Exercice 4

On n'oubliera pas de charger l'environnement virtuel défini lors de la première séance à l'aide de la commande `source`.

4.1 Modularité (suite)

- Modifier le fichier `euler_project.py`, créé lors du précédent TD, de telle sorte à ce que l'interaction avec l'utilisateur (saisie du numéro de projet à afficher) ne soit effective que lorsque le fichier est exécuté en tant que script.
- Faire en sorte que le fichier `euler_project.py` puisse être charger en tant que module dans un script `test_euler_project.py` qui se chargera de lancer et donc de tester chaque fonction sans intervention de l'utilisateur.
- Déplacer ce module/script dans un répertoire référencé par la variable `PYTHONPATH`

4.2 Objets python

1. Classe `Particle`
   (a) Dans un fichier `particle.py`, créer une classe/objet `Particle` qui prendra pour attributs, le nom de la particule, sa masse exprimée en eV et sa charge électrique. La méthode d'initialisation `__init__` permettra de fournir des valeurs par défaut à ces trois attributs tout en autorisant l'utilisateur à initialiser ces attributs.
   (b) Ajouter une méthode `dump` permettant d'afficher les valeurs des attributs et concevoir un programme `test_particle.py` qui créera diverses instances de type `Particle` en les stockant dans une liste puis affichera chacune de ces instances.
   (c) Renommer la méthode `dump` en `__str__` et faire en sorte qu'elle retourne une chaîne de caractères. Tester la fonction `print` sur un objet de type `Particle`.
2. Classe `Point`
   (a) Dans un fichier `Point.py`, créer une classe `Point` dont les attributs seront les valeurs des coordonnées cartésiennes x et y. Surcharger la méthode `__str__` afin d'afficher ces deux informations. Pour tester l'ensemble, on créera un programme test dans lequel diverses instances de type `Point` seront générées.
   (b) Définir une nouvelle méthode appelé `__add__` qui retournera un nouvel objet de type `Point`, résultat de la somme de deux instances de type `Point` .
   (c) Afficher le résultat de la somme de deux objets `Point` *via* la fonction `print`.
   (d) Créer une nouvelle classe `Vector2D` dont les attributs seront deux objets de type `Point`. Définir une méthode de `Vector2D` qui retournera la norme du vecteur.
   (e) Définir une méthode de `Vector2D` qui permettra d'afficher les coordonnées des deux points constituant le vecteur et que l'on pourra utiliser par le biais de la fonction `print`.