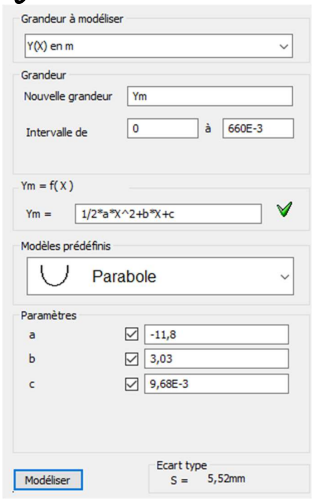
FICHE METHODE

Réaliser un graphe avec python

Le pointage d'un lancer de balle a donné les résultats ci-dessous

	A	B	C
Grd	t	X	Y
Unité	s	m	m
1	0,000	-0,000	-0,000
2	0,040	0,034	0,110
3	0,080	0,072	0,196
4	0,120	0,107	0,269
5	0,160	0,144	0,329
6	0,200	0,183	0,368
7	0,240	0,216	0,392
8	0,280	0,255	0,401
9	0,320	0,297	0,390
10	0,360	0,333	0,364
11	0,400	0,361	0,329
12	0,440	0,404	0,272
13	0,480	0,440	0,199
14	0,520	0,477	0,113
15	0,560	0,510	0,013
16	0,600	0,544	-0,104
17	0,640	0,583	-0,239
18	0,680	0,621	-0,391
19	0,720	0,660	-0,552
20			



Pour apprendre à coder le tracé d'un graphe à l'aide du langage de programmation python, on va utiliser un notebook *capytale*. Ouvrir le note book capytale de code : **5a4c-927196** https://capytale2.ac-paris.fr/web/c/5a4c-927196

Suivre les instructions du notebook pour comprendre la notion de liste et tracer votre premier graphe.

1. Tracer le graphe à partir de points dont on connaît les coordonnées

Les instructions minimales sont ci-contre :

```
# tout d'abord effacer le graphe précédent
plt.close()
# tracé du graphe
plt.plot(X,Y)
plt.show()
```

Améliorer le tracé du graphe

Pour améliorer le graphe basique, il faut modifier la ligne de code *plt.plot(X,Y)*.

Faire les modifications demandées ci-dessous dans la cellule de tracé du graphe du notebook. Exécuter cette cellule et noter vos observations ci-dessous.

Remplacer la ligne *plt.plot(X,Y)* par la ligne ci-contre : `plt.plot(X,Y,'r*')`
Quel est l'effet sur le graphe ?

Maintenant modifiez successivement la ligne comme ci-dessous. Quel sont les conséquences de chaque modification sur le graphe ?

```
plt.plot(X,Y,'r--*')
plt.plot(X,Y,'--b+')
plt.plot(X,Y,'gs')
```

On remarque que l'ordre des instructions entre les ' ' n'a aucune importance sur le résultat
Poursuivez les tests et compléter le tableau ci-dessous

Couleur	**b** bleu	**g**	**r**	**c**	**m**	**y**	**k**	**w**
Marqueur	**+**	*	**^**	**s** Des carrés	**o**	**x**	**.**	
Style de la ligne	**-**		**:** Des pointillés		**--**		**-.**	

Améliorer la zone du graphique

Un graphe en physique doit comporter de nombreuses indications sur l'expérience : grandeur et unité sur les axes, titre de l'expérience réalisée etc..

Titres des axes : instructions **xlabel** et **ylabel**
Dans la cellule de code, programmer votre tracé de graphe tel qu'indiqué.
Exécuter la cellule et visualiser le résultat

```
plt.plot(X,Y,'b+')
plt.xlabel("X (en m)")
plt.ylabel("Y (en m) ")
plt.show()
```

Une grille : instruction **grid**
Ajouter la ligne ci-contre avant plt.show() et observer : `plt.grid()`

Des axes : instruction **axis**
Avant plt.show() , ajouter successivement les lignes suivantes et identifier leur rôle :

```
plt.axis([0,1,-1,1])
plt.axis([0,0.7,0,0.5])
plt.axis([0,0.7,-0.6,0.45])
plt.axis('equal')
```

Un titre : instruction **title**
Avant plt.show(), ajouter la ligne ci-dessous :

```
plt.title("pointage et trajectoire d'un lancer parabolique")
```

Et beaucoup d'autres fonctionnalités à découvrir !

2. Méthode 2 : Tracer une courbe dont on connaît l'équation

Si on connaît l'équation de la courbe à tracer, le début du programme est différent car il faut faire calculer les coordonnées des points à placer sur le graphe.

Avec la modélisation du TP, la balle a pour trajectoire la courbe d'équation : $Y = -5{,}9\ X^2 + 3{,}03\ X + 9{,}68.10^{-3}$.

La bibliothèque numpy permet facilement de créer les couples (X,Y).

*X = np.linspace(**x_min, x_max, 20**)* (crée **20** points équitablement répartis entre **x_min** et **x_max)**
(valeurs numériques de x_min et x_max à renseigner !!)

$Y = a*X^2+b*X+c$ (calcul de chaque Y par l'équation, *les valeurs numériques a, b et c sont à renseigner*)

Dans la dernière cellule, on a les lignes de code suivantes :
Exécuter la cellule et observer.

```
X = np.linspace(0,0.66,20)
Y = -5.9*X**2 + 3.03*X + 0.00968

plt.close()
plt.plot(X,Y,'+')
plt.show()
```

Par la suite, les instructions permettant de mettre en forme la courbe sont les mêmes que dans le 1.

3. Tracer un vecteur sur le graphe

Préalable : il faut impérativement travailler dans un repère orthonormé en utilisant l'instruction plt.axis('equal'). Puis, on utilise l'instruction plt.arrow()

Dans une dernière cellule, on a ajouté au programme précédent les lignes ci-contre

```
plt.axis('equal')

plt.arrow(0,0,0.2,0.5,head_width=0.05,head_length=0.05,color='b',)
plt.arrow(X[7],Y[7],0.1,0,head_width=0.05,head_length=0.05,color='r')
```

Analyser les résultats obtenus pour comprendre le tracé du vecteur. Notamment :

- En anglais rechercher la signification des mots

  "arrow": ……..."head" : ……."width" : ……et "length" : ……..
- Observer l'origine des vecteurs ?
- Observer la position de la fin des vecteurs ?
- Vérifier vos hypothèses en changeant les valeurs identifiées
- A quoi servent les instructions "head_width=0.05" et "head_length=0.05"
- Vérifier vos hypothèses en changeant les valeurs
- Résumer vos observations en indiquant ce qu'il faut placer dans l'instruction arrow

plt.arrow(*terme1 , terme2, terme3, terme4, instruction d'esthétique*)