TP1
(Re)prise en main de python

1. Préambule

Pour ce premier TP, plusieurs problèmes sont proposés. Ne passez pas à l'exercice suivant avant d'avoir complètement terminé chacun des exercices et de l'avoir montré à un assistant qui pourra vous indiquer des améliorations possibles. Selon votre niveau, vous aurez le temps de faire un certain nombre d'exercices. Vous travaillerez seuls, sous l'environnement Linux, avec votre éditeur de texte préféré, comme expliqué en cours. Effectuer un **découpage en fonctions** de tous vos programmes, et respectez les consignes de bonnes pratiques vues en cours. Sauvegarder vos fichiers en utilisant le gestionnaire de version GIT.

2. Calendrier

2.1. Présentation

Le calendrier utilisé en France depuis 1582 est le calendrier dit *grégorien* :

- l'année est divisée en 12 mois numérotés de 1 à 12.
- les mois numéros 1, 3, 5, 7, 8, 10 et 12 ont 31 jours.
- Les mois numéros 4, 6, 9 et 11 ont 30 jours.
- le mois 2 a 29 jours si l'année est bissextile, 28 jours sinon.
- une année est bissextile si
  - elle ne se termine pas par 00 et est un multiple de 4
  - ou bien si elle est un multiple de 400.

2.2. Consignes

1) Écrire une fonction qui teste si une année est bissextile ou non avec la convention suivante : la fonction renvoie la valeur *True* si l'année est bissextile, la valeur *False* dans le cas contraire.
2) Écrire une fonction qui donne le nombre de jours d'un mois après avoir vérifié la validité des paramètres.
3) Écrire une fonction qui vérifiera si une date est valide ou non (*True* signifie date valide, *False* sinon)
4) Écrire le programme principal qui propose la saisie d'une date, la valide et affiche le message "date valide" ou "date non valide" selon le cas.

3. Montant de l'impôt d'une personne célibataire :

3.1. Objectif

En France, en 2022, une personne seule est imposable sur le revenu selon le tableau suivant :

Tranche	Montant :	Taux d'imposition
1	Jusqu'à 10 225 €	0 %
2	De 10 226 € à 26 070 €	11 %
3	De 26 071 € à 74 545 €	30%
4	De 74 546 € à 160 336 €	41 %
5	Plus de 160 336 €	45 %

Cela signifie que si elle gagne moins de 10 225 € par an, elle ne payera pas d'impôt sur le revenu. En revanche, si elle perçoit entre 10 226 € et 26 070 € par an, elle sera imposée à 11% sur le montant dépassant les 10 225€, et ainsi de suite.
Par exemple, une personne gagnant 50 000 euros par an payera :

(50 000 - 26 071) * 30/100 = 7 178,7 €
(26 070 – 10 225) * 11/100 = 1 743,1euros

Soit un total de 8 921 €. Le service des impôts, dans sa grande générosité, tronque à l'euro près !

3.2. Consignes

Implémenter une fonction `def mesImpots(revenu):` qui calcule le montant de l'impôt sur le revenu d'une personne seule. Le programme principal demandera à l'utilisateur le montant de ses revenus et appellera ladite fonction avant de faire un affichage.

4. Multiplication de matrices

4.1. Objectif

Les matrices sont des tableaux à deux dimensions sur lesquels on peut faire différentes opérations. En cas de compatibilité des dimensions, l'addition de deux matrices ne pose pas de problèmes particuliers, il suffit d'ajouter chacun des éléments correspondants.
Pour la multiplication, le processus est un peu plus compliqué : chaque élément de la matrice résultat est la somme des produits des éléments d'une ligne de la première matrice par les éléments d'une colonne de la seconde matrice. Par exemple, si on note $a_{i,j}$ le terme général de la matrice A se trouvant à la ligne i et à la colonne j, alors pour chaque élément de $A = B \times C$, on aura :

$$a_{i,i} = \sum_{k=0}^{n} b_{i,k} \times c_{k,j}$$

Pour cet exercice, il vous est demandé, dans un premier temps, d'écrire une fonction affichant le résultat de la multiplication de deux matrices carrées de 3 lignes et 3 colonnes. Ensuite, il faudra rendre cette fonction générique afin qu'elle permette de calculer le produit de matrices de tailles différentes si c'est possible.

4.2. Consignes

1) Vous devrez écrire une fonction `def multiplication():` qui prend deux paramètres (un pour chaque matrice à multiplier) et qui retourne le résultat. On travaillera avec des listes de listes.
2) La définition des matrices *A* et *B* ainsi que la déclaration de la matrice *C* se feront en dur dans le programme principal.
3) L'affichage du résultat se fera également dans le programme principal, dans un premier temps, puis à travers une fonction d'affichage dans un second temps.

5. Récursivité, Tours de Hanoï

5.1. Présentation

Le jeu des tours de Hanoï consiste à déplacer un certain nombre de palets d'un plot à un autre en un minimum de mouvements. Il existe trois plots. La difficulté du jeu tient à ce qu'il est interdit de placer un palet au-dessus d'un palet plus petit. Et à chaque mouvement, on ne doit déplacer qu'un seul palet.

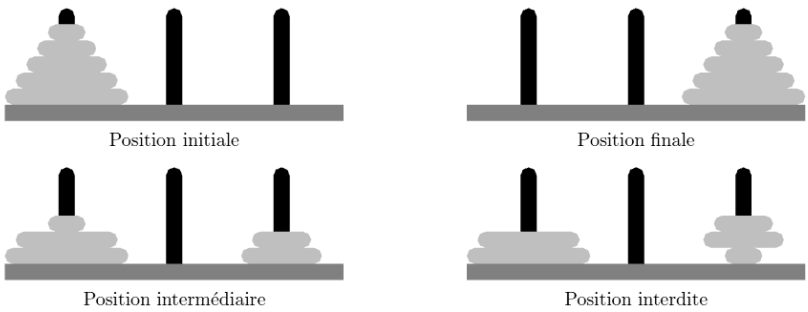

Figure 1 – Règles du jeu.

5.2. Objectif

Pour cet exercice, vous devrez réaliser un programme qui résout ce jeu de manière optimale. Bien entendu, il est hors de question de faire une représentation graphique du jeu. Le programme devra simplement afficher quelque chose comme ceci :

Nombre de disques : 4
Déplacer le disque du plot 1 vers le plot 2
Déplacer le disque du plot 1 vers le plot 3
. . .
Déplacer le disque du plot 1 vers le plot 3
Déplacer le disque du plot 2 vers le plot 3

5.3. Consignes

La fonction permettant de résoudre tout cela est en fait une fonction récursive assez courte qui repose sur l'idée suivante : pour arriver à déplacer une tour de n palets du plot 1 au plot 2, il suffit de

- déplacer la tour formée par les $n-1$ plus petits palets sur le plot 3,
- déplacer le plus grand palet sur le plot 2,
- puis déplacer la tour de $n - 1$ palets du plot 3 au plot 2.

Et finalement, l'affichage consiste à déplacer le palet du bas.
1) Implémenter la fonction Hanoi(…) qui effectuera le traitement demandé.
2) Modifier votre programme de manière à compter le nombre de déplacements nécessaire.

6. Conjecture de Syracuse

6.1. Présentation

En mathématiques, on appelle suite de Syracuse une suite d'entiers naturels définie de la manière suivante : * On part d'un nombre N entier plus grand que zéro
* s'il est pair, on le divise par 2
* s'il est impair, on le multiplie par 3 et on ajoute 1

Quel que soit le nombre de départ N choisi, on admettra qu'au bout d'un certain nombre d'itérations, on obtient toujours le cycle : 1, 4, 2, 1, 4, 2, 1, 4, 2… On appelle temps de vol, le nombre d'itérations nécessaires pour obtenir le premier 1.
On appelle également altitude maximum le nombre maximal atteint lors du parcours de cette suite.

6.2. Consignes

1) Écrire une fonction qui affiche la suite de Syracuse pour un entier donné en entrée.
2) Écrire une fonction qui retourne l'altitude maximale atteinte lors du déroulement de la suite.
3) Écrire une fonction qui calcule le temps de vol d'un nombre N.
4) Quel nombre entre 1 et 1000 possède le temps de vol le plus long ?
5) Quel nombre entre 1 et 1000 atteint l'altitude la plus haute ?

7. Nombres tricolores

7.1. Présentation

Un nombre est dit tricolore si son carré s'écrit uniquement avec des chiffres carrés parfaits non nuls, c'est-à-dire avec les chiffres 1, 4 et 9.
Par exemple, 7 est un nombre tricolore car $7^2=49$ et 49 s'écrit avec les chiffres 4 et 9.

7.2. Consignes

1) Écrire une fonction booléenne *tricolore(n)* permettant de vérifier si n est tricolore.
2) Écrire une fonction *tous_les_tricolores(N)* renvoyant la liste de tous les entiers tricolores inférieurs ou égaux à N.

8. Nombres amicaux

8.1. Présentation

En mathématiques, on dit que deux nombres entiers positifs *n* et *m* sont **amicaux** si les deux conditions suivantes sont réunies :

- la somme des diviseurs de l'un est égale à la somme des diviseurs de l'autre
- la somme de ces deux sommes vaut la somme des deux nombres.

8.2. Notations

- Les nombre manipulés ici sont des nombres entiers strictement positifs
- La notation a%b désigne le reste de la division entière de a par b. Par exemple : 21%7 = 0 et 23%5=3.
- Le nombre ***a*** est diviseur de ***n*** si le reste de la division entière de ***n*** par ***a*** vaut 0.
- On note $\sigma(X)$ la somme des diviseurs du nombre X.

8.3. Exemple

220 et 284 sont amicaux car :

- L'ensemble des diviseurs de 220 est : {1,2,4,5,10,11,20,22,44,55,110,220}
- L'ensemble des diviseurs de 284 est : {1,2,4,71,142,284}
- $\sigma(220) = 1 + 2 + 4 + 5 + 10 + 11 + 20 + 22 + 44 + 55 + 110 + 220 = 504$
- $\sigma(284) = 1 + 2 + 4 + 71 + 142 + 284 = 504$
- 220 + 284 = 504

Il existe 13 paires de nombres amicaux dont le premier à moins de 6 chiffres. Quatre sont donnés ici en jeux de tests : 220 et 284, 1184 et 1210, 2620 et 2924, 66928 et 66992.

8.4. Consignes

Effectuer un découpage algorithmique permettant de vérifier si deux nombres sont amicaux ou non.
Programmer votre solution.