Exercice 1

Créer une fonction qui prend un dictionnaire en argument.
La fonction renverra la clé du dictionnaire qui a la plus grande valeur. En cas d’égalité, la première clé à avoir la plus grande valeur sera retournée.

Dans un autre fichier, créez un dictionnaire avec au moins 5 noms auxquels vous associer un score de 1 à 10. Faites appel à la fonction pour déterminer qui a le score le plus haut.

Exercice 2

Écrivez un petit programme qui prend une liste en argument et qui renvoie un dictionnaire avec trois éléments:

- à la clé “min”: la valeur la plus petite de la liste
- à la clé “max”, la valeur la plus grande de la liste
- à la clé “avg”, la valeur moyenne de la liste

Dans un autre fichier, créer une liste d’au moins trois éléments, et à l’aide de la fonction que vous venez de créer, affichez la valeur la plus petite, plus grande et moyenne de cette liste.

Exercice 3

Ecrivez une classe Flower qui modélise une fleur.

Comme attributs:

- Un niveau de croissance (commence à 0)
- Un prix de vente qui vaut 5

Comme méthode:

- Une méthode "water" qui arrose la plante (c'est à dire qui augmente de 1 son niveau de croissance.

Créez une instance de fleur, affichez son prix, arrosez-là deux fois, et affichez son niveau de croissance.

Exercice 4

Modifiez votre classe Flower :

- La méthode water() ne doit faire grandir la fleur qu'une fois sur 3.
- La fleur ne peut pas augmenter son niveau de croissance à plus de 3.
- Ajoutez dans la classe une méthode sell() qui renvoie le prix de vente effectif de la fleur, calculé comme suit:
    - Prix de vente * niveau de croisance

Ensuite, utilisez votre classe dans un petit programme qui crées une instance de Flower, et qui l'arrose tant qu'elle n'a pas grandit à son niveau maximal, et qui affiche ensuite le nombre d'arrosages qui ont été nécessaires, ainsi que le prix final de vente de la fleur.

Exercice 5

Ecrivez une classe pour un petit animal.

Votre animal aura un attribut pour sa joie qui sera symbolisé par un entier et qui a une valeur initiale de 5 et un autre, aussi un entier pour symboliser sa faim et qui commencera à 0.

Il a deux méthode:

- **`feed`** qui descend sa faim de 2 avec un minimum de 0.
- **`pet`** qui augmente sa joie de 3 avec un maximum de 10 et augmente aussi sa faim de 1 avec un maximum de 10.

Créer une instance de cette classe, utilisé **`pet`** 3 fois et **`feed`** 2 fois et ensuite afficher les deux attributs.

Exercice 6

Modifier la classe de votre Animal:

- Ajoutez un attribut dédié au sommeil. C'est aussi un entier qui a pour valeur initiale 0.
- Dans la méthode **`feed`** faite en sorte que si le sommeil est égale à 10 l'animal ne puisse pas se nourrir. Si il peut se nourrir, le sommeil augmente de 1 en plus de l'autre comportement (maximum 10).
- Ajoutez un méthode **`play.`** Cette méthode ajouter 4 à sa joie (maximum 10), ajouter 2 à la faim (maximum 10) et ajoute aussi 4 au sommeil (maximum 10).
- Ajoutez un méthode **`sleep`** qui descend le sommeil de 3 (minimum 0) et descend la joie de 2 (minimum 0)

Créer un petit script qui instancie la classe. Ensuite utilisé **`play`** tant qu'il n'est pas au maximum de sommeil. Ensuite caressez le (**`pet`**) une fois et nourrissez (**`feed`**) une fois aussi. Affichez ensuite sa joie et sa faim.

Exercice 7

créez une classe **`Fruit`**, cette classe à deux attributs **`name`** et **`nutritional_value`**
Ces deux attributs sont initialisé via les arguments du constructeur.

Ensuite créez une classe **`Kirky`**

Cette classe à deux attributs:

- un attribut dédié à la vie nommé **`hp`** qui est égale à 1 au départ.
- un attribut dédié à son poid nommé **`weight`** qui a aussi 1 comme valeur initiale.

Elle a aussi une méthode **`eat`** qui prendra un fruit en argument. Cette méthode ajoute à l'attribut **`hp`** la **`nutritional_value`** du fruit et incrémente l'attribut **`weight`** de **`1`**.

Exercice 7 bis

Créer deux enfants à la classe **`Fruit`**: **`Pear`** et **`Apple`**.
`Pear` à une **`nutritional_value`** aléatoire allant de 1 à 3 et son attribut name vaudra `"poire"`.
`Apple` à une **`nutritional_value`** valant 2 et un nom aléatoire choisis parmis les noms suivant: `"Jonagold","Granny Smith" et "Golden"`

`Changer la méthode` **`eat`** de la classe **`Kirby`** pour afficher une phrase qui énonce le fruit mangé.

Enfin, écrivez un petit script qui crée une liste contenant deux instances de **`Apple`** et deux instances de **`Pear`**. Tant que la liste n'est pas vide faites manger les **`Fruits`** à une instance de **`Kirby`** (via la méthode **`eat`**). Pour terminer, afficher le poids et la vie du **`Kirby`**.

Exercice 8a

Créer deux classes, une classe **`Mario`** et une classe **`Luigi`**:

`Mario` a pour attribut:

- **`is_dead:`** représenté par un boolean: valeur initiale **`false`**.
- **`points:`** représenté par un entier: valeur initiale **`0`**.
- **`jump_height:`** représenté par un float: valeur initiale **`4`**.

`Luigi` a pour attribut:

- **`is_dead:`** représenté par un boolean: valeur initiale **`false`**.
- **`points:`** représenté par un entier: valeur initiale **`0`**.
- **`jump_height:`** représenté par un float: valeur initiale **`4.5`**.

Exercice 8b

Créer trois autres classes: **`Goomba`**, **`Turtle`** et **`Mushroom.`**

`Goomba` a pour attribut:

- **`point:`** représenté par un entier valeur initiale **`100`**.
- **`deadly:`** représenté par un booléen valeur initiale **`True`**.

`Turtle` a pour attribut:

- **`point:`** représenté par un entier valeur initiale **`200`**.
- **`deadly:`** représenté par un booléen valeur initiale **`True`**.

`Mushroom` a pour attribut:

- **`point:`** représenté par un entier valeur initiale **`1000`**.
- **`deadly:`** représenté par un booléen valeur initiale **`False`**.

Exercice 8C

Ajouter à **Mario** et **Luigi** une méthode **jump** qui prend en argument un **Mushroom** ou un **Goomba** ou une **Turtle**.

Si l'attribut **deadly** de l'instance passée en paramètre est **True** alors il y a une chance sur deux que l'attribut **is_dead** de **Mario** ou de **Luigi** deviennent lui aussi **True**
Si l'attribut **is_dead** est **False**, alors la méthode ajoute au point de **Mario** ou de **Luigi** les point de l'instance passée en paramètre.

Ensuite faite un script qui crée un **Mushroom**, un **Goomba** et une **Turtle**.
Créez aussi un **Mario** et un **Luigi.**
Faites sauter (via **jump**) les instances de **Mario** et **Luigi sur** le **Mushroom**, le **Goomba** et la **Turtle.** Ensuite afficher leurs points et si ils sont en vie.

Exercice 9A

Nous allons créer un micro-jeu de gestion textuel:

Première étapes on va créer des lieux:

La forêt qui a un attribut **`ressource`** qui est une string et qui vaudra **`"bois"`**
et un autre **`production_max`** qui est un entier et qui vaut **`3`.**
La forêt a une méthode **`extract`** qui renvoie un tuple.
Ce tuple contient deux valeurs:

- la valeur de **`ressource`**
- entier ayant une valeur aléatoire entre 1 et la valeur de **`production_max`**

La mine qui a un attribut **`ressource`** qui est une string et qui vaudra **`"pierre"`**
et un autre **`production_max`** qui est un entier et qui vaut **`3`.**
La mine a aussi une méthode **`extract`** qui a le même comportement que la forêt.

Exercice 9B

Nous allons aussi créer une classe pour symboliser le village.

Le village a un attribut pour simuler la réserve et qui sera un dictionnaire.
Les clés seront des noms de ressources et les valeurs seront les quantités correspondantes. À la base le dictionnaire est vide (`{}`)
Le village a aussi un attribut qui symbolise le nombre d'ouvriers dans le village. Cette attribut commence à `3`.
Enfin le village a un attribut qui symbolise son évolution et sera un entier avec une valeur de base de `1`.

Le village a aussi une méthode `upgrade`. Cette méthode va checker si il y a au moins `5 "bois"` et `5 "pierre"` dans la réserve, si c'est le cas elle les retira de la réserve et augmentera le niveau du village de `1` et son nombre d'ouvriers de `2`.

Exercice 9C

Dans un fichier à part, on va écrire un script.
Ce script va instancier un village, une forêt et une mine.

Vous allez ensuite répéter les instructions suivantes 5 fois:

- afficher le niveau du village et le nombre de travailleurs.
- afficher la réserve (afficher chaque élément du dictionnaire séparément).
- demander à l'utilisateur combien de personnes vont aller à la mine. Si l'utilisateur donne un nombre plus grand que le nombre d'ouvriers, réduisez le au nombre d'ouvriers.
- jouer autant de fois que le nombre d'ouvriers allant à la mine la méthode **`extract`** de la mine. Ajouter le retour de la fonction à la réserve. (le retour est un tuple dont le premier élément est la clé et le deuxième élément est le nombre à ajouter).
  Attention, si la clé n'existe pas encore, il faudra la créer.
- faite la même chose avec la forêt pour le nombre de travailleurs restant.
- Jouer la méthode upgrade du village.
  (les ouvriers "retourne" dans le village à la fin d'un cycle)

Au bout des 5 fois afficher le niveau du village.

Exercice 10-A

Ecrivez une classe qui symbolise un arme. L'arme a un attribut pour les dégâts (qui sera un entier) et un autre pour la solidité qui est aussi un entier.

Une arme a aussi une méthode **`attack`** qui retire **`1`** à sa solidité et renvoie la valeur des dégâts si sa solidité est plus grande que **`0`**, sinon elle renvoie **`1`**.

Créer trois classes qui hérite de l'arme:
un katana (dégât 5:, solidité: 3), une étoile du matin (dégât: 4, solidité: 4) et une fourche (dégât: 2 solidité: 6).

Exercice 10-B

Ecrivez une classe pour symboliser un aventurier.
L'aventurier à trois attributs, des points de vie (un entier), une force (un entier aussi) et un arme (qui sera une instance d'une des classes créées précédemment).

L'aventurier a aussi une méthode **`hit`**.
Cette méthode prend en argument un autre aventurier (donc une autre instance de la classe).

La méthode récupère la valeur renvoyée par la méthode **`attack`** de l'arme et multiplie cette valeur par l'attribut symbolisant la force. Ensuite elle va retirer le résultat au point de vie de l'aventurier passé en argument.

Exercice 10-C

Dans un script à part, créez un aventurier avec un katana, 3 point de force et 50 point de vie.

Créez un autre aventurier avec une fourche, 1 point de force et 5 point de vie.

Faites en sorte que l'aventurier à la fourche frappe (méthode hit) l'aventurier au katana et vis-versa.

Ensuite créez un aventurier avec une étoile du matin, 3 force et 10 point de vie.

Faites en sorte que l'aventurier à l'étoile du matin frappe (méthode **`hit`**) l'aventurier au katana et vis-versa. Répétez cette dernière action jusqu'à ce que l'un des deux n'est plus de point de vie.

Enfin affichez les points de vie de tous les aventuriers.