ITI 1520 - Devoir2

Date de remise : Dimanche 13 Octobre, 22h00

Ce devoir forme 5% de la note finale.

Vous devez faire ce travail individuellement. Il faut soumettre un répertoire compressé **Devoir2_<numEtudiant>.zip** (remplacez _<numEtudiant> par votre numéro d'étudiant) contenant les fichiers suivants:

- Nom du fichier à soumettre pour Question 1 : **D2Q1.py**
- Nom du fichier à soumettre pour Question 2 : **D2Q2.py**
- Nom du fichier à soumettre pour Question 3 : **D2Q3.py**
- Nom du fichier à soumettre pour Question 4 : **D2Q4.py**

Tous vos fichiers .py doivent avoir les lignes suivantes au début :

```
#Auteur : Votre nom et prénom
#Numéro d'étudiant : <Votre numéro d'étudiant>
```

Tous vos fichiers .py doivent aussi avoir les lignes suivantes au début de votre programme:

```
print("Auteur: votre nom et prenom")
print("Numéro d'étudiant: votre numéro d'étudiant ")
```

Aucun devoir en retard ne sera accepté. Un programme qui donne une erreur d'exécution (quand on clique sur Run) aura la note zéro.

Utilisez des commentaires dans votre code Python pour qu'il soit lisible.

Quand la question dit écrire une fonction, il **faut respecter le nom de la fonction exactement et les noms des paramètres quand ils sont donnés par l'énoncé**.

Le but de ce devoir est de pratiquer la conception et l'implémentation d'algorithmes et l'utilisation des expressions booléennes, des branchements et des boucles.

Barème (total de 20 points)

- Question 0: 0 points
- Question 1: 5 points
- Question 2: 5 points
- Question 3: 5 points
- Question 4: 5 points

ITI 1520 - Devoir2

Question 0 : réchauffement (optionnelle pour la remise)
Écrire une fonction *unites(nombre)* qui permet d'afficher sur 4 lignes le chiffre à chacune des positions de ce nombre. Cette fonction prend en entrée un nombre entre 0 et 9999.

Exemple 1:
>>> unites(13)
Nombre de milles : 0
Nombre de centaines : 0
Nombre de dizaines : 1
Nombre d'unites : 3

Exemple 2:
>>> unites(5628)
Nombre de milles : 5
Nombre de centaines : 6
Nombre de dizaines : 2
Nombre d'unites : 8

Question 1: Joueur de hockey

Un joueur de hockey des Sénateurs d'Ottawa se qualifie pour le programme de prime s'il compte plus de 20 buts ou récolte plus de 25 aides, ou alors obtient mois de 25 pénalités durant la saison régulière. Les joueurs qui se qualifient au programme selon ces règles sont séparés en trois groupes :

- Groupe 1. Les joueurs ayant été membre de l'équipe pour 5 ans ou plus et qui ont participé à plus de 55 matchs durant la saison régulière.
- Groupe 2. Les joueurs ayant été membre de l'équipe pour 5 ans ou plus et qui ont participé à 55 matchs ou moins durant la saison régulière.
- Groupe 3. Les joueurs ayant été membre de l'équipe pour mois de 5 ans, peu importe le nombre de matchs joues durant la saison régulière.

Selon ces groupes, les joueurs obtiennent une prime ceci :

- Joueurs du groupe 1 : obtiennent une prime complète.
- Joueurs du groupe 2 : obtiennent une prime partielle.
- Joueurs du groupe 3 : obtiennent une prime conditionnelle.
- Joueurs n'appartenant à aucun des 3 groupes : n'obtiennent aucune prime.

Dans la base de données financière, les primes ont les codes suivants :

Prime	Code de prime
Prime complète	3
Prime partielle	2
Prime conditionnelle	1
Aucune prime	0

ITI 1520 - Devoir2

a) Écrivez un programme Python (une fonction) qui prend comme données les statistiques d'un joueur (les buts, les aides, les pénalités, les parties jouées, les années de service), et qui donne comme résultat le code de prime qu'obtient le joueur. Exemple : si le joueur obtient une prime conditionnelle, le résultat est 1.

b) Écrivez le programme principal Python qui demande à l'utilisateur les statistiques d'un joueur et les lit au clavier. Les données entrées doivent être positives ou nulles donc il faut vérifier ceci, et il faut redemander à l'utilisateur de saisir les données jusqu'à ce que cette condition soit vérifiée.
L'algorithme principal fait appel à la fonction de la question (a). Ensuite l'algorithme principal affiche le code de prime du joueur.

Question 2 : suite Fibonacci

Écrivez une fonction appelée *fibonacci(n)* qui permet de calculer les *n* termes de la suite de Fibonacci. La suite de Fibonacci permet de bien illustrer les expressions mathématiques récursives, c'est-à-dire qu'un terme de la suite est défini à partir du/des termes précédents. Soit le premier terme de la série $n_0 = 0$ et le deuxième terme $n_1 = 1$, les termes suivants représentent la somme des deux termes précédents selon la relation suivante :

$$n_i = n_{i-1} + n_{i-2}, \text{ avec } n_0 = 0 \text{ et } n_1 = 1$$

La fonction *fibonacci(n)* devra prendre en paramètre un entier *n* supérieur à 2. Puis, il devra afficher les *n* valeurs de la suite de Fibonacci (n_0 et n_1 inclues).
Écrire un programme principal qui demande à l'utilisateur d'entrée le nombre de termes et vérifier que le nombre est supérieur à 2. Si l'utilisateur n'entre pas le bon numéro, redemandé de nouveau.
Utiliser le nombre entré par l'utilisateur pour tester la fonction *fibonacci().*

Exemple:
Entrez le nombre de termes : -3
Entrez le nombre de termes : 6
0 1 1 2 3 5

Entrez le nombre de termes: 8
0 1 1 2 3 5 8 13

Question 3 : triangle isocèle

Écrivez une fonction *triangle(hauteur)* qui permet d'afficher un triangle isocèle formé du caractère '*' (et d'espaces pour bien les aligner) d'une hauteur déterminée. Cette fonction prend en entrée la hauteur de triangle. NB: Vous devez utiliser des boucle for ou while.

ITI 1520 - Devoir2

Exemple:
>>> triangle(5)

```
    *
   ***
  *****
 *******
*********
```

Question 4 : palindrome

Écrire une fonction *palindrome(chaine)* qui prend une chaine de caractères en paramètre et retourne un booléen. Ce booléen retourné sera *TRUE* si la chaine de caractères est un palindrome et *FALSE* sinon. Un mot est dit palindrome si on peut le lire dans les deux sens. Nous allons nous contenter des mots pour le devoir. Si vous voulez vous pouvez considérer des phrases palindromes.

NB: Vous devez utiliser une boucle *for* ou *while.* Vous n'avez pas le droit d'utiliser des méthodes prédéfinies comme la méthode *reverse* qui permet d'inverser la chaine passée en paramètre.

Exemple:
>>> palindrome('laval')
True
>>> palindrome('hello')
False

Bon chance