Terminale Spécialité Physique-Chimie	Thème : Mouvement et interactions	M.KUNST-MEDICA

Chapitre 6 : Mouvement et deuxième loi de Newton

Feuille d'évaluation à rendre obligatoirement avec la copie

Activité numérique n°6.1 : La grande roue parisienne

Inspiré de Belin éducation – Cahier Python-Arduino

	Questions	Compétence visée	Points attribués
Appel n°1	1	S'approprier	/0,5
	2		/0,5
Appel n°2	3	Réaliser	/0,5 /0,5
Appel n°3	4	Analyser, raisonner	/0,5
	5		/0,5
	6		/0,5
Appel n°4	7	Analyser, raisonner, communiquer	/0,5
	8		/0,5
Devoir global	Rendre compte à l'écrit en utilisant un vocabulaire scientifique adapté et présenter son travail sous une forme appropriée et être vigilant vis-à-vis de l'orthographe	Communiquer	/0,25
Total 1 :	**Remarques :**		**/4,75**

Notation individuelle :

CLASSE :	NOMS – PRENOMS des élèves du groupe		Élève n° 1 :		Élève n° 2 :		Élève n° 3 :	
Activité	**Capacités attendues**	**Compétence visée**	**Points attribués**	**Signatures**	**Points attribués**	**Signatures**	**Points attribués**	**Signatures**
Séance en groupe	Travailler en équipe, partager des tâches, s'engager dans un dialogue constructif, respecter ses camarades, son professeur et les lieux de travail ...	Être autonome et faire preuve d'initiative	/0,25		/0,25		/0,25	
TOTAL 2			/0,25		/0,25		/0,25	
Total 1 + 2			/5		/5		/5	

Capacité numérique exigible : Représenter, à l'aide d'un langage de programmation, des vecteurs accélération d'un point lors d'un mouvement.

Annexes :

- **Savoir rentrer des valeurs numériques dans un programme Python**
- **Utiliser un programme Python pour tracer un graphique**
- **Savoir tracer des vecteurs.**
- **Reconnaître les instructions de base : plt, plot(), range(), append().**

Lors de l'Exposition universelle de 1900, une grande roue fait son apparition à Paris. Elle sera démantelée en 1937 par manque de rentabilité. Pourtant, Paris accueille depuis de nombreuses grandes roues, par exemple place de la Concorde ou au jardin des Tuileries.

Partie 1 : Chronophotographie

On étudie le mouvement du centre de masse *M* d'une cabine de grande roue au cours du temps. On prend comme origine du repère le centre de la grande roue. Les positions *x* (m) et *y* (m) du centre de masse de la cabine sont notées à intervalle de temps régulier Δt = 30 s dans le tableau suivant.

Δt (s)	0	30															
x (m)	53	52,74	51,98	50,72	48,97	46,74	44,07	40,97	37,48	33,62	29,44	24,98	20,28	15,38	10,34	5,19	0
y (m)	0	5,19	10,34	15,39	20,28	24,98	29,45	33,62	37,48	40,97	44,07	46,74	48,96	50,72	51,98	52,74	53

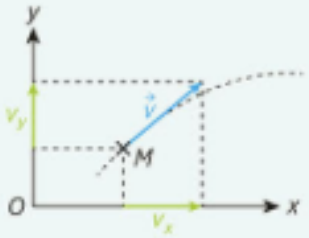
Vecteur vitesse

Avec le point *O* comme origine du repère, le vecteur vitesse instantanée au point *M* se définit comme la dérivée par rapport au temps du vecteur position $\overrightarrow{OM}$:

$$\vec{v} = \frac{d\overrightarrow{OM}(t)}{dt}$$

$$v_x(t) = \frac{dx(t)}{dt} \text{ et } v_y(t) = \frac{dy(t)}{dt}$$

Vecteur vitesse au point M



Vecteur accélération

Le vecteur accélération $\vec{a}$ du point *M* se définit comme la dérivée par rapport au temps du vecteur vitesse :

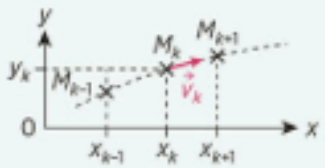
$$\vec{a} = \frac{d\vec{v}(t)}{dt}$$

$$a_x(t) = \frac{dv_x(t)}{dt} \text{ et } a_y(t) = \frac{dv_y(t)}{dt}$$

On peut calculer de manière approchée les composantes à la position *k* :

– de la vitesse instantanée : $v_{x_k} = \frac{x_{k+1} - x_{k-1}}{2\Delta t}$ et $v_{y_k} =$ .....

– de l'accélération : $a_{x_k} = \frac{v_{x_{k+1}} - v_{x_{k-1}}}{2\Delta t}$ et $a_{y_k} =$ .....

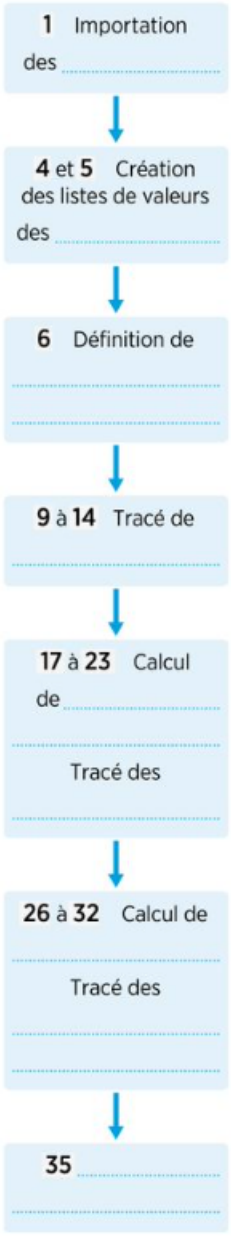


Repère de Frenet

Pour une trajectoire circulaire de centre *O* et de rayon *R*, le repère de Frenet est défini à partir de deux vecteurs unitaires ayant pour origine un point *M* en mouvement :
– le vecteur $\vec{u}_t$ tangent à la trajectoire au point *M*, orienté dans le sens du mouvement ;
– le vecteur $\vec{u}_n$ perpendiculaire à la trajectoire au point *M*, orienté vers l'intérieur de la trajectoire.

Le vecteur accélération $\vec{a}(t)$ s'écrit dans ce repère : $\vec{a}(t) = \frac{dv(t)}{dt}\vec{u}_t + \frac{v(t)^2}{R}\vec{u}_n$.

Partie 2 : Tracé de la trajectoire et des vecteurs vitesse et accélération




```
1   import matplotlib.pyplot as plt
2
3   # Construction des listes de valeurs utiles et définition de l'intervalle de temps
4   x = [.....
    .....]
5   y = [.....
    .....]
6   dt = .....                  # définit l'intervalle de temps en s
7
8   # Tracé de la trajectoire (chronophotographie)
9   plt.plot(x, y, 'x', markersize = 4)
10  plt.xlabel("x (en m)")
11  plt.ylabel("y (en m)")
12  plt.title("Étude du mouvement d'une grande roue")
13  plt.xlim(0, 53)             # définit les limites de valeurs de l'axe des abscisses
14  plt.ylim(0, 53)
15
16  # Tracé des vecteurs vitesse et calcul de la norme
17  Vx = []                     # crée une liste de valeurs de Vx
18  Vy = []
19  for k in range(1, len(x)- 1): # len(x) renvoie le nombre d'éléments contenus dans la liste x
20      Vx.append((x[k + 1] - x[k - 1])/(2*dt))
21      Vy.append(.....)
22  for i in range(0, len(x) - 2):
23      plt.quiver(x[1 + i], y[1 + i], 100*Vx[i], 100*Vy[i], color = "r", scale = 1,
    scale_units = 'xy')         # trace les vecteurs vitesse avec x[1 + i] et y[1 + i] les
    coordonnées du point de départ, Vx[i] et Vy[i] les composantes du vecteur vitesse respec-
    tivement selon l'abscisse et l'ordonnée
24
25  # Tracé des vecteurs accélération et calcul de la norme
26  Ax = []
27  Ay = []
28  for k in range(1, len(x)- 3):
29      Ax.append((Vx[k + 1] - Vx[k - 1])/(2*dt))
30      Ay.append(.....)
31  for i in range(0, len(x) - 4):
32      plt.quiver(x[2 + i], y[2 + i], 10000*Ax[i], 10000*Ay[i], color = "b",
    scale = 1, scale_units = 'xy')
33
34  # Affichage
35  plt.show()
```

Questions :

S'approprier , Réaliser, analyser - raisonner

1. **Compléter** la partie 1

2. **Compléter** le programme Python.

Appel n°1 du professeur pour validation

Réaliser, analyser – raisonner

3. **Exécuter** le programme mis à disposition et complété. **Donner** le diamètre de la grande roue étudiée et caractériser le mouvement obtenu.

.....
.....
.....
.....
.....
.....
.....

Appel n°2 du professeur pour validation

4. Le mouvement étudié correspond-il au démarrage de la grande roue ? **Justifier**

.....
.....
.....
.....
.....
.....
.....

5. En utilisant le tracé des vecteurs accélération obtenus, **donner** l'expression du vecteur accélération dans le repère de Frenet pour le mouvement de la grande roue étudié.

.....
.....
.....
.....
.....
.....
.....

6. **Discuter** de l'orientation du vecteur accélération suivant $\overrightarrow{u_n}$.

.....
.....
.....
.....
.....
.....
.....

Appel n°3 du professeur pour validation

Analyser, raisonner - communiquer

7. La composante $\frac{dv(t)}{dt}\overrightarrow{u_t}$ indique si le mouvement est accéléré ou décéléré. **Discuter** du signe de $\frac{dv(t)}{dt}$ selon le mouvement

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

8. **Conclure** en donnant la signification des différentes composantes du vecteur accélération dans le repère de Frenet selon le mouvement circulaire étudié : uniforme, accéléré ou décéléré.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Appel n°4 du professeur pour validation

Annexe 1 : Savoir rentrer des valeurs numériques dans un programme Python

Calculer en Python

Calcul à la main	Calcul en Python	Résultat en Python
$2+3{,}5$	`2 + 3.5`	`5.5`
3×5	`3*5`	`15`
2^3	`2**3`	`8`
$125{,}3\times 10^{-2}$	`125.3e-2`	`1.253`
$\lvert -2{,}6\rvert$	`abs(-2.6)`	`2.6`

Calcul à la main	Calcul en Python	Résultat en Python
$\frac{3}{8}$	`3/8`	`0.375`
Arrondir $\frac{3}{8}$ à 0,1 près	`round(3/8, 1)`	`0.4`
Division euclidienne de 154 par 6 (quotient et reste)	`154%6` → 4 ; `154//6` → 25	

Types de nombres int et float

Dans un ordinateur, les nombres ne sont pas représentés par leur écriture décimale. On distingue de ce fait deux types de nombres en Python :
– les **entiers**, qui sont représentés de façon exacte. On dit qu'ils sont de type **int** ;
– les autres nombres, qui sont représentés de façon approchée : on dit que ce sont des **flottants** ou qu'ils sont de type **float**. Ils s'écrivent avec un point décimal.

ATTENTION !
- En maths, 4 et 4,0 sont deux écritures du même nombre.
- En Python, 4 et 4.0 ne sont pas le même objet.

Types de nombres et résultats de calcul
- Tout calcul comportant au moins un flottant se fait de façon approchée.

Exemple : `0.1 + 0.7` donne `0.7999999999999999`.
- Pour les entiers, cela dépend de l'opération :

– les additions, soustractions et calculs de puissance d'exposant entier se font de façon exacte, même sur de très grands entiers comme 2^{60} ci-contre ;
– une division a/b donne toujours un flottant ;
– l'utilisation des puissances de 10 avec ...e... donne toujours un flottant.

Exemples
- Le calcul `2**60` donne : `1152921504606846976`
- Le calcul `12/3` donne : `4.0`
- Le calcul `12e4` donne : `120000.0`

Annexe 2 : Utiliser un programme Python pour tracer un graphique

On importe le module `matplotlib.pyplot` avec l'alias `plt`.
Les fonctions importées de ce module seront donc précédées de `plt`.
On termine le programme par `plt.show()` pour faire afficher le graphique.

```
import matplotlib.pyplot as plt
    instructions pour les tracés
plt.show()
```

A. Nuages de points et courbes : plt.plot

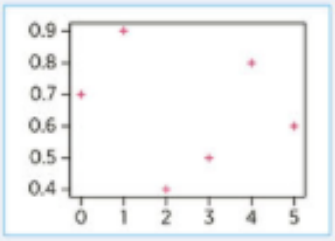
L'instruction `plt.plot(x, y, 'r+')` permet de marquer par le signe + en rouge :
– le point de coordonnées (*x* ; *y*), si *x* et *y* sont deux nombres ;
– tous les points ayant pour abscisses les éléments de *x* et pour ordonnées ceux de *y* de même indice, si *x* et *y* sont des listes ou des tableaux de nombres.

NOTE
Pour un nuage de points, on peut aussi utiliser `plt.scatter()`.

Exemple 1 : Nuage de points avec des listes x et y

- x est la liste des abscisses.
- y est la liste des ordonnées.
- Les points de coordonnées (0 ; 0,7), (1 ; 0,9), ..., (5 ; 0,6) sont marqués par un + rouge (`'r+'`).

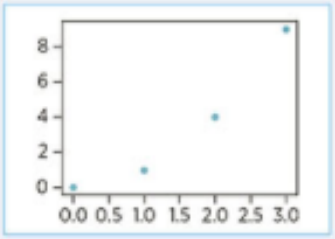
```
1  import matplotlib.pyplot as plt
2  x = [0, 1, 2, 3, 4, 5]
3  y = [0.7, 0.9, 0.4, 0.5, 0.8, 0.6]
4  plt.plot(x, y, 'r+')
5  plt.show()
```



Exemple 2 : Nuage de points avec une boucle

- x prend les valeurs 0, 1, 2 et 3.
- Les points de coordonnées (0 ; 0), (1 ; 1), (2 ; 4) et (3 ; 9) sont marqués par un rond bleu (`'bo'`).

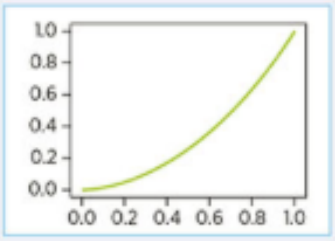
```
1  import matplotlib.pyplot as plt
2  for x in range(4):
3      plt.plot(x, x**2, 'bo')
4  plt.show()
```



Exemple 3 : Courbe avec numpy et un tableau

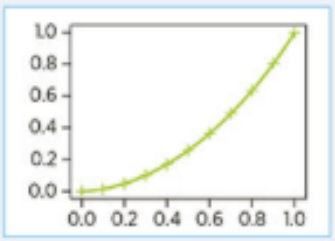
- On importe `numpy` pour créer le tableau x de 11 nombres de 0 à 1 avec un pas de 0,1.
- Les points de coordonnées (0 ; 0^2), (0,1 ; $0{,}1^2$), ..., (1 ; 1^2) sont reliés par un trait vert (`'g–'`).

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3  x = np.linspace(0, 1, 11)
4  plt.plot(x, x**2, 'g–')
5  plt.show()
```



On peut préciser davantage de paramètres, en utilisant des formulations plus explicites que les raccourcis bien pratiques comme `'r+'` ou `'bo'`.
On obtient ainsi le graphique ci-contre en remplaçant la ligne 4 de l'exemple 3 par :

```
4  plt.plot(x, x**2, 'g+–', markersize = 10)
```



ou encore, pour que la courbe soit en pointillé :

```
4  plt.plot(x, x**2, linestyle = ':', linewidth = 2, color = 'g',
   marker = '+', markersize = 10)
```

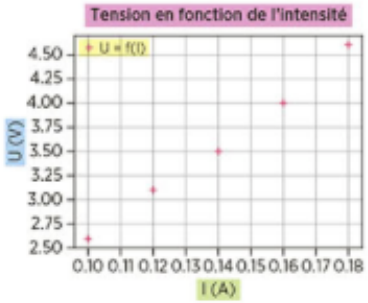
Paramètres graphiques :

- **`color`** = `'...'` : précise la couleur. Les choix de couleurs sont : `'b'` ou `'blue'` ; `'g'` ou `'green'` ; `'r'` ou `'red'` ; `'c'` ou `'cyan'` ; `'m'` ou `'magenta'` ; `'y'` ou `'yellow'` ; `'k'` ou `'black'` ; `'w'` ou `'white'`.
- **`marker`** = `'...'` : donne le style de la marque, par exemple, `'+'` (signe +), `'o'` (rond), `'x'` (croix), `'s'` (*square* : carré), `'v'` ou `'>'` ou `'<'` (triangles).
- **`markersize`** = *nombre* : précise la taille de la marque.
- **`linestyle`** = `'...'` : précise le style de trait, par exemple `'none'` (pas de trait), `'-'` ou `'solid'` (trait continu), `'--'` ou `'dashed'`, `':'` ou `'dotted'` (trait discontinu).
- **`linewidth`** = *nombre* : précise l'épaisseur du trait.

NOTE
`plt.plot(x, y, linestyle = 'none', color = 'black', marker = '+')`
a le même effet que :
`plt.plot(x, y, 'k+')`

B. Titres et légendes d'un graphique : label, xlabel, ylabel, legend

```
1   import numpy as np
2   import matplotlib.pyplot as plt
3
4   I = np.array([0.10, 0.12, 0.14, 0.16, 0.18])
5   U = np.array([2.6, 3.1, 3.5, 4.0, 4.6])
6   plt.plot(I, U, 'r+', label = 'U = f(I)')
7   plt.legend()  # affiche la légende entrée dans plot
8   plt.title("Tension en fonction de l'intensité")
9   plt.xlabel('I (A)')
10  plt.ylabel('U (V)')
11  plt.grid()    # affiche le quadrillage
12  plt.show()
```



Remarque : remplacer `plt.legend()` par `plt.legend(loc = 'best')` permet de placer la légende au meilleur endroit sur le graphique.

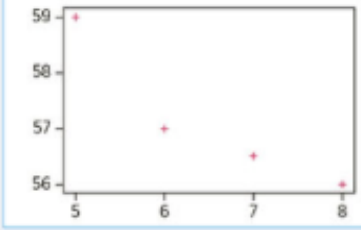
C. Réglage des axes : plt.axis

L'instruction `plt.axis([xmin, xmax, ymin, ymax])` permet de préciser les bornes des axes.

Exemple :

La figure ci-contre est obtenue avec le réglage des axes par l'instruction :
`plt.axis([5, 8, 56, 59])`

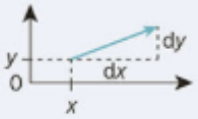
On aurait aussi pu régler :
– chaque axe séparément : `plt.xlim(5, 8)` et `plt.ylim(56, 59)` ;
– une seule des bornes, comme : `plt.xlim(xmax = 8)`
ou `plt.ylim(ymin = 59)`.



Annexe 3 : Savoir tracer des vecteurs

D. Représentation d'un vecteur : matplotlib.pyplot

Pour tracer le vecteur de coordonnées (dx ; dy) à partir du point de coordonnées (x ; y), on utilise l'instruction :

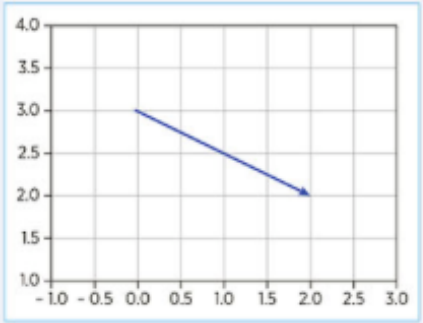


`plt.quiver(x, y, dx, dy, angles = 'xy', scale = 1, scale_units = 'xy')`

Ces réglages, à ne pas modifier, assurent la cohérence de la direction du vecteur tracé et de ses dimensions avec le repère et les unités sur les axes.

Exemple

```
1   import matplotlib.pyplot as plt
2   plt.quiver(0, 3, 2, -1, color = 'b', angles = 'xy',
    scale = 1, scale_units = 'xy')
3   plt.axis([-1, 3, 1, 4])
4   plt.grid()
5   plt.show()
```



Remarque : il est préférable d'utiliser `plt.quiver()` mais `plt.arrow()` peut aussi être utilisé, avec des réglages pour éviter des problèmes de positionnement et de dimensions par rapport aux axes.

`plt.arrow(x, y, dx*echelle, dy*echelle, head_width = 0.05, length_includes_head = True)`

Largeur de la pointe de la flèche

La pointe de la flèche est incluse dans les dimensions données.

<u>Annexe 4</u> : Reconnaître les instructions de base : plt, plot(), range(), append().

Lors de l'exécution d'un programme, un ordinateur doit stocker des informations en mémoire, puis les retrouver. On crée pour cela une **variable** que l'on peut imaginer comme une boîte avec un nom, que l'on choisit, et qui contient une « valeur ».
Si cette valeur est un nombre, on parle de **variable numérique**.

Instruction	En Python	Effet
Affecter 6 à la variable m	`m = 6`	Créer la variable m (si elle ne l'était pas déjà) et lui affecter 6
Afficher la valeur de m	`print(m)`	Afficher le contenu de m, c'est-à-dire 6
Affecter le double de m à la variable n	`n = m*2`	Créer la variable n (si nécessaire) et lui affecter le double de la valeur de m, c'est-à-dire 12

Attention ! L'affectation ne s'écrit que dans un sens. On n'écrira pas `6 = m`. Le signe d'égalité n'a pas la même signification qu'en mathématiques.

Exemple :
Ci-contre, l'instruction `m = m*10` a pour effet :
– de calculer le produit de la valeur de m, c'est-à-dire 5, par 10 ;
– d'affecter le résultat 50 à m (la valeur précédente 5 est effacée).

NOMMER UNE VARIABLE

Donner un nom évocateur à une variable dans un programme en facilite la compréhension. Ce nom doit débuter par une lettre et peut contenir des lettres, des chiffres ou le caractère « _ », mais ne peut pas être un mot réservé du langage Python comme `print`, `if`, `for`, etc.

AVEC DEUX VARIABLES

On peut les traiter séparément ou simultanément en Python :
- L'instruction `a,b = 2,3` affecte en même temps 2 à a et 3 à b.
- L'instruction `print(a, b)` fait afficher les deux valeurs de a et b séparées par une espace.

```
1  m = 5
2  m = m*10
3  print(m)
```

On obtient : `50`.