TP1: Boule de flipper.

L'objectif de ce TP est d'utiliser le langage **Python** pour programmer le déplacement d'une "boule de flipper" (qui sera plus prosaïquement un cercle du plan euclidien). On utilisera pour modéliser ce déplacement les résultats du cours de géométrie analytique.

Quelques commandes Python qui vous seront utiles dans ce TP:

Sous **Python** on charge d'abord la bibliothèque **numpy** qui permet de gérer les tableaux et qui contient les principales fonctions mathématiques:

import numpy **as** np

Pour ce TP nous aurons besoin des commandes suivantes:

1. **np.arange(0,2,0.5)**
   Le résultat est la création du tableau : **array([0. , 0.5, 1. , 1.5])**

2. On peut aussi créer un tableau de valeurs équidistantes avec "linspace":
   np.linspace(0,2,5)
   Le résultat est la création du tableau : **array([0. , 0.5, 1. , 1.5, 2])**

3. **np.pi** : pour obtenir une valeur approchée du nombre pi

4. **np.cos(teta)** et **np.sin(teta)** pour les fonctions trigonométriques cosinus et sinus. Si **teta** est un tableau le résultat de ces deux fonctions est un tableau:
   teta=np.arange(0,2*np.pi,np.pi/6)
   np.cos(teta)
   Out[4]:array([1.00000000e+00, 8.66025404e-01, 5.00000000e-01, 6.12323400e-17, -5.00000000e-01, -8.66025404e-01, -1.00000000e+00, -8.66025404e-01, -5.00000000e-01, -1.83697020e-16, 5.00000000e-01, 8.66025404e-01])

5. L'addition d'un tableau et d'une constante réelle permet d'additionner cette constante à toutes les valeurs du tableau:
   T=np.arange(0,5,1)
   T
   Out[6]: array([0, 1, 2, 3, 4])
   T+3
   Out[7]: array([3, 4, 5, 6, 7])

On charge ensuite la bibliothèque **pyplot** pour le tracé de courbes:

import matplotlib.pyplot **as** plt

Cette bibliothèque dispose de la fonction **plt.plot (X,Y)** qui permet de définir un nuage de points reliés par des segments. L'argument **X** est un tableau qui contient les abscisses successives de ces points tandis que **Y** contient les ordonnées. Pour tracer ce nuage de points on utilise la commande **plt.show()**.
On peut tracer plusieurs nuages de points en entrant successivement les abscisses et les ordonnées de ces nuages:
plt.plot (X1,Y1,X2,Y2)
On peut donner un nom à chaque nuage de points:
l1,l2=plt.plot(X1,Y1,X2,Y2)
plt.show()
Ces 2 commandes permettent à la fois de tracer les deux nuages et elle stocke les données de chaque nuage associé au tracé dans **l1** et **l2** respectivement.

(Attention: s'il n'y a qu'un nuage il faut taper :
l1,=plt.plot(X1,Y1) (la virgule est obligatoire))
On peut obtenir un nouveau tracé en ne modifiant que le premier nuage, sans toucher au second, par les commandes suivantes:
l1.set_data(nouveauX,nouveauY)
plt.show()

TP: réalisation d'une animation du déplacement d'une boule de flipper

Partie 1: tracé d'un cercle.

On considère que le plan euclidien est muni d'un repère orthonormal $R = \left(0, \vec{i}, \vec{j}\right)$.

1. Donnez une équation paramétrée du cercle B de centre $C \begin{pmatrix} x_c \\ y_c \end{pmatrix}$ et de rayon r (vous pouvez aller chercher l'information à la fin du cours de géométrie analytique dans le plan qui est sur l'ENT).
2. Déduisez-en les commandes python pour tracer ce cercle à l'aide de 100 segments de même longueur (on notera BX le tableau des abscisses et BY celui des ordonnées).

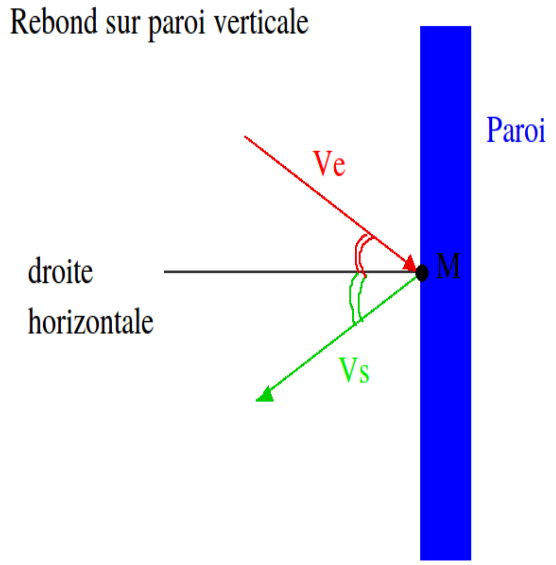
Partie 2: déplacement du cercle.

Dans cette partie on considère que la boule se déplace dans un plan infini sans obstacles. Il s'agit de comprendre le mécanisme d'une animation sous Python:
En vous inspirant du programme ci-dessous, faites un programme qui simule le déplacement d'un cercle de rayon r=0,1 en partant du point $A \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ et en arrivant au point $B \begin{pmatrix} 10 \\ 5 \end{pmatrix}$ avec une vitesse constante (une unité par seconde).

```
 8 import numpy as np
 9 import matplotlib.pyplot as plt
10 dt = 0.01
11 nImage=100
12 #définition du carré:
13 #les abscisses
14 x=np.array([0,1,1,0,0])
15 #les ordonnées:
16 y=np.array([0,0,1,1,0])
17 #boucle pour "l'animation":
18 for i in range(nImage):
19     if i == 0:
20         line, = plt.plot(x, y)
21         plt.axis([-1,10,-1,10])
22     else:
23         x=x+0.1
24         y=y+0.1
25         line.set_data(x, y)
26     plt.pause(dt) # pause avec duree en secondes
```

Partie 3: modélisation du rebond sur des parois fixes

On veut modéliser le rebond élastique (sans perte de quantité de mouvement) de la boule sur une paroi verticale puis sur une paroi horizontale.



La "boule" arrive sur la paroi au point M avec un vecteur vitesse $\overrightarrow{V_e}$ et repart avec un vecteur vitesse $\overrightarrow{V_s}$. Les deux vecteurs vitesses sont de même norme. La représentation graphique du vecteur $\overrightarrow{V_s}$ (en vert sur la figure) est l'opposée du symétrique du vecteur $\overrightarrow{V_e}$ par rapport à la droite horizontale.

Sur la figure ci-dessus le bord de la paroi où a lieu le contact a pour équation $x = h$ (avec $h > 0$) et la boule avant le contact se déplace dans le demi-plan d'équation $x \leq h$.

1. Donnez, en fonction des coordonnées de $\overrightarrow{V_e}\begin{pmatrix} v_x \\ v_y \end{pmatrix}$ les coordonnées de $\overrightarrow{V_s}$.

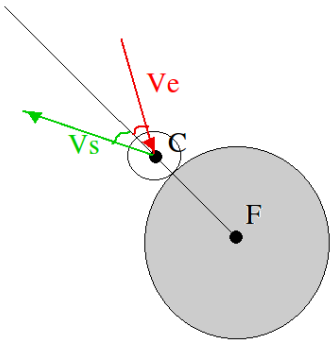
2. On considère la boule B de centre $C\begin{pmatrix} x_c \\ y_c \end{pmatrix}$ et de rayon r . Quelles conditions doivent vérifier les coordonnées de C pour être sûr qu'il y a eu un contact avec la paroi d'équation $x = h$ avec $h > 0$. On considère que la boule est, avant le contact, dans le demi-plan d'équation $x \leq h$.

3. Quelles conditions doivent vérifier les coordonnées de C pour être sûr qu'il y a eu un contact avec la paroi d'équation $x = 0$. On considère que la boule est, avant le contact, dans le demi-plan d'équation $x \geq 0$.

4. Reprenez les questions 1,2 et 3 pour des parois horizontales ($y = 0$ et $y = h$).

5. Programmez une animation de la boule de flipper avec des rebonds sur les parois verticales d'équations respectives $x = 0$ et $x = 10$ et sur les parois horizontales $y = 0$ et $y = 10$. Vous choisirez vous-même la position initiale de la boule et la vitesse initiale.

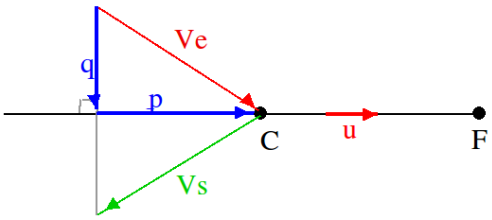
Partie 4: modélisation du rebond sur un obstacle circulaire fixe.



L'obstacle circulaire fixe a pour centre $F\begin{pmatrix} x_f \\ y_f \end{pmatrix}$ et pour rayon R .

La boule de flipper a pour centre $C\begin{pmatrix} x_c \\ y_c \end{pmatrix}$ et pour rayon r . Elle arrive avec une vitesse $\overrightarrow{v_e}$ et repart avec une vitesse $\overrightarrow{v_s}$. Le vecteur $\overrightarrow{v_s}$ est l'opposé du vecteur symétrique de $\overrightarrow{v_e}$ par rapport à la droite (CF)

1. Quelles conditions doit-on avoir sur les coordonnées de C au moment du contact?
2. Que vaut $\left\|\overrightarrow{CF}\right\|$ en fonction des paramètres du problème? (les paramètres du problème sont les coordonnées des centres et les rayons des deux boules, ainsi que les coordonnées de la vitesse $\overrightarrow{v_e}$ de la boule mobile au moment du contact)



3. On considère le vecteur $\overrightarrow{u} = \dfrac{\overrightarrow{CF}}{\left\|\overrightarrow{CF}\right\|}$. Exprimez les coordonnées du vecteur $\overrightarrow{u}$ en fonction des paramètres du problème.
4. Le vecteur $\overrightarrow{p}$ est le projeté orthogonal du vecteur $\overrightarrow{v_e}$ sur la droite (CF) .
   (a) exprimez $\overrightarrow{p}$ en fonction des vecteurs $\overrightarrow{v_e}$ et $\overrightarrow{u}$ et à l'aide du produit scalaire.
   (b) exprimez $\overrightarrow{v_s}$ en fonction de $\overrightarrow{v_e}$ et $\overrightarrow{p}$.
   (c) Déduisez-en les coordonnées de $\overrightarrow{v_s}$ en fonctions des coordonnées de $\overrightarrow{v_e}$, celles de F et de C et des rayons R et r .
5. On garde les parois horizontales et verticales de la partie 3 et on rajoute un obstacle circulaire fixe de centre $F\begin{pmatrix} 3 \\ 3 \end{pmatrix}$ et de rayon $R = 2$. Programmez une animation montrant la simulation du déplacement de la boule de la partie 2 dans ce nouveau cadre.