Exercice 1

Les habitants de paris paient l'impôt selon les règles suivantes :

- les hommes de plus de 20 ans paient l'impôt

- les femmes paient l'impôt si elles ont entre 18 et 35 ans

- les autres ne paient pas d'impôt

Le programme demandera donc l'âge et le sexe du parisien, et se prononcera donc ensuite sur le fait que l'habitant est imposable.

Exercice 2

Une autre boucle while : calculez la somme d'une suite de nombres positifs ou nuls. Comptez combien il y avait de données et combien étaient supérieures à 100.

Entrer un nombre inférieur ou égal à 0 indique la fin de la suite.

Exercices 3

L'utilisateur donne un entier positif et le programme annonce combien de fois de suite cet entier est divisible par 2.

Exercices 4

L'utilisateur donne un entier supérieur à 1 et le programme affiche, s'il y en a, tous ses diviseurs propres sans répétition ainsi que leur nombre. S'il n'y en a pas, il indique qu'il est premier. Par exemple :

```
Entrez un entier strictement positif : 12
Diviseurs propres sans répétition de 12 : 2 3 4 6 (soit 4 diviseurs propres)
Entrez un entier strictement positif : 13
Diviseurs propres sans répétition de 13 : aucun ! Il est premier
```

Exercices 5 :

Un permis de chasse à points remplace désormais le permis de chasse traditionnel. Chaque chasseur possède au départ un capital de 100 points. S'il tue une poule, il perd 1 point, 3 points pour un chien, 5 points pour une vache et 10 points pour un ami. Le permis coûte 200 euros.

Écrire une fonction amende qui reçoit le nombre de victimes du chasseur et qui renvoie la somme due.

Utilisez cette fonction dans un programme principal qui saisit le nombre de victimes et qui affiche la somme que le chasseur doit débourser.

Exercice 6 :

Un programme principal saisit une chaîne d'ADN valide et une séquence d'ADN valide (valide signifie qu'elles ne sont pas vides et sont formées exclusivement d'une combinaison arbitraire de "a", "t", "g" ou "c").

Écrire une fonction valide qui renvoie vrai si la saisie est valide, faux sinon.

Écrire une fonction saisie qui effectue une saisie valide et renvoie la valeur saisie sous forme d'une chaîne de caractères.

Écrire une fonction proportion qui reçoit deux arguments, la chaîne et la séquence et qui retourne la proportion de séquence dans la chaîne (c'est-à-dire son nombre d'occurrences).

Le programme principal appelle la fonction saisie pour la chaîne et pour la séquence et affiche le résultat.

Exemple d'affichage :

```
chaîne : attgcaatggtggtacatg
séquence : ca
Il y a 10.53 % de "ca" dans votre chaîne.
```

Exercice 7 :

Fonction renvoyant plusieurs valeurs sous forme d'un tuple.

Écrire une fonction minMaxMoy() qui reçoit une liste d'entiers et qui renvoie le minimum, le maximum et la moyenne de cette liste. Le programme principal appellera cette fonction av

ec la liste : [10, 18, 14, 20, 12, 16].

Exercice 8 :

Saisir un entier entre 1 et 3999 (pourquoi cette limitation ?). L'afficher en nombre romain.

Exercice 9

Un tableau contient n entiers ($2<n<100$) aléatoires tous compris entre 0 et 500. Vérifier qu'ils sont tous différents.

Exercice 10 :

Nombres parfaits et nombres chanceux.

- On appelle nombre premier tout entier naturel supérieur à 1 qui possède exactement deux diviseurs, lui-même et l'unité.
- On appelle diviseur propre de n, un diviseur quelconque de n, n exclu.
- Un entier naturel est dit parfait s'il est égal à la somme de tous ses diviseurs propres.
- Un entier n

tel que : $(n+i+i_2)$ est premier pour tout i dans $[0,n-2]$ est dit chanceux.

Écrire un module (parfait_chanceux_m.py) définissant quatre fonctions : somDiv, estParfait, estPremier, estChanceux et un autotest.

- La fonction somDiv retourne la somme des diviseurs propres de son argument.
- Les trois autres fonctions vérifient la propriété donnée par leur définition et retourne un booléen. Si par exemple la fonction estPremier vérifie que son argument est premier, elle retourne True, sinon elle retourne False.