Feuille d'exercices

1 Partie 1 : Les bases de la programmation en Python

Exercice 1 On souhaite calculer les coordonnées du point d'intersection de deux droites données sous la forme

$$\begin{cases} y = ax + b \\ y = cx + d \end{cases}$$

On suppose ici que a, b, c et d sont des entiers. Écrire un programme qui demande ces quatre valeurs à l'utilisateur puis affiche les coordonnées du point d'intersection. Que se passe-t-il si les droites sont parallèles ?

Exercice 2 Écrire un programme qui demande un entier `n` à l'utilisateur, puis calcule et le résultat de la multiplication $2 \times 2 \times 2 \cdots \times 2$ (où on a `n` occurrences de `2`).

Exercice 3 Écrire un programme qui demande un entier `n` à l'utilisateur, puis calcule et affiche $1 + 2 + \cdots + \texttt{n}$. Vérifier que ce calcul vaut bien `n*(n+1)//2`.

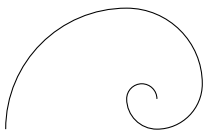
Exercice 4 Écrire un programme qui demande à l'utilisateur une somme initiale `s` déposée sur un livret, un taux d'intérêts annuel `t` exprimé en pourcents et un nombre d'années `n`, et qui affiche les intérêts perçus chaque année ainsi que le montant total présent sur le livret après `n` années.

Exercice 5 Écrire un programme Turtle demandant à l'utilisateur un entier `n` et traçant un escalier à `n` marches.

Exercice 6 Écrire un programme Turtle demandant à l'utilisateur deux entiers `n` et `m` et traçant une grille de $\texttt{n} \times \texttt{m}$ cases (on pourra choisi un facteur d'échelle `c` pour que le dessin soit plus joli)q.

Exercice 7 La suite de Fibonacci est la suite d'entiers (F_n) définie par $F_0 = 0$, $F_1 = 1$ et, pour tout entier n à partir de 2, $F_n = F_{n-1} + F_{n-2}$. Écrire un programme qui demande à l'utilisateur un entier `n`, qu'on supposera supérieur ou égal à 1, et qui affiche la valeur de F_n. Indication : on pourra utiliser deux variables pour mémoriser F_n et F_{n-1}, ainsi qu'une variable temporaire.

Exercice 8 Écrire un programme Turtle qui demande un entier `n` à l'utilisateur et trace une spirale de Fibonacci effectuant `n/4` tours. On obtient une spirale de Fibonacci en traçant, pour chaque valeur successive de la suite de Fibonacci, un quart d'arc de cercle de ce diamètre.



Exercice 9 Écrire un programme qui demande un entier `n` à l'utilisateur et affiche tous ses diviseurs.

Exercice 10 On considère un billard dont les dimensions sont données par les variables `longueur` et `largeur`, avec une seule bille de coordonnées (x, y) (en supposant que le coin inférieur gauche a les coordonnées $(0, 0)$ et le coin supérieur droit les coordonnées $(\texttt{longueur}, \texttt{largeur})$). Après application d'un vecteur de déplacement (d_x, d_y) les nouvelles coordonnées de la bille sont $(x + d_x, y + d_y)$, si ce déplacement n'implique pas de transpercer une paroi du billard. Sinon, la bille effectue un rebond parfait sur chacune des parois rencontrées : la nouvelle direction est symétrique de l'ancienne par rapport à la paroi sur laquelle la bille rebondit, et la distance restant à parcourir est inchangée.

Écrire un programme qui demande les coordonnées de la bille et un vecteur de déplacement (d_x, d_y), pour lequel on supposera $-\texttt{longueur} \leq d_x \leq \texttt{longueur}$ et $-\texttt{largeur} \leq d_y \leq \texttt{largeur}$, et qui affiche les coordonnées de la bille à la fin du mouvement.

Exercice 11 Écrire un programme pour résoudre une équation du second degré $ax^2 + bx + c = 0$.

Exercice 12 Écrire une fonction `occurrences(v, t)` qui renvoie le nombre d'occurrences de la valeur `v` dans le tableau `t`.

Exercice 13 Écrire un programme qui tire au hasard mille entiers entre 1 et 10 et affiche ensuite le nombre de fois que chaque nombre a été tiré. Relancer le programme plusieurs fois.

Exercice 14 Pour mélanger les éléments d'un tableau aléatoirement, il existe un algorithme très simple qui procède ainsi : on parcourt le tableau de la gauche vers la droite et, pour chaque élément à l'indice i, on l'échange avec un élément situé à un indice tiré aléatoirement entre 0 et i (inclus). Écrire une fonction `melange(tab)` qui réalise cet algorithme. On pourra se resservir de la fonction `echange` (Cet algorithme s'appelle le *mélange de Knuth.*)

Exercice 15 Écrire une fonction `hamming(tab1, tab2)` qui prend en paramètres deux tableaux, que l'on supposera de la même taille, et qui renvoie le nombre d'indices auxquels les deux tableaux diffèrent.

Exercice 16 Écrire une fonction qui prend un entier positif ou nul en argument et renvoie son nombre de chiffres.

Exercice 17 Écrire une fonction `sous_mot(a, b, tab)` qui teste s'il existe dans le tableau `tab` une occurrence de l'élément `a` et une occurrence de l'élément `b` dans cet ordre (mais pas nécessairement consécutives).

Exercice 18 Écrire une fonction `pgs(e, tab)` qui renvoie la longueur de la plus grande séquence d'occurrences consécutives de l'élément `e` dans le tableau `tab`.

Exercice 19 Écrire une fonction `pgpc(tab1, tab2)` qui prend deux tableaux et qui renvoie la longueur du plus grand préfixe commun à ces deux tableaux, c'est-à-dire un nombre `l` tel que les `l` premières valeurs de `tab1` et `tab2` sont égales mais pas celles d'indice `l`.

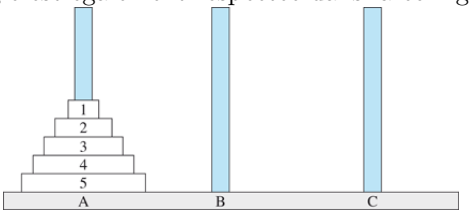
Exercice 20 Écrire un programme qui affiche les tables de multiplications sous la forme

```
Table de 1 :
  1 x 1 = 1
  1 x 2 = 2
  ...
Table de 2 :
  ...
```

Exercice 21 Écrire une fonction qui prend en paramètre un tableau à deux dimensions de hauteur et de largeur non nulles et qui renvoie le maximum parmi les minima de chaque ligne. Indication : on pourra utiliser une fonction auxiliaire pour calculer le minimum d'une ligne.

Exercice 22 Les tours de Hanoï sont un jeu de réflexion consistant à déplacer des disques de diamètres différents d'une tour de « départ » à une tour d'« arrivée » en passant par une tour « intermédiaire », et ceci en un minimum de coups, tout en respectant les règles suivantes :

— On ne peut déplacer plus d'un disque à la fois.
— On ne peut placer un disque que sur un autre disque plus grand que lui ou sur un emplacement vide.
— On suppose que cette dernière règle est également respectée dans la configuration de départ.



Écrire une fonction `hanoi(n)` qui résoud le problème des tours de Hanoï en affichant les mouvements à effectuer, comme ci-dessous (où on utilise les nombres 1, 2 et 3 pour désigner les tours). L'argument `n` indiquera le nombre de disques à déplacer entre la tour de départ et la tour d'arrivée.

```
>>> hanoi(3)
 Move  1  on 3
 Move  1  on 2
 Move  3  on 2
 Move  1  on 3
 Move  2  on 1
 Move  2  on 3
 Move  1  on 3
```

Exercice 23 La courbe de Koch est une figure qui s'obtient de manière récursive. Le cas de base à l'ordre 0 de la récurrence est simplement le dessin d'un segment d'une certaine longueur l, comme ci-dessous (figure de gauche).

Le cas récursif d'ordre n s'obtient en divisant ce segment en trois morceaux de même longueur $l/3$, puis en dessinant un triangle équilatéral dont la base est le morceau du milieu, en prenant soin de ne pas dessiner cette base. Cela forme une sorte de chapeau comme dessiné sur la figure de droite ci-dessus. On réitère ce processus à l'ordre $n-1$ pour chaque segment de ce chapeau (qui sont tous de longueur $l/3$). Par exemple, les courbes obtenues à l'ordre 2 et 3 sont données ci-dessous (à gauche et à droite, respectivement).

Écrire une fonction `koch(n, l)` qui dessine avec `Turtle` un flocon de Koch de profondeur `n` à partir d'un segment de longueur `l`.