BUT INFORMATIQUE
IUT de Valenciennes
Antenne de Maubeuge

Ressource R1.01

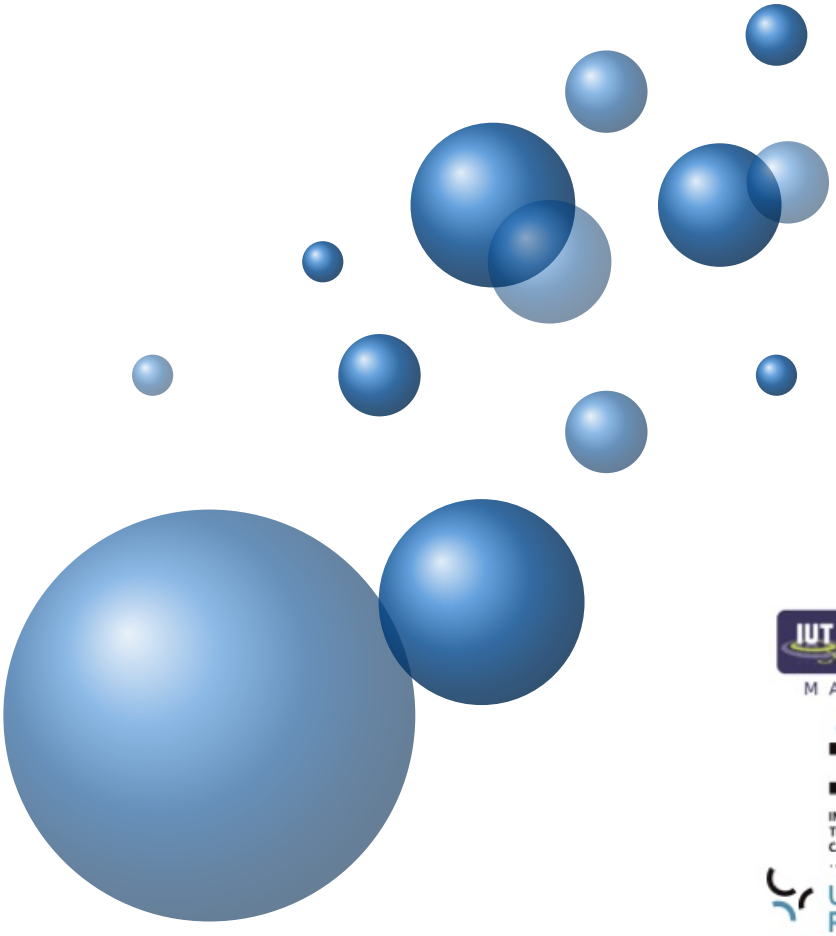
Introduction à l'algorithmique et à la programmation

Sujets de Travaux Pratiques

Philippe Polet

philippe.polet@uphf.fr

Année 2021 – 2022



Semestre 1

Avant-propos

Ce recueil compile les sujets de TP en R1.01 . Les TP pourront faire l'objet d'une préparation avant de venir en séance. L'environnement de développement préconisé, au début, sous linux (debian), est Thonny. Thonny est un outil de développement en python grandement utilisé pour l'apprentissage du langage. D'autres environnements pourront être utilisés par la suite.

Table des matières

Sujet de TP 1

1 Préparation de l'environnement de travail

Après vous être connecté sur le poste de travail (normalement login et mot de passe de l'ENT, sinon l'enseignant vous donnera un compte temporaire), lancer un terminal. La commande **ls** vous permet de visualiser les fichiers et répertoire de votre compte. Utiliser la commande **mkdir** pour créer le répertoire M1102.

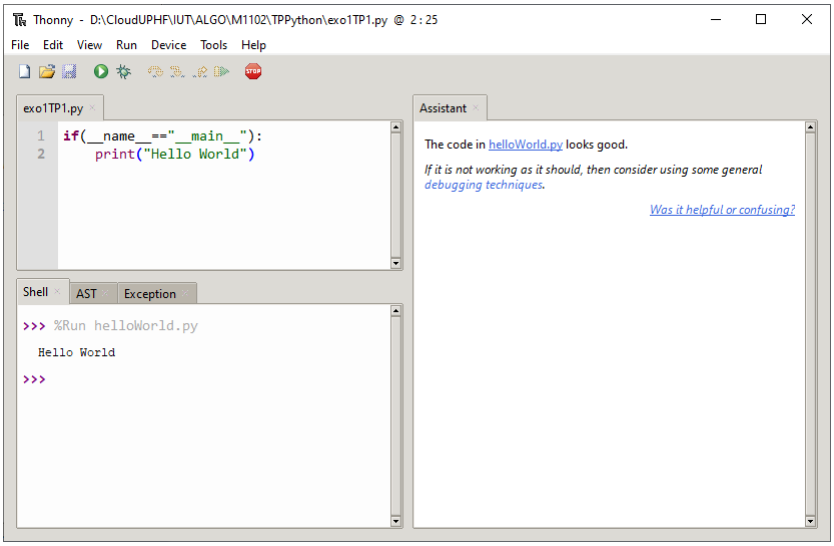
2 Votre premier script

Dans le terminal lancer *thonny* en tapant **thonny &**.

Vous allez maintenant créer un nouveau script Python. Choisissez dans le menu File–>new.

Tapez le code suivant :

```
if(__name__=="__main__"):
    print("Hello World")
```



Pour exécuter le script : cliquez sur la petite flèche verte ou dans le menu Run -> Run Current Script. Il vous est alors demandé de sauvegarder votre script : sauvegardez-le dans le répertoire M1102 sous le nom exo1TP1.py

Vous devriez voir le message Hello World apparaitre dans la console d'exécution de Thonny.

Après avoir vérifié que le programme fonctionnait correctement. Passez à l'exercice suivant.

3 Test du debogger

Créez comme pour le précédent exercice un nouveau script (exo2TP1.py). Tapez le code Python ci-dessous :

```
def f(a:int, b:int):
    c:int
    c=a
    a=b
    b=c

def prog():
    a:int
    b:int

    a=5
    b=10
```

```
    print("a= ",a," b= ",b)
    f(a,b)
    print("a= ",a," b= ",b)

if(__name__=="__main__"):
    prog()
```

Avant d'exécuter le programme essayez de deviner les valeurs qui seront affichées ! Pour pouvoir comprendre comment fonctionne un programme et suivre les valeurs des variables il faut utiliser un debogger. Assurez-vous que dans le menu "view" la vue "variable" est bien sélectionnée. Pour lancer le debogger cliquez sur l'insecte à côté de la flèche verte ou dans le menu Run–>Debug Current Script. Une fois le debogger lancé, exécutez pas à pas le script à l'aide de la flèche "step into". Chaque action est décomposée, vous pouvez constater l'évolution des variables dans leur contexte (cf. cours)

Il est évident que la fonction **f()** ne peut pas changer le contenu des variables de la fonction *prog()* ! On ne passe pas des "variables" en paramètre, on passe des valeurs (dans notre cas ces valeurs sont celles contenues dans les variable a et b de la fonction *prog()*). D'ailleurs, pour vous en convaincre, au lieu dans fonction *prog()* d'avoir $f(a, b)$ vous pouvez écrire $f(3, 6)$.

4 Exercices de la fiche de TD 1

Pour vous exercer codez, puis exécutez les algorithmes de la première fiche de TD.

Sujet de TP 2

L'objectif de cette séance est de travailler les branchements conditionnels (Si Alors Sinon) et les répétitions (Tant Que...).

1 Saisir une date valide

Nous allons pour la suite représenter une date par 3 entiers : un pour le jour, un pour le mois et le dernier pour l'année (de la forme 2016).

Une date est valide si elle existe, c'est à dire :

— le mois est bien compris entre 1 et 12,

— le jour est au moins égal à 1 et ne dépasse pas le nombre de jours du mois.

A Déterminer si une année est bissextile

Voici la définition donnée par Wikipédia d'une année bissextile :

> "*Depuis l'ajustement du calendrier grégorien, l'année sera bissextile (elle aura 366 jours) :*
> *— si l'année est divisible par 4 et non divisible par 100, ou*
> *— si l'année est divisible par 400.*
> *Sinon, l'année n'est pas bissextile (elle a 365 jours).*
>
> *Ainsi, 2016 est bissextile. L'an 2008 était bissextile suivant la première règle (divisible par 4). L'an 1900 n'était pas bissextile, car divisible par 100 mais non divisible par 400. L'an 2000 était bissextile car divisible par 400.*"

Écrire une fonction (**estBissextile**) qui admet un entier en paramètre (représentant une année) et qui retourne un booléen valant True si l'année est bissextile et False dans le cas contraire. Vous sauvegarderez ce script sous le nom de fichier : estBissextile.py

Il est important de tester systématiquement chaque fonction qu'on écrit. Pour tester votre fonction **estBissextile()**, vous pourrez écrire une fonction **testerBissextile** et le programme principal suivant :

```
from estBissextile import estBissextile
def testerBissextile():
    if(estBissextile(2016)):
        print("test de 2016 OK")
    else:
        print("test de 2016 ECHEC")
    if(estBissextile(2008)):
        print("test de 2008 OK")
    else:
        print("test de 2008 ECHEC")
    if(not estBissextile(1900)):
        print("test de 1900 OK")
    else:
        print("test de 1900 ECHEC")
    if(estBissextile(2000)):
        print("test de 2000 OK")
    else:
        print("test de 2000 ECHEC")

def prog():
    testerBissextile()

if(__name__=="__main__"):
    prog()
```

Vous remarquerez que pour importer la fonction *estBissextile*() du fichier estBissextile.py on met en début de script la ligne suivante :

```
from estBissextile import estBissextile
```

B Déterminer si un mois est valide

Écrire une fonction (**moisValide**) qui admet un entier en paramètre (représentant un mois) et qui retourne un booléen valant true si le mois est valide et false dans le cas contraire.
Écrire une fonction (**testerMoisValide**) qui, sur le modèle de la fonction **testerBissextile**, teste les différents cas (mois compris entre 1 et 12, mois plus petit que 1 et mois plus grand que 12).

C Déterminer le nombre de jours dans le mois

Écrire une fonction (**nbJours**) qui admet deux entier en paramètre (représentant un mois et une année) et qui retourne le nombre de jours dans le mois. L'année est importante pour le nombre de jour si le mois est 2 (si l'année est bissextile février comporte 29 jours, sinon 28).

Vous trouverez ci-dessous le code de la fonction permettant de tester la fonction **nbJours** :

```
def testerNbJours():
    if(nbJours(1,2000)==31):
        print("test de 01/2000 OK")
    else:
        print("test de 01/2000 ECHEC")
    if(nbJours(2,2000)==29):
        print("test de 02/2000 OK")
    else:
        print("test de 02/2000 ECHEC")
    if(nbJours(2,2001)==28):
        print("test de 02/2001 OK")
    else:
        print("test de 02/2001 ECHEC")
    if(nbJours(3,2000)==31):
        print("test de 03/2000 OK")
    else:
        print("test de 03/2000 ECHEC")
    if(nbJours(4,2000)==30):
        print("test de 04/2000 OK")
    else:
        print("test de 04/2000 ECHEC")
    if(nbJours(5,2000)==31):
        print("test de 05/2000 OK")
    else:
        print("test de 05/2000 ECHEC")
    if(nbJours(6,2000)==30):
        print("test de 06/2000 OK")
    else:
        print("test de 06/2000 ECHEC")
    if(nbJours(7,2000)==31):
        print("test de 07/2000 OK")
    else:
```

```
        print("test de 07/2000 ECHEC")
    if(nbJours(8,2000)==31):
        print("test de 08/2000 OK")
    else:
        print("test de 08/2000 ECHEC")
    if(nbJours(9,2000)==30):
        print("test de 09/2000 OK")
    else:
        print("test de 09/2000 ECHEC")
    if(nbJours(10,2000)==31):
        print("test de 10/2000 OK")
    else:
        print("test de 10/2000 ECHEC")
    if(nbJours(11,2000)==30):
        print("test de 11/2000 OK")
    else:
        print("test de 11/2000 ECHEC")
    if(nbJours(12,2000)==31):
        print("test de 12/2000 OK")
    else:
        print("test de 12/2000 ECHEC")
```

D Déterminer si une date est valide

Écrire une fonction (**dateValide**) qui admet 3 entiers en paramètre (représentant un jour, un mois et une année) et qui retourne un booléen valant true si la date est valide et false dans le cas contraire.

Vous utiliserez la fonction **testerDateValide**, dont on vous donne ci-dessous le code, pour vérifier que votre fonction est correcte.

```
def testerDateValide():
    if(dateValide(2,2,2000)):
        print("test de 02/02/2000 OK")
    else:
        print("test de 02/02/2000 ECHEC")
    if(not dateValide(32,1,2000)):
        print("test de 32/01/2000 OK")
    else:
        print("test de 32/01/2000 ECHEC")
    if(dateValide(29,2,2000)):
        print("test de 29/02/2000 OK")
```

```
else:
    print("test de 29/02/2000 ECHEC")
if(not dateValide(29,2,2001)):
    print("test de 29/02/2001 OK")
else:
    print("test de 29/02/2001 ECHEC")
if(not dateValide(-2,2,2000)):
    print("test de -2/02/2000 OK")
else:
    print("test de -2/02/2000 ECHEC")
```

E Programme principal

Écrire un programme principal qui demande à un utilisateur de saisir un jour, un mois et une année, qui vérifie si cela correspond à une date valide, si ce n'est pas le cas on répète l'opération. Une fois que les jour, mois et année sont corrects, le programme affichera combien de tentatives ont été nécessaire pour saisir une date valide.

Ci-dessous un exemple d'exécution du programme principal (avec affichage des tests) :

```
test de 2016 OK
test de 2008 OK
test de 1900 OK
test de 2000 OK
test de 1 OK
test de 12 OK
test de -1 OK
test de 13 OK
test de 01/2000 OK
test de 02/2000 OK
test de 02/2001 OK
test de 03/2000 OK
test de 04/2000 OK
test de 05/2000 OK
test de 06/2000 OK
test de 07/2000 OK
test de 08/2000 OK
test de 09/2000 OK
test de 10/2000 OK
test de 11/2000 OK
```

```
test de 12/2000 OK
test de 02/02/2000 OK
test de 32/01/2000 OK
test de 29/02/2000 OK
test de 29/02/2001 OK
test de -2/02/2000 OK
saisir le jour>29
saisir le mois>2
saisir l annee>2003
Il ne s agit pas d une date valide! veuillez ressaisir:
saisir le jour>32
saisir le mois>12
saisir l annee>2016
Il ne s agit pas d une date valide! veuillez ressaisir:
saisir le jour>31
saisir le mois>9
saisir l annee>2016
Il ne s agit pas d une date valide! veuillez ressaisir:
saisir le jour>31
saisir le mois>10
saisir l annee>2016
Bravo : la date : 31/10/2016 est valide! Nombre de tentatives :4
```

2 Pour aller plus loin

Écrire une fonction qui calcule le nombre de jours écoulés entre deux dates.

Sujet de TP 3

Ce TP a pour objectif de manipuler les chaînes de caractères et des listes avec pour application la conversion en binaire/hexadécimal/décimal.

Nous ne manipulerons que des **entiers positifs codés sur un octet**.

1 Questions de cours R1.03

Quelle est la plus grande valeur entière positive décimale qu'on puisse coder sur 1 octet ?
Quelle est sa valeur en binaire ?
Quelle est sa valeur en hexadécimal ?

Pour la suite on codera une valeur binaire par une liste de 8 entiers (bien évidement seules les valeurs 0 et 1 seront utilisés) ; on codera une valeur héxadécimale par une liste de 2 entiers et un nombre décimal sera codé par une liste de 3 entiers. Par exemple l'entier 13 sera représenté :

- en binaire par la liste d'entiers : [1,0,1,1,0,0,0,0]
- en hexadécimal par la liste d'entiers : [13,0]
- en décimal par la liste d'entiers : [3,1,0]

Le digit de poids faible étant toujours le premier élément de la liste

2 Saisie de nombres

Pour saisir un nombre binaire ou hexadecimal on passera par une chaîne de caractères.

- Écrire une fonction qui permet de saisir un nombre binaire, tant que le nombre n'est pas correct (formé de 8 caractères ('0' ou '1')) l'utilisateur est invité à ressaisir le nombre. La fonction renvoie une liste de 0 ou de 1. La fonction transformera donc la chaine de caractères en une liste de 8 entiers (de 0 à 1).

— Écrire une fonction qui saisit les valeurs d'un nombre hexadécimal selon le même principe. Mais cette fois, la chaîne est constituée de 2 caractères. Chaque caractère est alors l'un des suivants : 0,1,2,3,4,5,6,7,8,9 A,B,C,D,E,F. La fonction retournera une liste de 2 entiers (de 0 à 15).
— Écrire une fonction qui saisit un entier selon le même principe. L'entier doit être positif et inferieur strictement à 128. La fonction retournera une liste de 3 entiers (de 0 à 9).

Par exemple :

— Le nombre binaire saisi est 0111 (soit 7 en decimal) : la liste retournée (que nous appelerons l) par la fonction sera [1,1,1,0,0,0,0,0]. $l[0]$ correspond au bit de poids faible.
— Le nombre hexadécimal saisi est F (soit 15 en decimal) : la liste retournée (que nous appelerons l) par la fonction sera [15,0]. $l[0]$ correspond au digit de poids faible.
— Le nombre décimal saisi est 45 : la liste retournée (que nous appelerons l) par la fonction sera [5,4,0]. $l[0]$ correspond au digit de poids faible.

Ces 3 fonctions s'assureront qu'aucune erreur de saisie n'a été réalisée.

3 Affichage de nombres

— Écrire une fonction qui permet d'afficher un nombre binaire, passé en paramètre sous la forme d'une liste telle que décrite précédement. Par exemple, si on passe en paramètre la liste [1,0,0,1,0,0,1,0] la fonction affichera : 01001001
— Écrire une fonction qui permet d'afficher un nombre hexadécimal, passé en paramètre sous la forme d'une liste telle que décrite précédement. Par exemple, si on passe en paramètre la liste [1,12] la fonction affichera : C1
— Écrire une fonction qui permet d'afficher un nombre décimal, passé en paramètre sous la forme d'une liste telle que décrite précédement. Par exemple, si on passe en paramètre la liste [1,3,2] la fonction affichera : 231

Pour la suite, vous allez écrire des fonctions de conversion. Vous n'utiliserez aucune fonction de conversion de base du langage Python.

4 Conversion en entier

— Écrire une fonction qui convertit un nombre binaire passé en paramètre sous la forme d'une liste, en entier (int).
— Écrire une fonction qui convertit un nombre hexadécimal passé en paramètre sous la forme d'une liste, en entier (int).
— Écrire une fonction qui convertit un nombre décimal passé en paramètre sous la forme d'une liste, en entier (int).

5 Conversion de décimaux en binaire ou hexadécimal

— Écrire une fonction qui convertit un nombre décimal, passé sous la forme d'une liste, en binaire, également sous la forme d'une liste.
— Écrire une fonction qui convertit un nombre décimal, passé sous la forme d'une liste, en héxadécimal, également sous la forme d'une liste.
— Écrire une fonction qui convertit un nombre entier, passé en paramètre, en décimal, sous la forme d'une liste.

6 Programme principal

Écrire un programme qui demande à l'utilisateur de saisir un nombre (au choix de l'utilisateur : en décimal, en héxadécimal ou en binaire), puis qui demande en quelle base l'utilisateur désire avoir la conversion, et qui affiche la conversion.

7 test des fonctions

Vous testerez chaque fonction individuellement pour vous assurer de leur bon fonctionnement.

Sujet de TP 4

Le Jeu de la Vie

Ce TP se déroulera sur 2 séances !

Vous allez apprendre lors de cette séance de T.P. l'art de vivre et de survivre en Python ! Le Jeu de la Vie a été inventé par J. Conway et présenté dans le Scientific American en Octobre 1970. L'univers de jeu est représenté par un damier. Une case du damier ne peut être occupée que par une et une seule cellule. Le jeu simule la vie des cellules du damier (naissance, survie et mort). La simulation repose sur 4 règles extrêmement simples entre deux générations :

— Une cellule continue à vivre si elle a 2 ou 3 cellules voisines.
— Une cellule meurt d'isolement si elle a moins de 2 voisines.
— Une cellule meurt d'étouffement si elle a plus de 3 voisines.
— Une cellule naît dans une case vide, si 3 cases voisines exactement sont occupées.

Toutes les cellules de la grille évoluent simultanément à chaque saut de génération. Votre travail consiste à traduire ces règles en langage Python et de suivre la vie des cellules.

FIGURE 4.1 – Exemple d'évolution

A Travail demandé

Vous représenterez la grille par une liste de listes d'entiers (une liste de lignes...). Si la case vaut 1, cela signifie qu'il y a une cellule, et 0 sinon.

- Écrire une fonction permettant de créer une grille vide. Pour cela, la fonction admet deux entiers en paramètre (nombre de lignes : l, et nombre de colonnes : c) et qui renvoie une liste de l listes contenant c 0.
- Écrire une fonction pour initialiser la grille (qui définit les cellules en vie à la première génération). On pourra demander par exemple à l'utilisateur combien de cellule il souhaite mettre dans la grille et leur coordonnées. La grille sera passée en paramètre.
- Écrire une fonction permettant l'affichage de la grille, passée en paramètre, (simplement les 0 et 1) par exemple :
  0100
  1111
  0000
  0110

- Écrire une fonction retournant le nombre de voisin(s) d'une cellule (attention en bordures de la grille !). La fonction admettra en paramètre : la grille et les coordonnées de la cellule.
- Écrire une fonction qui crée la génération suivante. La grille sera passée en paramètre
- Écrire une fonction permettant l'affichage de la grille en mode texte comme ci-dessous (la grille est passée en paramètre) :

```
---------
| |*| | |
---------
|*|*|*|*|
---------
| | | | |
---------
| |*|*| |
---------
```

Le programme principal demandera à l'opérateur de saisir les dimensions de la grille, le nombre de cycles de simulation et affichera pour chaque cycle la grille.

Important : Vous n'utiliserez aucune variable globale. Pensez à utiliser 2 listes : 1 pour la génération actuelle et un autre pour placer les cellules de

la génération suivante (sinon vous écraserez des données essentielles pour le calcul du nombre de voisins).

1 Quelques prolongements pour la seconde séance

— Placement aléatoire d'un nombre aléatoire de cellules. Il existe en Python, la fonction $randint(a, b)$ qui retourne un nombre entier aléatoire compris entre a et b compris. Pour l'utiliser il faut importer la bibliotèque *random*

— Dessiner avec la tortue. A partir du code suivant et de la documentation (https ://docs.python.org/fr/3.7/library/turtle.html) proposer une version graphique de votre Jeu de la vie :

```
import turtle

def creerFenetre()->turtle.Turtle:
    return turtle.Turtle()

def dessinerCarre(t:turtle.Turtle,couleur:str,x:int,y:int,taille:int):
    t.penup()
    t.goto(x,y)
    t.pendown()
    t.pencolor(couleur)
    t.forward(taille)
    t.left(90)
    t.forward(taille)
    t.left(90)
    t.forward(taille)
    t.left(90)
    t.forward(taille)
    t.left(90)
    t.penup()

def dessinerSegment(t:turtle.Turtle, couleur:str,x1:int,y1:int,x2:int,y2:int):
    t.penup()
    t.goto(x1,y1)
    t.pendown()
    t.pencolor(couleur)
    t.goto(x2,y2)
```

```
    t.penup()


def effacerTout(t:turtle.Turtle):
    t.clear()


tortue=creerFenetre()
tortue.speed(2)
dessinerCarre(tortue,"red",0,100,50)
dessinerSegment(tortue,"blue",50,25,200,300)
```

Sujet de TP 5

Ce TP se déroulera sur 1 séance et demie (4,5h) :

On veut saisir les renseignements d'une liste d'étudiants d'une formation. Ces renseignements sont :

- Le nom (une chaîne de caractères)
- Le prénom (une chaîne de caractères)
- L'année de naissance (entier)
- La moyenne générale (réel)
- La promotion (entier : 1 ou 2)

1 Partie 1

- Définir en Python le type (class) ETUDIANT
- Écrire une fonction permettant de saisir les informations d'un ETUDIANT. La fonction retournera l'ETUDIANT ainsi créé.
- Écrire une fonction permettant l'affichage d'une liste d'ETUDIANT.
- Écrire une fonction pour l'ajout d'un nouvel étudiant à la fin de la liste d'ETUDIANT passée en paramètre
- Écrire une fonction qui recherche un étudiant selon son nom (la fonction retournera l'indice de l'ETUDIANT dans la liste)
- Écrire une fonction pour le retrait d'un étudiant d'une liste d'ETUDIANT passée en paramètre
- Écrire une fonction pour trier la liste d'ETUDIANT passée en paramètre selon la moyenne en utilisant un algorithme de tri de votre choix
- Écrire une fonction permettant de retrouver le major de chaque promotion.

2 Partie 2

— Écrire une fonction permettant de sauvegarder une liste d'ETUDIANT dans un fichier.
— Écrire une fonction permettant de récupérer dans une liste d'ETUDIANT sauvegardée dans un fichier.
— Écrire une fonction permettant de modifier les renseignements relatifs à un étudiant figurant dans le fichier.
— Écrire une fonction permettant de supprimer un étudiant du fichier.
— Écrire une fonction pour le retrait d'un étudiant d'une liste d'ETUDIANT passée en paramètre
— Écrire une fonction permettant de visualiser les ETUDIANT sauvegardés dans fichier.

Pour l'ensemble de ces fonctions, faire une version en fichier "texte" et en fichier "binaire"

Sujet de TP 6

Traitement d'images et fichiers
Ce TP se déroulera sur 1 séance et demie (4,5h)
Il s'agit dans ce TP d'une introduction à la manipulation de fichiers d'images simples : le format BPM Une image numérique est une suite d'informations stockées dans un fichier. Ce qui indique que le fichier contient une image (et pas autre chose) ce sont les extensions, comme par exemple : jpg, png, gif, bmp etc... On parle de format jpg, format gif... Un format simple pour s'initier au codage d'une image est le format Portable PixMap. Celui-ci se décline en 3 modes selon le degré de coloration souhaité : noir et blanc, dégradés de gris, couleurs. Les images utilisées ou construites dans cette activité seront visionnées avec un logiciel comme GIMP2.

1 Le format PBM pour *Portable Bit Map*

Un fichier au format PBM comporte des informations diverses pour aider le logiciel à afficher l'image. Voici ces informations :

- P1
- Un caractère d'espacement (espace, nouvelle ligne)
- Largeur de l'image
- Un caractère d'espacement
- Hauteur de l'image
- Un caractère d'espacement
- Données de l'image :
  - L'image est codée ligne par ligne en partant du haut
  - Chaque ligne est codée de gauche à droite
  - Un pixel noir est codé par un 1, un pixel blanc est codé par un 0 (d'où le nom bit map)
  - Les caractères d'espacement (espaces, tabulations, retours à la

ligne ...) à l'intérieur de cette section sont ignorés.

Exemple de fichier :

```
P1
# Exemple de fichier PBM :
# ces lignes sont des
# commentaires
20 12
00000000000000000000
00001111111111100000
00001100000001100000
00001100000001100000
00001100000001100000
00001100000001100000
00001111111111000000
00001100000001100000
00001100000001100000
00001100000001100000
00001100000001100000
00001111111111100000
```

Travail à faire :

— Copier le contenu de ce cadre dans un éditeur de texte et l'enregistrer au format .pbm

— Ouvrir le fichier avec GIMP et le zoomer à 1600

— Ecrire un programme python qui ouvre un fichier PBM et crée un second fichier (même nom fichier mais suivi de Reverse, par exemple im1.pbm -> im1Reverse.pbm). Le second fichier sera le « négatif » du premier (les pixels blancs deviendront noirs et réciproquement).

2 Le format pgm :des dégradés de gris

Difficile de rendre une image attrayante avec seulement le noir et le blanc. Le format pgm (Portable Grey Map) permet de rendre des dégradés de gris sur une échelle allant de 0 (noir) à 255 (blanc). La nomenclature du fichier est sensiblement la même que pour le pbm, on remplace les 0 et les 1 par

les nombres décimaux correspondant « au code de gris ». Plus ce nombre est élevé, plus le gris est proche du blanc.

- P2
- Un caractère d'espacement (espace, nouvelle ligne)
- Largeur de l'image
- Un caractère d'espacement
- Hauteur de l'image
- Un caractère d'espacement
- La valeur maximale utilisée pour coder les niveaux de gris, cette valeur doit être inférieure à 65536
- Un caractère d'espacement
- Données de l'image :
  - L'image est codée ligne par ligne en partant du haut
  - Chaque ligne est codée de gauche à droite
  - Chaque pixel est codé par un nombre, précédé et suivi par un caractère d'espacement. Un pixel noir est codé par la valeur 0, un pixel blanc est codé par la valeur maximale et chaque niveau de gris est codé par une valeur entre ces deux extrêmes, proportionnellement à son intensité.

Exemple :

```
P2
# Exemple de fichier PBM :
#taille 10 par 10
10 10
#50 = nuance max=blanc
50
15 50 50 50 50 50 50 50 50 50
15 20 20 20 20 20 20 20 20 15
15 15 15 20 20 20 20 20 15 15
15 15 15 0  0  0  0   0  15 15
15 15 15 0  0  0  0  0  15 15
15 50 50 0  0  0  0  0  50 50
15 50 50 0  0  0  0  0  50 50
15 50 50 20 20 20 20 20 50 50
15 50 50 20 20 20 20 20 50 50
15 50 50 50 50 50 50 50 50 50
```

Travail à faire :

— Copier le contenu de ce cadre dans un éditeur de texte et l'enregistrer au format .pgm.
— Ouvrir le fichier avec GIMP et le zoomer à 1600
— Écrire un programme python qui ouvre un fichier PGM et crée un second fichier (même nom fichier mais suivi de Reverse, par exemple im1.pgm -> im1Reverse.pgm). Le second fichier sera le « négatif » du premier (les pixels clairs deviendront foncés et réciproquement).

L'image en Rouge Vert Bleu (Portable PixMap) Le format correspondant aux deux formats simples précédents, adapté à la couleur est le format ppm (Portable PixMap).

Le début du fichier est très ressemblant aux deux précédents :

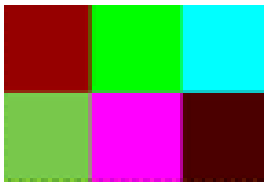
```
P3
# Exemple de fichier PBM :
10 10
255
```

On retrouve les mêmes paramètres, type (P3), ligne de commentaire, dimensions puis la valeur maximale pour l'intensité des couleurs. Celle-ci est couramment 255.

On code ensuite chaque ligne pixel par pixel avec les trois composantes RVB en partant de gauche à droite. On peut donc trouver un début comme celui-ci :

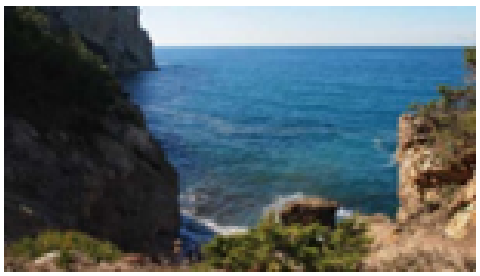
```
P3
# exemple 3 colonnes and 2 lignes,
3 2
255
150   0     0        0  255    0       0  255  255
120 200   075   255 0   255   075    0     0
```

Nous remarquons que pour coder une minuscule image de 6 pixels, nous avons utilisé 18 codes couleurs. Voici le rendu (très « zoomé ») de cet exemple. . ..



Travail à faire :

- Écrire un programme python qui demande à l'utilisateur la largeur et la hauteur (en pixels) désirées et qui crée un fichier francais.ppm et qui codera une image portable pixmap du drapeau français (bleu blanc rouge) selon les dimensions souhaitées par l'utilisateur.
- Ouvrir avec gimp l'image cassis.JPG et l'exporter au format ppm (si l'option est demandée répondre ASCII)
- Ouvrir le fichier avec un éditeur de texte
- Pour convertir un pixel R, V, B en niveau de gris G, G, G il suffit de calculer G = (R+V+B)/3. On notera que G doit être entier. Ecrire un programme Python qui convertit le fichier cassis.ppm en cassisNG.ppm correspondant à l'image en niveau de gris.
- Écrire un programme python qui convertit une image PPM (couleur) en image PGM (niveaux de gris)



Séance 2

1 Principe d'un filtre *simple*

On s'intéresse d'abord aux filtres qui fonctionnent de la façon suivante : Pour chaque pixel d'une image : On lit les composantes RVB On effectue une opération sur les composantes RVB et on obtient ainsi de nouvelles composantes. On remplace les anciennes composantes par les nouvelles.

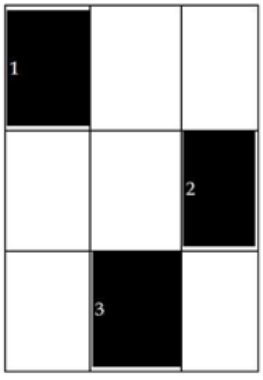
2. Filtre niveaux de gris

Pour passer d'une image couleur à une image en niveaux de gris, il faut remplacer chaque composante par une même valeur L, appelée luminance

(à ne pas confondre avec la luminosité du mode TSL). La luminance est le résultat d'un calcul de moyenne sur les 3 composantes d'un pixel arrondie à l'entier le plus proche. L est donc un entier entre 0 (noir) et 255 (blanc). On constate qu'avec la moyenne non pondérée, il est impossible de distinguer les trois dernières couleurs en niveaux de gris. Pour éviter cela, on utilise une moyenne pondérée. Les pondérations dépendent de la sensibilité des couleurs primaires à l'œil humain et du support utilisé pour afficher l'image. La formule couramment utilisée est : L = 0,21R + 0,71V + 0,08B. Modifier le code de conversion d'une image PPM en PGM en tenant compte de cette formule.

3. Filtre noir et blanc
Le principe du filtre noir et blanc reprend en partie celui du niveau de gris en fixant un seuil. On fixe un seuil entre 0 et 255. On calcule la luminance L du pixel. Si L est supérieure au seuil, alors on remplace les trois composantes RVB par 255. Sinon, on remplace les 3 composantes par 0 Écrire un programme python qui convertit une image PPM (couleur) en image PBM (Noir ou blanc).

4. Détection du contour
L'objectif est de ne garder que les contours d'une image bicolore en noir et blanc. Il n'y a pas qu'une seule technique pour réaliser un tel filtre. Certaines sont complexes. En voici une assez simple qui donne des résultats très corrects : Pour chaque pixel, nous allons compter combien il y a de pixel(s) noir(s) autour. Dans l'exemple ci-dessous, le pixel central est entouré de 3 pixels noirs et de 5 blancs.



On ne peut pas modifier directement l'image de départ sans altérer le comptage. On a donc besoin d'ouvrir au départ deux images identiques. La première sera lue pour le comptage mentionné ci-dessus (l'originale). La seconde image servira de support pour "écrire" (l'image finale) en utilisant la règle

suivante :
Si le nombre de pixel(s) est compris entre 3 et 6 inclus, alors on considère que l'on est près d'un contour et on écrit un pixel noir dans l'image finale. Sinon, on considère que l'on est dans une zone unicolore blanche ou noire et on écrit un pixel blanc dans l'image finale.

Écrire un programme python qui convertit une image PPM (couleur) en image PBM (Noir ou blanc) ne contenant que le contour des objets. On pourra prendre comme image le chat.