Exercices sur les boucles sélectives (while)

Exercice n° 1

Ecrire et tester une fonction `Xmax = StopLog(YMax)` qui retourne le premier entier `Xmax>1` tel que `ln(Xmax+1) > Ymax > ln(Xmax)`.

Pour le test, `Ymax` sera un réel strictement positif, saisi par l'utilisateur.

Suite à l'appel à `StopLog`, on affichera les valeurs de `ln(Xmax)` et de `ln(Xmax +1)` avec 3 décimales pour vérifier l'inégalité recherchée.

Exercice n°2

Ecrire une fonction qui attende `Ts` seconde. `Ts` sera l'argument d'entrée de la fonction. On utilisera la fonction `Seconde()` du module fourni `Donne_Heure.py`

La durée demandée sera obligatoirement inférieure à 60 seconde. Si ce n'est pas le cas il n'y aura pas d'attente.

On pensera pour écrire cette fonction que `Donne_Heure.Seconde()` est une fonction périodique modulo 60 !

Exercice n°3

Écrire et coder un programme qui demande une valeur d'angle en degré comprise entre 0° et 360°, jusqu'à ce que l'utilisateur saisisse un nombre correct. Si la réponse supérieure à 360, on affichera le message : « Trop grand ! ». De même si on est inférieur à 0 on affichera « Trop faible ! ».

Suite à cette saisie on affichera avec 2 décimales la valeur du cosinus et du sinus de cet angle.

Vous pourrez vérifier votre programme avec des angles connus comme π/2.

A partir de cet essai, écrire une fonction générique `Val = SaisieIn(Texto,Vmin,Vmax)` qui retournera la valeur réel `Val` qui sera saisie par l'utilisateur. Les arguments d'entrée de cette fonction sont :

`Texto` : chaine de caractères qui contiendra le message d'invite

`Vmin` et `Vmax` définissent les bornes de l'intervalle d'appartenance de la valeur à saisir

Ainsi pour l'exemple précédent on pourrait faire l'appel à cette fonction comme :

```
Invite = "Valeur d'un angle en degré dans l'intervalle [0°, 360°] :"
Inf = 0
Sup = 360
Val = SaisieIn(Invite,Inf,Sup)
```

Exercice n°4

Créer un module de fonction `Saisie` qui contienne la fonction `SaisieIn` précédente.

Ajouter à ce module la fonction `SaisieOut` qui réalise la même chose que `SaisieIn` sauf qu'elle vérifiera que la valeur saisie n'appartienne pas à l'intervalle `[min,max]`

Exercice n°5

Ecrire et tester une fonction `Y=DeciPi(x)` qui retourne la place de la première décimale de π qui vaut `x`.

Exemple : `DeciPi(6)` retournera 7 (π = 3. 141592653..)

Exercice n°6

Modifier la fonction **DeciPi** précédente pour créer la fonction **Y=DeciPim(x,M)** qui retourne la place de la décimale de la valeur de π qui vaut **x** pour le **M**ième fois

Exemple : **DeciPim(1,2)** retournera 3 (π = 3. 141592653..)

Exercice n°7 : Problème « Simulateur d'ascenseur »

On étudie le fonctionnement de l'ascenseur de la tour Histik qui assure 30 allers/retours par heure. A chaque aller, **N** personnes montent et à chaque retour **M** personnes descendent.

Afin de mettre en place un simulateur pour tester la gestion de la sécurité de la tour Histik, vous êtes en charge d'écrire ce simulateur qui détermine après chaque A/R de l'ascenseur combien de personnes sont encore en haut.

On vérifiera les contraintes suivantes :

- **N** et **M** sont a priori des nombres aléatoires différents l'un de l'autre et à chaque A/R compris. Ils sont entre 0 et 15 (capacité maximale de l'ascenseur).
- **M** ne sera jamais supérieur au nombre de personnes encore en haut.
- S'il y a plus que 50 personnes en haut, **N** sera obligatoirement nul (interdiction de monter)
- Le simulateur affichera après chaque heure (et non à chaque A/R !) combien de personnes sont en haut.
- L'ascenseur fonctionne sur 13h00 d'affilées. A partir de la 13ième heure, aucune personne ne pourra plus monter (**N**=0) et il y aura autant d'A/R nécessaires pour évacuer les personnes restantes en haut, toujours en utilisant une valeur aléatoire pour déterminer le nombre de personnes qui descendent à chaque passage.

Pour le codage on utilisera évidement des boucles « tant … que » ainsi qu'une ou plusieurs fonctions.

Exercice n°8 : Problème « Table de multiplication »

On souhaite écrire une petite application pour faire réviser les tables de multiplication. Cette application prendra la forme d'un jeu attribuant des points et mesurant le temps mis pour répondre.

Le « jeu » est le suivant :

- On demande à l'utilisateur quelle table il veut réviser. On attend donc un entier entre 2 et 9… sauf si vous êtes mesquin et que vous voulez aussi faire réviser la table par 37…
- L'ordinateur proposera ensuite 15 multiplications tirées au hasard auxquelles le joueur doit répondre, par exemple 5x8 = ?
- Le joueur aura 3 possibilités de répondre. S'il répond juste au premier essai, il marque 3 points, s'il répond juste au deuxième essai il marque 1 point, s'il répond juste au troisième essai il ne marque pas de point. S'il a faux aux trois essais, son score est amputé de 2 points.
- Au bout des 3 fausses réponses, on affichera le bon résultat.
- On mesurera le temps mis pour répondre à chaque question sans distinguer s'il répond en 1, 2 ou 3 essais, ni si la réponse finale est fausse (utilisation du module **Donne_Heure**)
- A la fin des 15 questions, on affichera le score réalisé, le temps cumulé mis pour répondre aux 15 questions, le temps moyen mis pour répondre aux 15 questions ainsi que le temps le plus long mis pour répondre à une question.

Pour le codage on utilisera évidement des boucles « Tant… que » ainsi qu'une ou plusieurs fonctions. On pourra évidement réutiliser des fonctions déjà écrites dans les exercices précédents.