Préambule

Les exercices suivants sont des exercices type concours, posés ou proposés pour l'oral.

Exercice 1

1. On considère le code Python de la fonction **d** suivante :

```
def d(n):
        L = [1]
        for nombre un range(2, n + 1):
                if n % nombre == 0:
                        L.append(nombre)
        return L
```

   Quel est le résultat de l'appel **d(4)** ? Puis de l'appel **d(10)** ?
   Que fait la fonction **d** ?
2. Un *diviseur nom trivial* d'un entier **n** est un diviseur de **n** différent de 1 et de **n**. Ecrire une fonction **DNT**, d'argument **n**, renvoyant la liste des diviseurs non-triviaux de l'entier **n**.
3. Ecrire une fonction sommeCarresDNT, d'argument **n**, renvoyant la somme des carrés des diviseurs non-triviaux de l'entier **n**. Ecrire la suite des instructions permettant d'afficher tous les nombres entiers inférieurs à 1000 égaux à la somme des carrés de leurs diviseurs non-triviaux. Que peut-on conjecturer ?

Exercice 2 Une matrice carrée d'ordre n est dite "magique" si elle contient tous les nombres de 1 à n^2 et si les sommes des nombres de chaque ligne, de chaque colonne et de chaque diagonale sont toutes égales à une même constante **s** .

1. Au brouillon, exprimer la constante **s** en fonction de **n**.
2. Créer une fonction **EstMagique** , d'argument une matrice **T** (carrée de taille n) et qui renvoie un booléen indiquant si **T** est magique ou pas.
   Tester cette fonction avec les matrices :

$$\text{A} = \begin{pmatrix} 4 & 9 & 2 \\ 3 & 5 & 7 \\ 8 & 1 & 6 \end{pmatrix} \text{ et B} = \begin{pmatrix} 1 & 8 & 2 \\ 4 & 5 & 7 \\ 6 & 9 & 3 \end{pmatrix}$$

   Une méthode permettant de construire une matrice magique de taille impaire n = 2 p + 1 est la suivante :
   - On construit une matrice de taille n remplie de zéros. On considère cette matrice comme la représentation sur une période d'une matrice infinie n-périodique en lignes et en colonnes.
   - On remplace ensuite les zéros de la matrice avec les nombres de 1 à n^2 comme suit :
     - on met le 1 dans la case située sous la case centrale de la matrice ;
     - on place ensuite chaque nombre de 2 à n 2 dans la case située ligne suivante et colonne suivante de celles où on a mis le nombre précédent. Si cette case est déjà remplie, on avance encore d'une ligne et on recule d'une colonne.

   On admet que la matrice ainsi construite est magique.
3. Construire "à la main", selon cette méthode, une matrice magique d'ordre 3.
4. Écrire une fonction **Magique** d'argument un entier p qui renvoie la matrice magique de taille 2p + 1 créée à l'aide de la méthode précédente.

Exercice 3

1. On considère un nombre n . Que donne la commande list(str(n)) ?
2. Écrire une fonction nommée **somme** d'argument un nombre entier naturel et qui renvoie la somme de ses chiffres.
3. On considère qu'un nombre est "adéquat" si la somme de ses chiffres est un multiple de 10. Écrire une fonction nommée **adequat** d'argument un entier naturel n et qui renvoie un booléen indiquant si n est "adéquat", ou pas.
4. Écrire une fonction modification d'argument un entier naturel n qui renvoie un nombre p ayant les mêmes chiffres des dizaines, centaines, etc, que n et avec un chiffre des unités tel que p soit adéquat. Si n est déjà adéquat, on aura n = p.
5. Tester cette fonction en demandant d'afficher **adequat(modification(n))** pour 10 valeurs aléatoires de n entre 10 000 et 100 000 (fonction randint du module random).
6. Tirer au hasard plusieurs milliers d'entiers entre 1 et 100 000 et calculer la proportion de nombres adéquats. Ce résultat vous surprend-il ?

Exercice 4 Soit n un entier naturel impair et non multiple de 5. On admet qu'un de ses multiples noté N s'écrit en base 10 avec seulement le chiffre 1 (de la forme 111 ů ů ů 111). Le but de cet exercice est de trouver pour un n vérifiant ces conditions le plus petit multiple N de cette forme, et de déterminer le nombre de 1 nécessaires.

1. Écrire une fonction **QueDesUn** d'un argument n qui renvoie le premier multiple N de n ne s'écrivant qu'avec des 1, si n est impair non multiple de 5, et 'erreur' sinon.
2. Pour n allant de 1 à 150, pour les n impairs et non multiples de 5 afficher les couples (n, N) .
3. Écrire une fonction **Longueur(k)**, donnant le nombre de chiffres de l'entier naturel k . [Indication : la fonction log ne pouvant pas s'appliquer aux entiers longs, on pourra compter le nombre de divisions entières par 10 que l'on peut faire pour obtenir finalement 0.]
4. Pour n allant de 1 à 1000, chercher le(s) couple(s) (n, N) tel que le nombre de 1 de N soit le plus grand. (En cas d'égalité, donner toutes les solutions.)