Exercices

Mini Projet 1

Créer un module ***multiplication*** qui contiendra les fonctions suivantes:

- *addition(nbre1, nbre2):* qui retourne le résultat de *nbre1 + nbre2*
- *soustraction(nbre1, nbre2):* qui retourne le résultat de *nbre1 - nbre2*
- *multiplication(nbre1, nbre2):* qui retourne le résultat de *nbre1 * nbre2*
- *division(nbre1, nbre2):* qui retourne le résultat de *nbre1 / nbre2*

Créer un deuxième module ***helper*** qui contiendra les fonctions suivantes:

- *afficheMenu():* qui affiche le menu des actions (addition, soustraction, multiplication, division)
- *saisirAction(minAction, maxAction):* qui demande le numéro de l'action à réaliser numéro compris entre *minAction* et *maxAction* par exemple dans notre cas çà sera entre 1 et 4
- *saisirNombre(typeNbre):* qui va demander à l'utilisateur de saisir un nombre *typeNbre* permet de dire si c'est le nombre 1 ou le nombre 2 qui doit être renseigné.

Créer un fichier main.py qui va utiliser l'ensemble des fonctions des modules précédents pour faire réaliser une calculatrice simple.

Cette fois-ci il faut gérer les cas d'erreurs possibles avec les blocs ***try/catch*** et si besoin ***finally/else/assert.***

Exercices

Mini Projet 2

Coder un jeu de nombre magique où l'utilisateur doit deviner un nombre aléatoire généré en 10 tentatives maximum.
Le jeu doit être constitué de trois modes, le mode facile (nombre magique entre 0 et 100), le mode moyen (nombre magique entre 0 et 1000) et le mode difficile (nombre magique entre 0 et 10000).
Après choix du mode faire deviner le nombre magique à l'utilisateur quand l'utilisateur entre un nombre inférieur au nombre magique afficher « Le nombre magique est plus grand que nombreSaisi ».
Quand l'utilisateur entre un nombre supérieur au nombre magique afficher « Le nombre magique est plus petit que nombreSaisi ».
Sinon afficher « Bingo vous avez trouvé le nombre magique en x essais ».
En cas d'échec afficher un message d'échec et demander si l'utilisateur veut recommencer si oui réafficher le menu.
Cette fois-ci il faut gérer les cas d'erreurs possibles avec les blocs ***try/catch*** et si besoin ***finally/else/assert.***

Exercices

Mini Projet 3

Le rôle de ce programme est d'importer la fonction ***drawRosace*** du module ***rosace*** ensuite votre rôle est de comprendre comment utiliser cette fonction. Ensuite demander à l'utilisateur les paramètres d'entrées et appeler la fonction avec la fonction avec les paramètres fournis par l'utilisateur.
Votre travail ici et de savoir utiliser un module externe et comment demander des valeurs correctes à l'utilisateur.
Il faut gérer les cas d'erreurs possibles lors de la saisie des paramètres avec les blocs ***try/catch*** et si besoin ***finally/else/assert.***

Mini Projet 4

L'objectif de ce mini projet est de créer une application de quiz sur les tables de multiplication.
Le but est de tirer au hasard un nombre qui sera mis dans une variable par exemple nbre1 et un autre nombre dans une variable nbre2 ensuite afficher la question suivante à l'écran: « Combien vaut le produit de nbre1 x nbre2 ? » en remplaçant nbre1 et nbre2 par leur valeur.
Si l'utilisateur(trice) saisit une réponse correcte afficher « Bravo ! » sinon afficher « Perdu ! La bonne réponse était x » x étant le résultat de la multiplication.

Exercices

Mini Projet 5

Refaire le mini projet 4 mais cette fois-ci l'on doit faire le choix entre trois modes:

- Mode facile (table de 0 à 10 composée de 10 questions uniquement les nombres positifs)
- Mode moyen (table de 0 à 20 composée 20 questions uniquement les nombres positifs)
- Mode difficile (table de 0 à 30 composée de 30 questions et multiplication de nombres positifs et négatifs)

L'utilisateur devra faire un choix du mode au départ ensuite on affiche les questions l'une à la suite de l'autre.

A la fin on devra afficher le score de l'utilisateur.