Travaux pratiques

Pour tous les exercices vous pouvez vous aider des slides et de la cheat sheet. N'hésitez pas à nous appeler pour demander de l'aide et vous pouvez bien évidemment vous entraider!

Exercice 1:

Le but de ce premier exercice va être de mettre en pratique tout ce qui a été présenté avec des exercices simples.

1) Écrire un programme qui demande son nom à l'utilisateur et lui répond "Bonjour " suivie de son nom
2) Sur ce même programme, stocker maintenant le nom dans une variable, demander l'âge de l'utilisateur et stocké le également dans une variable. Afficher ensuite "Bonjour <nom>! Vous avez <âge>."
3) Ajoutez le test suivant :
    - si l'utilisateur a plus de 18 ans affichez "Vous êtes majeur!"
    - sinon affichez "Vous êtes mineur!".
4) Maintenant on va permettre à l'utilisateur d'entrer plusieurs nom et plusieurs âges
    - On commence par demander à l'utilisateur un nom et un âge
    - Ensuite on lui demande s'il veut encore entrer un nom et un âge
    - S'il répond oui, on lui redemande un nom et un âge ainsi de suite
    - S'il répond non, on arrête la boucle et on affiche la liste avec tous les noms et tous les âges.
    - *Il faut donc utiliser des listes et une boucle, ainsi qu'une condition pour vérifier la réponse de l'utilisateur, pour savoir s'il veut continuer ou non et adapter le comportement de la boucle en conséquence.*

Exercice 2:

On va maintenant faire un petit jeu de calcul mental, plus évolué que celui montré dans le cours. Voici quelques indications

1) Commencez par créer trois variables, deux pour les valeurs du calcul, la troisième pour le choix de l'opérateur (parmis +,-,* et /), les trois choisies aléatoirement
2) Utiliser plusieurs if pour choisir l'opération à effectuer, dans ces if, affichez le calcul à effectuer à l'utilisateur, récupérez sa réponse, et effectuez également le calcul de la bonne réponse.
3) Dans le cas de la division, attention à ne pas faire une division par 0!
4) Après ces if, comparez la bonne réponse et la réponse de l'utilisateur (attention à bien convertir le résultat de l'input en entier, avec int()!). Si la réponse est bonne, félicitez l'utilisateur, si elle est fausse, donnez lui la bonne réponse.
5) Une fois tout ça fait, vous pouvez inclure tout ça dans une grand boucle while, et à la fin de chaque partie, demander à l'utilisateur s'il veut rejouer.

Exercice 3:

Cet exercice consistera à créer un jeu où il faut deviner un mot mystère, choisi aléatoirement dans un dictionnaire. Voici les étapes du jeu:

- le programme choisi un mot dans un fichier
- Il affiche le mot avec seulement la première lettre, puis des étoiles à la place des autres lettres.
- L'utilisateur peut alors entrer un essaie.
- Le jeu compart le mot entré avec le bon mot. Pour chaque lettre bien placée, on enlève l'étoile dans le mot mystère et on la remplace par la bonne lettre.
- On continue jusqu'à ce que le mot soit découvert ou qu'on arrive à court d'essais (vous choisirez le nombre d'essai que le joueur peut effectuer).

La partie qui permet d'ouvrir le fichier est disponible dans le fichier "exo3.py", voici le détail de ce qu'elle fait:

- file contient l'emplacement du fichier, comme "exo3.py" et la liste de mots français est dans le même répertoire, seul le nom suffit.
- ensuite j'utilise une structure un peu particulière qu'on appelle "compréhension de liste". Cela va remplir la liste avec les mots du fichier (ouvert avec open(file)) et en enlevant le saut de ligne ('\n') à la fin des mots. Autrement dit pour chaque ligne dans le fichier, on va le mettre dans la liste en retirant le saut de ligne à la fin.

Voici quelques indications:

- la liste 'liste' contient l'ensemble des mots disponibles dans le fichier
- Commencez par en choisir un au hasard parmis tous
- copiez le et remplacez toute les lettres par '*' sauf la première.
- maintenant, tant que l'utilisateur a encore des essais et n'a pas trouvé la bonne réponse, demandez lui d'essayer un mot
- si les deux mots sont identiques, l'utilisateur a gagné! Vous pouvez terminer le jeu ou relancer une partie
- Si ce n'est pas la bonne réponse, comparez chaque lettre de sa réponse avec le bon mot, si les lettres correspondent, enlever l'étoile dans la copie du mot et mettez la bonne lettre à la place. Retourner ensuite au début de la boucle pour que le joueur réessaie!

Exercice 4:
Un peu de dessin maintenant! Nous allons utiliser le module turtle pour cela!

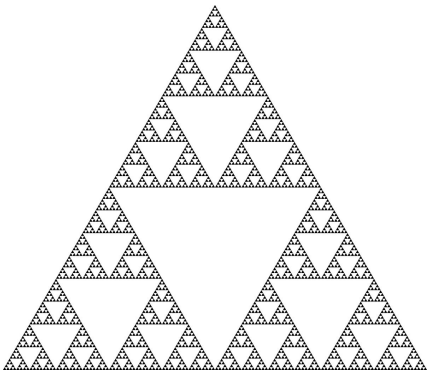
Turtle permet de diriger un curseur, qui va pouvoir se déplacer, tourner, et colorer des pixels.
Vous aurez ici la liste de toute les fonctions disponibles:
https://docs.python.org/3/library/turtle.html#turtle.pensize
Mais voici quelques bases qui nous suffiront:

- Avancer de x pixels : turtle.forward(x) ou turtle.fd(10)
- Reculer de x pixels : turtle.backward(x) ou turtle.bk(10)

- Tourner de x degrés : turtle.right(x) ou turtle.left(x)
- lever et baisser le crayon : turtle.up() et turtle.down()
- Aller à une position x,y : turtle.goto(x,y)
- Dessiner un cercle de rayons r : turtle.circle(r)
- Dessiner un point d'épaisseur x (doit être supérieur à 1) : turtle.dot(x)

Commencer par faire quelques fonction, comme pour dessiner un carré, un rectangle, un triangle…

Vous pensez avoir pris turtle en main? Alors passons à quelque chose de plus intéressant, une fractale! Le triangle de Sierpienski, voici à quoi il ressemble :



Cette fractale est composées de triangles eux même composés de trois triangles à l'infini. Une façon simple de la

Tout d'abord, il nous faut une petite fonction qui calcule le milieu de deux points, à partir de x1,y1 et x2,y2! Si vous ne savez plus comment faire, il suffit de faire la somme des x sur 2 et la somme des y sur 2.

Si ce n'est pas déjà fait, faites une fonction triangle(a,b,c) qui dessine un triangle de coordonnées a,b et c.

Maintenant, faisons une fonction Sierpienski(a,b,c,n). a,b et c sont les coordonnées du plus grand triangle et n un nombre qu'on choisi au début (10 par exemple) et qui va décroitre à chaque appel de la fonction.
Dans cette fonction on commence par vérifier que n>0, si ce n'est pas le cas on sort. Si c'est le cas alors on dessine le triangle a,b,c, et on décrémente n de 1.
Ensuite on va rappeler Sierpienski (toujours dans la fonction) pour les 3 triangles inclus dans le grand. Pour cela on l'appel pour le triangle a, milieu[a,b] et milieu[a,c] puis même chose pour les triangles liés aux points b et c, avec les milieux des côtés auxquels ils appartiennent respectivement.
Et c'est tout! C'est ce qu'on appelle une fonction récursive, elle va s'appeler elle même, avec des arguments différents, et s'arrêter lorsque n sera tombé à 0.

Si vous arrivez à comprendre cet algorithme, bravo! Si vous avez tout terminé et que vous ne savez pas quoi programmer, demandé nous, nous aurons des idées pour aller plus loin!