Module Python

Série 1

Exercice 1 :

Écrire un script multiple3.py qui affiche en fonction d'une valeur saisie l'un des messages suivants :

– "Ce nombre est pair"

– "Ce nombre est impair, mais est multiple de 3"

– "Ce nombre n'est ni pair ni multiple de 3"

Afficher la date d'aujourd'hui en haut à droite (20 tabulations) de la page (indication : utiliser datetime comme module et la fonction date.today())

Exercice 2 :

Donner la table de multiplication jusqu'à 10 d'un entier donné par l'utilisateur (en utilisant nombre.isdigit() == True), vous devez aussi avoir l'avis de l'utilisateur s'il veut continuer avec votre programme ou quitter.

Exercice 3 :

Donner le nombre de fois qu'un nombre se divise par 2.

Exercice 4 :

Tester ce programme

```
def function(*nbre):
    resu=0
    for x in nbre:
        resu+=x
    print(resu)
```

Exercice 5 :

Donner le programme de la Factorielle version itérative et celle de la version récursive.

Exercice 6 :

```
def function(x):
    x=x+2
    return (x,x**3)
print(function(2))
```

Comment faire pour afficher par exemple x vaut : 2,23 ET x^3 vaut 11.089.

Exercice 7 :

Écrivez une fonction menu() qui permet de retourner le choix de l'utilisateur
parmi une liste de choix, et selon le choix il exécute une instruction.

1 : Nouvelle partie => On commence tte suite

2 : Recharger la partie => la partie se recharge .....

3 : Quitter => quitte le programme avec "pass"

Sinon réaffiche le menu.

Exercice 8 : (Mini projet)

Il s'agira d'un jeu dans lequel vous pourrez miser une certaine somme et gagner ou perdre de l'argent. Quand vous n'avez plus d'argent, vous avez perdu.

Les règles du jeu

Le joueur mise sur un numéro compris entre 0 et 999 (1000 numéros en tout). En choisissant son numéro, il y dépose la somme qu'il souhaite miser.

La roulette est constituée de 1000 cases allant de 0 à 999. Les numéros pairs sont de couleur noire,

les numéros impairs sont de couleur rouge. Un agent lance la roulette, lâche la bille et quand la roulette s'arrête, relève le numéro de la case dans laquelle la bille s'est arrêtée. Le numéro sur lequel s'est arrêtée la bille est, naturellement, le numéro gagnant.
Si le numéro gagnant est celui sur lequel le joueur a misé (probabilité de 1/1000, plutôt faible), l'agent lui remet 3 fois la somme misée.
Sinon, l'agent regarde si le numéro misé par le joueur est de la même couleur que le numéro gagnant (s'ils sont tous les deux pairs ou tous les deux impairs). Si c'est le cas, l'agent lui remet 50 % de la somme misée. Si ce n'est pas le cas, le joueur perd sa mise.
Dans les deux scénarios gagnants vus ci-dessus (le numéro misé et le numéro gagnant sont identiques ou ont la même couleur), l'agent remet au joueur la somme initialement misée avant d'y ajouter ses gains. Cela veut dire que, dans ces deux scénarios, le joueur récupère de l'argent. Il n'y a que dans le troisième cas qu'il perd la somme misée.
Indications :
Python a dédié tout un module à la génération d'éléments pseudo-aléatoires, le module random, et en particulier à la fonction randrange( ). Pour arrondir au nombre supérieur la somme d'argent et éviter plusieurs chiffres après la virgule flottante, vous pouvez utiliser la fonction ceil dans le module math.

Quand on demande à l'utilisateur de taper un nombre entre 0 et 999, il faut s'assurer qu'il l'a bien fait. On attend que le joueur saisisse un nombre. Si le nombre n'est pas valide, on demande à nouveau au joueur de saisir ce nombre. Utiliser le concept des exceptions afin de vérifier que l'utilisateur saisit bien un nombre. Si ce n'est pas le cas, on affiche un message d'erreur. La valeur de la variable qui contient le nombre est remise à -1 (c'est-à-dire une valeur qui indique à la boucle que nous n'avons toujours pas obtenu de l'utilisateur une valeur valide) et on utilise le mot-clé continue pour passer les autres instructions du bloc. De cette façon, si l'utilisateur fournit une donnée inconvertible, le jeu ne plante pas et lui redemande tout simplement de taper une valeur valide.

Utiliser une variable booléenne "continuer_partie" qui vaut « vrai » tant qu'on doit continuer la partie.