Exercice 2.1

Créer un programme qui possède 4 fonctions : `addition()`, `soustraction()`, `multiplication()` et `division()`.

Chacune des fonctions prend deux paramètres et renvoient toutes une valeur.

La fonction `division()` renvoie une liste de deux valeurs :

- Le résultat de la division décimale.
- Une séquence contenant le résultat de la division euclidienne et le reste.

Faire quelques opérations dans le *main*.

Précision : entrer directement les valeurs des paramètres dans les appels de fonction, ne pas faire de input.

Exercice 2.2

A partir de l'exercice précédent, adapter les fonctions `addition()` et `soustraction()` pour qu'elles puissent prendre plusieurs paramètres (au minimum deux).

Vérifier également sur toutes les fonctions que les valeurs entrées soient bien numériques.

Exercice 2.3

Créer une fonction générateur qui reproduit exactement le même comportement que **`range()`** (avec 1, 2 ou 3 paramètres). Il ne faut pas utiliser de **`list`**

Exercice 2.4 – La bataille navale

Créer une classe *Navire*.

Un navire possède, au minimum : un nom, une couleur, un kilométrage.

Etant donné qu'il s'agit de navires de guerre, on indiquera une puissance de tir, une résistance (ce sont des `float` de 0 à 100, 0 = faible, 100 = fort), et des points de vie correspondant à l'état du navire (un `float` de 0 à 100, 100 = aucun dégât, 0 = navire détruit).

Créer le constructeur pour la classe *Navire*.

Exercice 2.5

L’état du navire est déterminé par le nombre de points de vie. Deux états possibles : le navire est détruit (si les points de vie sont nuls) ou non.

Exercice 2.6

On veut être en mesure afficher les informations du navire dans la console de la forme suivante :

« [NOM]
Couleur – Kilométrage actuel
Puissance de tir : XXX
Résistance : XXX
Points de vie : XXX/100
Etat : *En marche* ou *Détruit*
---------- »

Précisions : ne pas utiliser de print ici.

Exercice 2.7

Dans le *main* du programme, créer plusieurs navires différents (au moins trois) et afficher leurs informations.

Exercice 2.8

Ajouter une méthode `naviguer()`, qui prend en paramètre un certain nombre de milles nautiques.

Naviguer aura donc pour effet d’augmenter le kilométrage.

À chaque déplacement, afficher dans la console (*N1* correspond au nom du navire qui navigue) :

« *N1* navigue.
xxxx NM parcourus.
Kilométrage actuel : *xxxxx* NM.
---------- »

Exercice 2.9

Ajouter une méthode `subir_degats()`, qui prend un paramètre un certain nombre.

Il faudra déduire ce nombre passé en paramètre aux points de vie. Les points de vie ne peuvent pas être négatifs.

Pour rappel un navire dont les points de vie sont nuls est détruit.

Exercice 2.10

Ajouter une méthode `tirer_sur()`, qui prend un paramètre correspondant à un navire ennemi. Tirer sur un navire aura pour effet de faire subir des dégâts à ce navire.

Ce niveau de dégâts sera déterminé par :

- la puissance de tir du navire qui tire ;
- la résistance du navire qui subit le tir ;
- un facteur aléatoire.

Voici la formule :

*Dégâts infligés = (((0,5 + random1) * puissance de tir) - (random2 * résistance)) * random3*

random1, *random2* et *random3* sont trois nombres aléatoires différents.

Pour générer un nombre aléatoire, on utilise ***`random.random()`*** *(il faudra indiquer* ***`import random`*** *en première ligne du fichier).*

Utiliser la fonction ***`round()`*** *pour arrondir le résultat.*

Si le résultat du calcul est nul ou négatif, on considère que le tir a échoué, on n'inflige alors aucun dégât.

Exercice 2.11

En réalité, un navire est plus complexe que cela. Il possède une coque, et des armes.

Créer une classe *Coque*, qui possède une résistance, des points de vie, une matière et une couleur. Supprimer les attributs redondants dans la classe *Navire*.

Créer une classe *Arme*, qui possède un nom, une puissance de tir et un prix. Supprimer les redondances dans la classe *Navire*. Pour le moment, un navire ne possède qu'une seule arme.

Adapter / déplacer les méthodes existantes si nécessaire.

Dans la méthode permettant d'afficher les informations d'un navire, remplacer l'affichage de la couleur, puissance de tir, résistance et points de vie par les informations sur la coque et l'arme.

Exercice 2.12

Dans le *main*, adapter les navires existants suite aux modifications.

Chaque navire doit se déplacer 3 fois et parcourir 2000 NM au minimum. Chaque navire doit avoir subi au moins un tir, et tirer au moins une fois sur chaque navire (si son état le permet).

Afficher les informations de chaque navire à la fin du programme.

Exercice 2.13 – Bonus

Ajouter une méthode permettant de déterminer le navire vainqueur après chaque bataille. On se basera sur les points de vie restants.

Exercice 2.14 – Bonus

Ajouter une méthode permettant d'établir le classement après chaque bataille. Gérer le cas des ex-aequo.