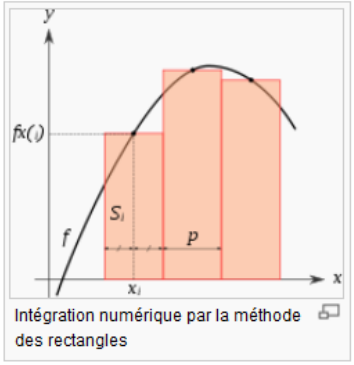


Dans ce qui suit nous allons voir deux méthodes pour calculer l'aire (surface):
la méthode des rectangles et la méthode des trapèzes.

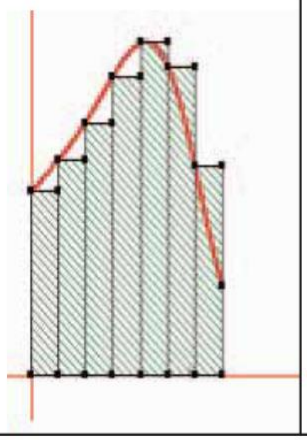
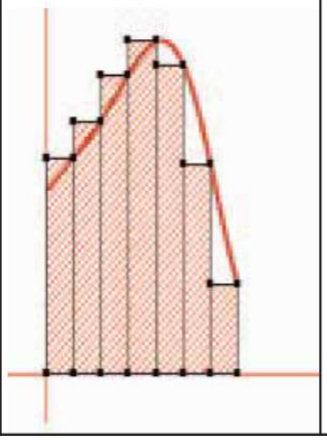
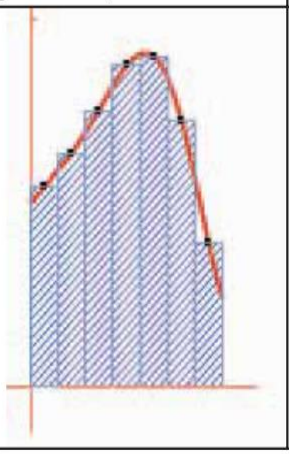


3.2 / Méthode des rectangles :

La méthode des rectangles consiste à remplacer l'aire sous la courbe par la somme des aires de rectangles obtenues.



Intégration numérique par la méthode des rectangles

Soit $f(a_i)$: Rectangles à gauche,	Soit $f(a_{i+1})$: Rectangles à droite,	Soit $f((a_i + a_{i+1})/2)$: Rectangles du point milieu,
		

On veut déterminer la valeur approchée de l'intégrale pour cela on va décomposer l'intervalle [a,b] en sous-intervalle de même largeur **(b-a)/n**

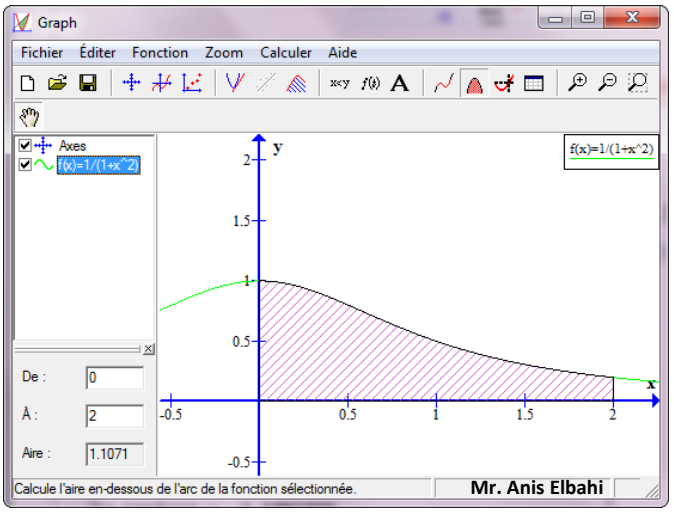
a=borne inferieure de l'intégrale
b=borne supérieure de l'intégrale
n=nombre de sous-intervalles

Exercice 63 (Calcul d'aire « méthode des rectangles »)

Nous proposons de calculer en utilisant la méthode des rectangles, l'aire résultante de la courbe de la fonction **f : x→ 1/ (1+x²)** sur l'intervalle [a,b]. (sachant que b>a≥0).

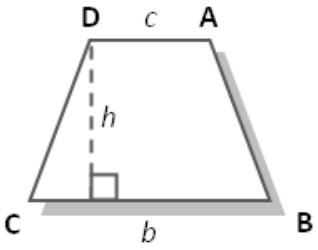
1- Lancer l'outil graph et calculer une valeur approchée de la fonction f(x) pour a=0 et b=2

2- Donner le programme Python qui permet de calculer et d'afficher une valeur approchée de $\int_a^b f(x)dx$ pour des valeur données de a,b et N



3.3 / Méthode des trapèzes :

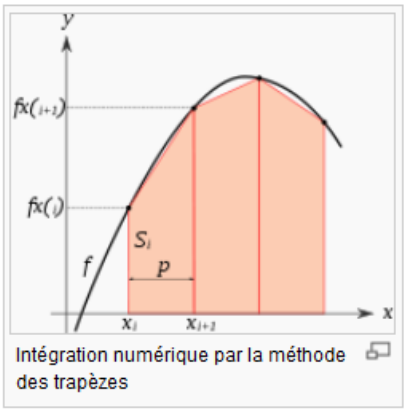
Calcul de l'aire d'un trapèze



DA = Petite base = c
CB = Grande base = b
Hauteur = h

L'aire A d'un trapèze (ou surface d'un trapèze) est égale à :

$$\text{Aire A} = \frac{(b+c)\,x\,h}{2}$$



Intégration numérique par la méthode des trapèzes

La méthode d'approximation d'une intégrale dite « des trapèzes » repose sur le calcul de l'aire d'un trapèze. On divise l'intervalle [a,b] comme dans le cas de la méthode des rectangles en n sous-intervalle de tailles égales comme suit : h=(b-a)/n

Rappelons que la surface d'un trapèze = (b+c) *(h/2)
Pour le calcul de la surface du premier trapèze de notre courbe il faut appliquer la formule suivante :

(f (a+h*1) + f(a+h*2)) *(h/2)

Donc la surface de chaque trapèze peut être définie par : Ai = (h/2) * (f(a+ih) + f(a+(i+1)h)).
Donc pour calculer l'aire de toute la surface, il suffit de calculer la somme de toutes les surfaces des trapèzes entre a et b.

Exercice 64 (Calcul d'aire « méthode des trapèzes»)

Refaire l'application précédente en utilisant la méthode des trapèzes.

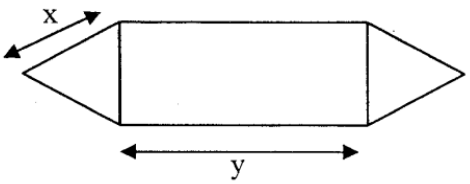
5- Problème d'optimisation :

Exercice 65 (Optimisation – Bac2009)

On veut réaliser une pièce métallique ayant la forme de la figure ci-contre, composée d'un rectangle et de deux triangles équilatéraux (x et y sont exprimés en mètres)
En déterminant le périmètre P de la pièce en fonction de x et y et en choisissant P=2m, nous aurons

$y = 1 - 2*x$ ce qui donne $0<x<\frac{1}{2}$.

On peut conclure donc que l'aire S de la pièce est égale à $S = ((\sqrt{3} - 4) / 2)*x^2 + x$.



Question :

On se propose de calculer une valeur approchée de x pour laquelle S est maximal. Faire l'algorithme d'un module qui permet de déterminer une valeur approchée de x à 10^{-2} près.