Présentation, Examen durée 4 heures

Vous devez choisir soit, le problème Juniper Green soit, faire les trois exercices complétements pour valider votre examen sur Python.
Faites un dossier Juniper et un fichier app.py pour le problème. Si vous choisissez les exercices : faites un dossier exercices et séparez chaque exercice dans un fichier à part : ex1.py, ex2.py et ex3.py
Une fois terminé vous m'enverrez vos productions à mon adresse mail.
Bon courage et bonne lecture.

Problème Juniper Green 1/2 règles

Présentation des règles du jeux :

- Le joueur 1 choisi un nombre compris entre 1 et 100
- À tour de rôle, chaque joueur choisi un nombre parmi les multiples ou diviseurs du nombre choisi précédemment par son adversaire, ce nombre est également inférieur à 100.
- Un nombre déjà joué ne peut être rejoué.
- Le perdant est le joueur qui ne peut plus proposer un multiple ou diviseur.

Problème Juniper Green 2/2 fonctions utiles 5pts

Écrivez les fonctions utiles suivantes pour le jeu :

Générer une liste des valeurs possibles :
$possibles = list(range(1, 101))$. À chaque fois qu'un joueur choisira une valeur il faudra retirer cette valeur de la liste des valeurs possibles, utilisez la méthode remove :
$possibles.remove(5)$ retirer 5 de la liste si il existe.

Créez les deux fonctions suivantes :

possible_multiple(n), cette fonction retournera tous les multiples possibles de n, compris entre 2*n et 100.

possible_divisor(n), même chose cette fonction ajoutera tous les diviseurs de n compris entre d = 2 et 100.

Pour fusionner deux listes, l, m vous pouvez, en Python, utiliser l'opérateur + : $l + m$.

Exercice 1

On souhaite mettre en place un algorithme permettant d'odronner une liste de nombres par ordre croissant en utilisant un tri particulier : **tri à bulle**.
Au départ on compare chaque nombre de la liste à trier deux à deux et on permute, à chaque comparaison, le nombre le plus grand est placé à droite de la liste, une fois la liste parcourue en entier, on aura déplacé le plus grand nombre complétement à droite. On répéte l'opération sur la même liste moins le dernier nombre qui est déjà placé complètement à droite, ainsi de suite. L'algorithme s'arrête lorsqu'on ne peut plus parcourir la liste, elle est alors ordonnée par ordre croissant.
Mettez en place cet algorithme en Python.

Exercice 2

Soit la chaîne de caractères text suivantes :
text="aaabbhhdddcccjjhhggffqqllkki iooouuurrrrpppppqqqooo okkk llllooooppprrrr tttt"
Écrire un script qui permet de compter le nombre d'occurences de chaque lettre dans la chaîne de caractères text. Les espaces ne sont pas à compter. Vous placerez le résultat dans un dictionnaire : D ={ 'a' : 3, ... }.
Puis vous calculerez la fréquence de chaque lettre dans le dictionnaire D dans une liste frequences.

Exercice 3

Soient les deux listes suivantes (voir à la fin de l'énoncé) : colors et values qui correspondent au couleurs et valeurs d'un jeu de 32 cartes. Vous devez générer, dans une liste, le jeu de 32 cartes puis, le mélanger et enfin simuler une pioche dans ce jeu.
Pour finir vous devez écrire un script permettant de simuler le jeu suivant : le jeu est gagnant si on tire un roi. Vous afficherez un message de succès ou d'échec en suivant cette condition.
Les données des deux listes :
values = {'7','8', '9', '10', 'Valet', 'Dame', 'Roi', 'As'} et colors = {'piques', 'carreau', 'treffle', 'coeur'}