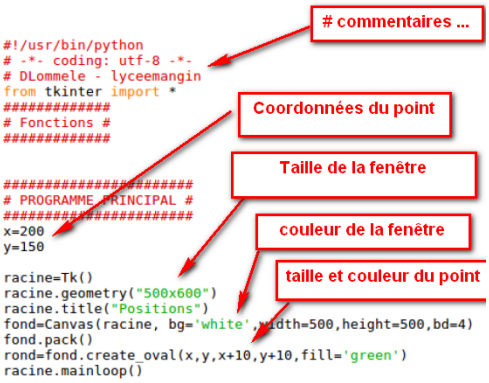
VECTEUR VITESSE D'UN POINT

I.But de la séance

Nous avons vu que décrire un mouvement consister, une fois le système et le référentiel précisés, à caractériser la trajectoire et à calculer la 'norme' de la vitesse. Il existe un outil mathématique permettant d'exprimer en même temps ces deux caractéristiques : **le vecteur** .

Comment peut-on construire un vecteur vitesse grâce au langage de programmation Python.

Vous pouvez le télécharger ici [http://www.lommele.com/activite_vitesse.zip]



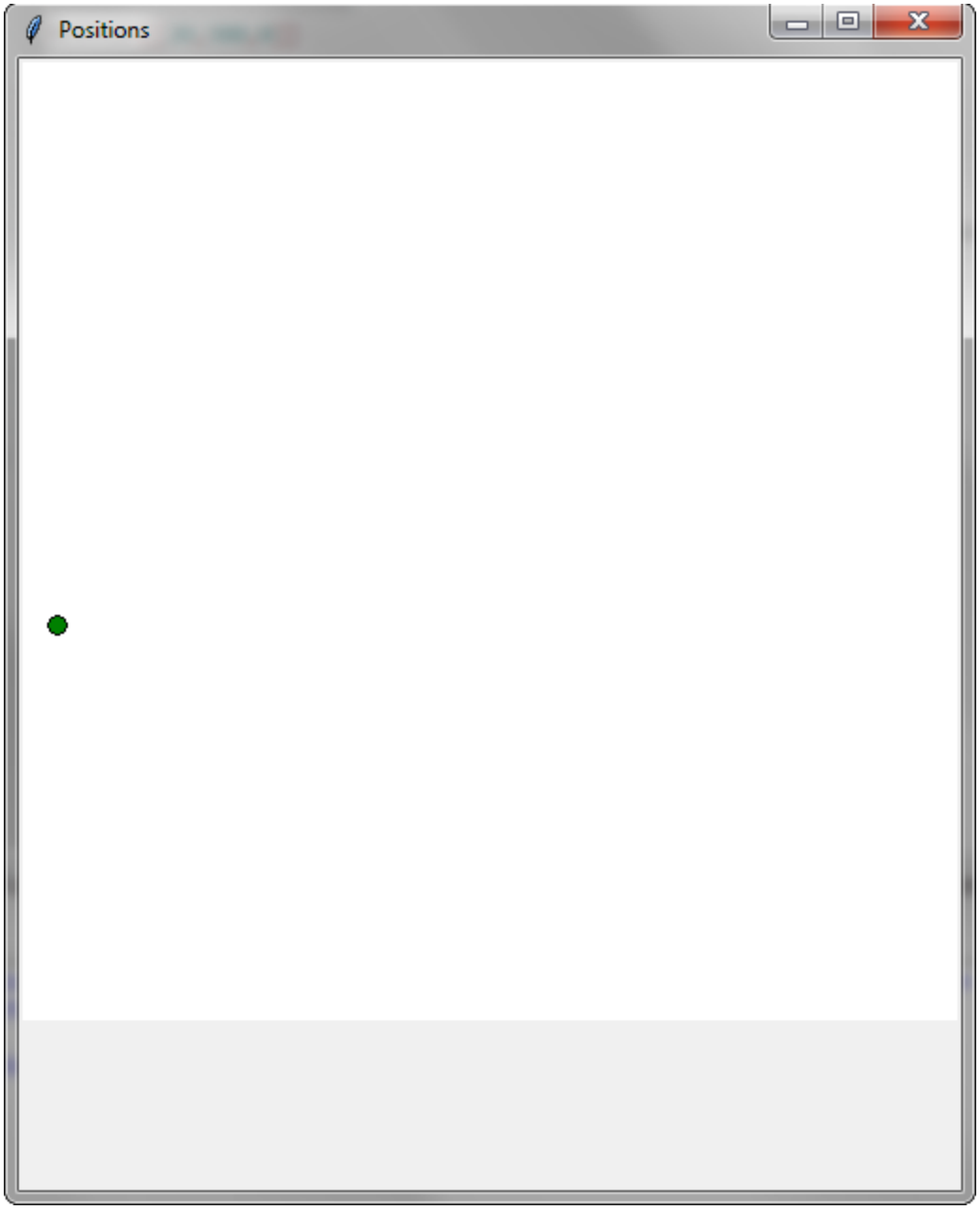

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# DLommele - lyceemangin
from tkinter import *
#############
# Fonctions #
#############


########################
# PROGRAMME PRINCIPAL #
########################
x=200
y=150

racine=Tk()
racine.geometry("500x600")
racine.title("Positions")
fond=Canvas(racine, bg='white',width=500,height=500,bd=4)
fond.pack()
rond=fond.create_oval(x,y,x+10,y+10,fill='green')
racine.mainloop()
```

II.Placer des points dans une fenêtre graphique

II.I.Plaçons un premier point.

Pour placer un point dans une fenêtre graphique, un code en langage python serait par exemple, le programme ci-contre.
Ce code (vitesse1.py) place un point de rayon 5 pixels, de couleur verte, à une position définie par le couple de coordonnées (200,150)
⇒ Comme dans le TP précédent, amusez-vous à modifier la position du point, la taille du point et la couleur de celui-ci.

On execute le programme en changeant la position du point : x = 20 , y = 300

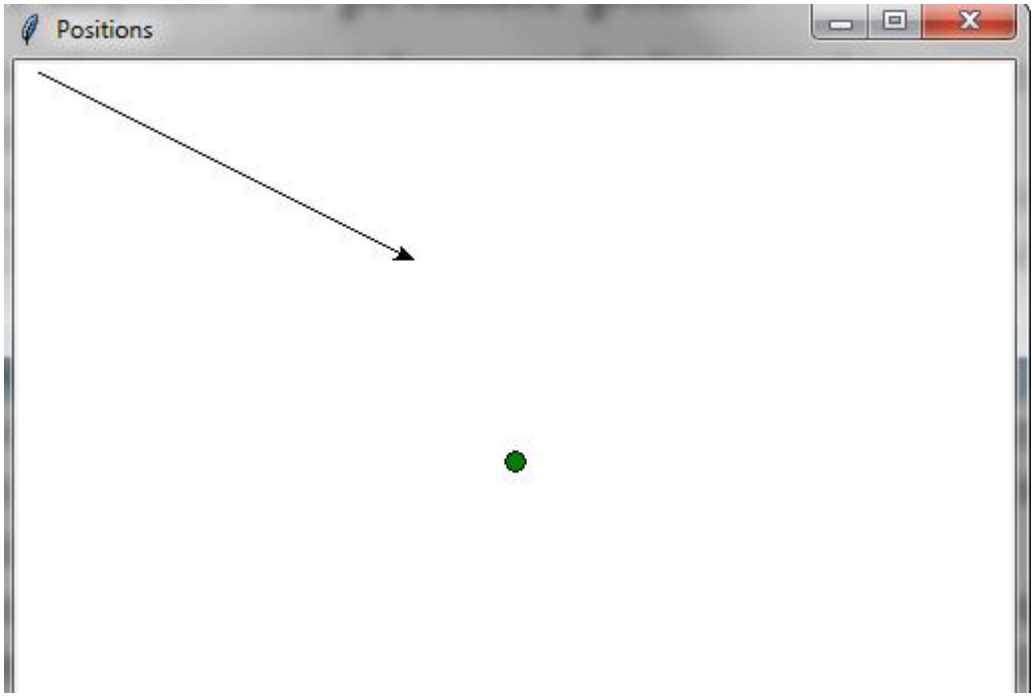


II.II.Le vecteur position

Le vecteur position prend comme point de départ l'origine du repère et comme point d'arrivée le point que nous venons de placer. (Attention, dans notre programme, l'origine du repèr est en haut à gauche)
⇒ Le programme vitesse3.py ne dessine pas correctement le vecteur position. Modifiez le programme pour avoir le résultat voulu à savoir : [vous pourrez consulter la correction vitesse3_corr.py après avoir essayé]

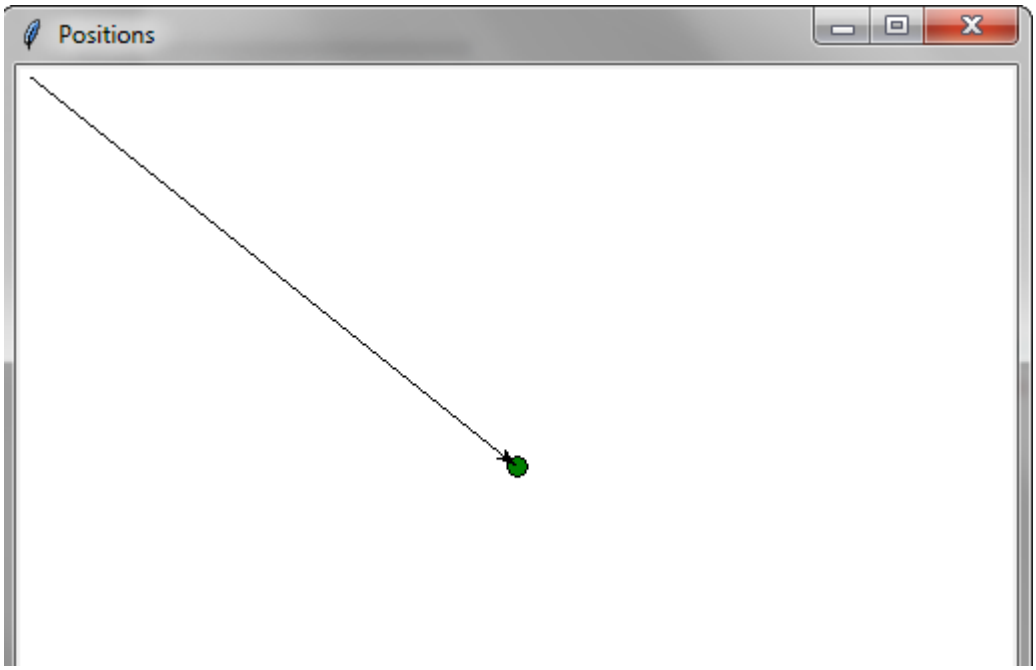
On exécute le programme 3 :

```
Points=[[250,200,0]]

racine=Tk()
racine.geometry("500x600")
racine.title("Positions")
fond=Canvas(racine, bg='white',width=500,height=500,bd=4)
fond.pack()
rond=fond.create_oval(Points[0][0]-5,Points[0][1]-5,Points[0][0]+5,Points[0][1]+5,fill='green')
vecteurPosition=fond.create_line(0,0,200,100, fill='black',arrow='last')
racine.mainloop()
```



On modifie le programme 3 :

```
Points=[[250,200,0]]
racine=Tk()
racine.geometry("500x600")
racine.title("Positions")
fond=Canvas(racine, bg='white',width=500,height=500,bd=4)
fond.pack()
rond=fond.create_oval(Points[0][0]-5,Points[0][1]-5,Points[0][0]+5,Points[0][1]+5,fill='green')
vecteurPosition=fond.create_line(0,0,Points[0][0],Points[0][1], fill='black',arrow='last')
racine.mainloop()
```



II.III.Plaçons deux points

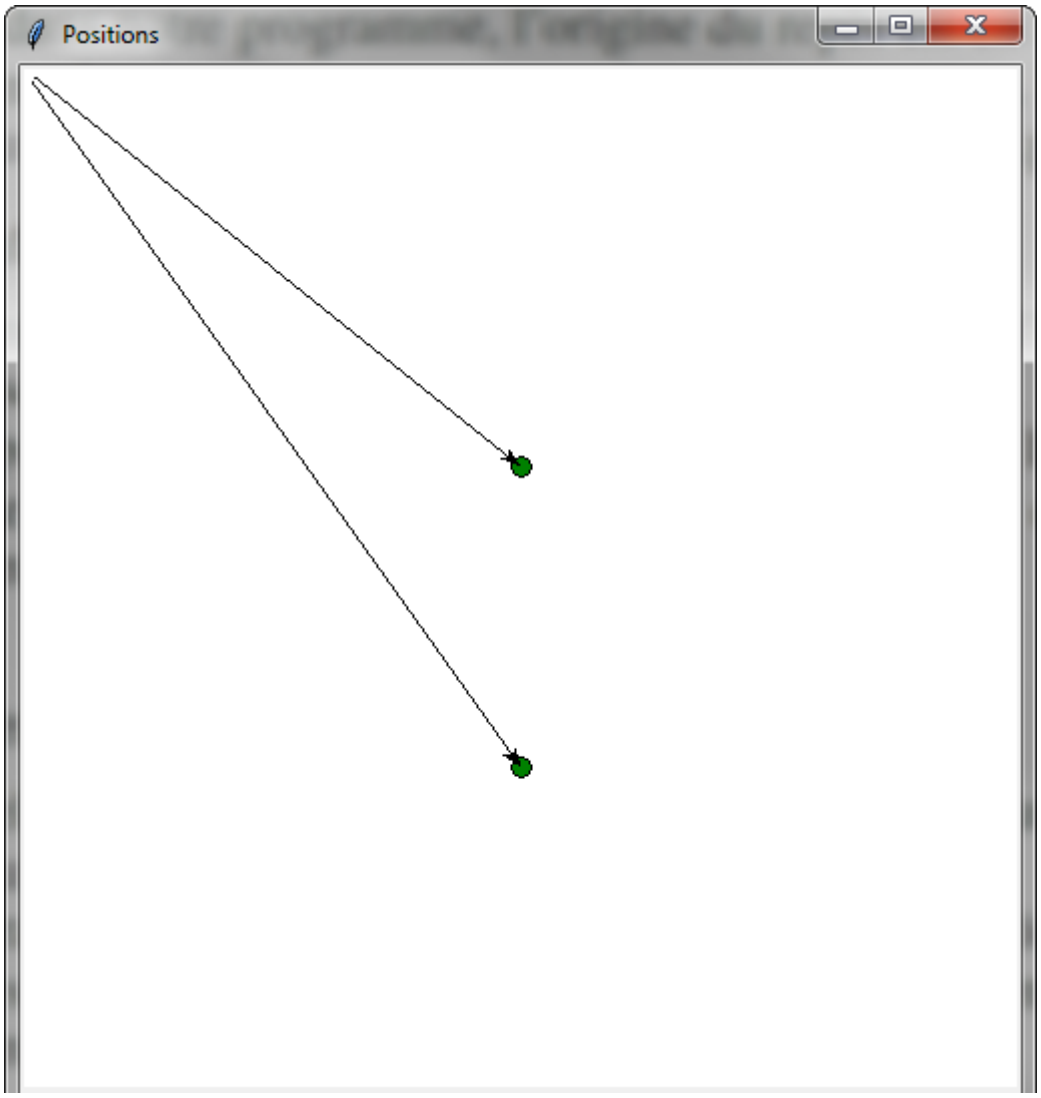
Rajoutons un second point dans la liste. On a maintenant **Points =** [[250,200,0],[250,350,0.2]]
⇒Quelle est la position de ce second point ?
⇒Quel est l'intervalle de temps nécessaire au passage de la position du point 1 à celle du point 2 ?
⇒Complétez le programme vitesse4.py pour faire apparaître les deux points, comme ci-contre. [vous pourrez consulter la correction vitesse4_corr.py après avoir essayé].

On exécute le programme 4 :

```
Points=[[250,200,0],[250,350,0,2]]

racine=Tk()
racine.geometry("500x600")
racine.title("Positions")
fond=Canvas(racine, bg='white',width=500,height=500,bd=4)
fond.pack()
rond=fond.create_oval(Points[0][0]-5,Points[0][1]-5,Points[0][0]+5,Points[0][1]+5,fill='green')
vecteurPosition1=fond.create_line(0,0,Points[0][0],Points[0][1], fill='black',arrow='last')
# rond=fond.create_oval(a completer,fill='green')
# vecteurPosition2=fond.create_line(a completer)
racine.mainloop()
```

On modifie le programme 4 et on l'exécute :

```
Points=[[250,200,0],[250,350,0,2]]

racine=Tk()
racine.geometry("500x600")
racine.title("Positions")
fond=Canvas(racine, bg='white',width=500,height=500,bd=4)
fond.pack()
rond=fond.create_oval(Points[0][0]-5,Points[0][1]-5,Points[0][0]+5,Points[0][1]+5,fill='green')
vecteurPosition=fond.create_line(0,0,Points[0][0],Points[0][1], fill='black',arrow='last')
rond=fond.create_oval(Points[1][0]-5,Points[1][1]-5,Points[1][0]+5,Points[1][1]+5,fill='green')
vecteurPosition2=fond.create_line(0,0,Points[1][0],Points[1][1], fill='black',arrow='last')
racine.mainloop()
```



II.IV.Vecteur déplacements

Le vecteur déplacement est celui qui représente le passage d'une position à une autre. En mathématiques, il correspond à la différence des deux vecteurs positions.
Pour le tracer dans notre programme il suffit de tracer le vecteur entre la position 1 et la position 2.
⇒Modifiez le programme vitesse5.py pour faire apparaître ce vecteur déplacement en rouge.
[vous pourrez consulter la correction vitesse5_corr.py après avoir essayé].

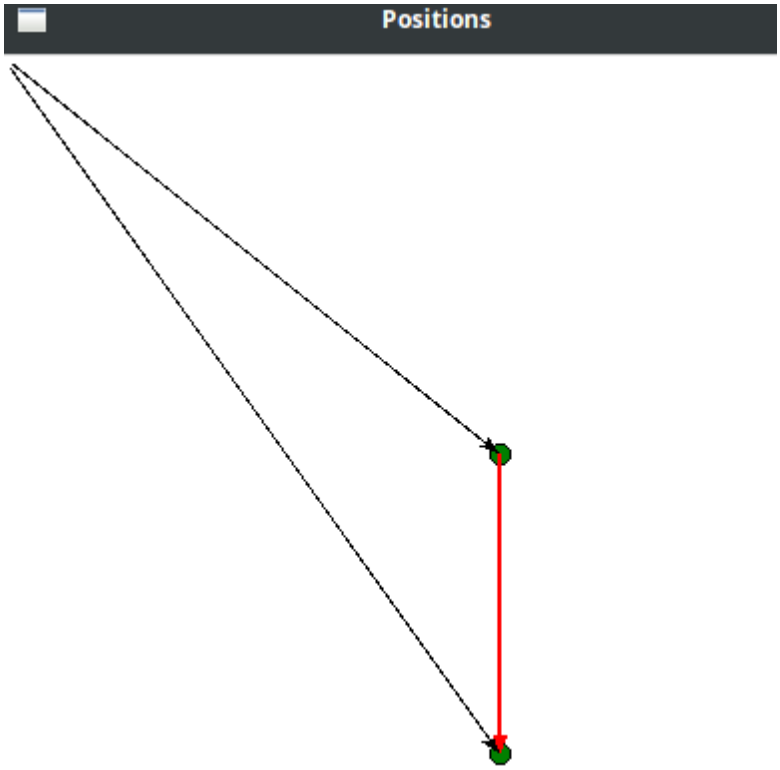
On modifie le programme 5 (ligne verte à compléter) :

```
Points=[[250,200,0],[250,350,0,2]]

racine=Tk()
racine.geometry("500x600")
racine.title("Positions")
fond=Canvas(racine, bg='white',width=500,height=500,bd=4)
fond.pack()
rond=fond.create_oval(Points[0][0]-5,Points[0][1]-5,Points[0][0]+5,Points[0][1]+5,fill='green')
vecteurPosition1=fond.create_line(0,0,Points[0][0],Points[0][1], fill='black',arrow='last')
rond=fond.create_oval(Points[1][0]-5,Points[1][1]-5,Points[1][0]+5,Points[1][1]+5,fill='green')
vecteurPosition2=fond.create_line(0,0,Points[1][0],Points[1][1], fill='black',arrow='last')

#vecteurDeplacement=fond.create_line(a completer)
racine.mainloop()
```

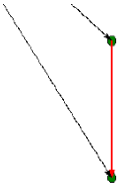
On modifie le programme 5 (modification en rouge) :

```
Points=[[250,200,0],[250,350,0,2]]

racine=Tk()
racine.geometry("500x600")
racine.title("Positions")
fond=Canvas(racine, bg='white',width=500,height=500,bd=4)
fond.pack()
rond=fond.create_oval(Points[0][0]-5,Points[0][1]-5,Points[0][0]+5,Points[0][1]+5,fill='green')
vecteurPosition1=fond.create_line(0,0,Points[0][0],Points[0][1], fill='black',arrow='last')
rond=fond.create_oval(Points[1][0]-5,Points[1][1]-5,Points[1][0]+5,Points[1][1]+5,fill='green')
vecteurPosition2=fond.create_line(0,0,Points[1][0],Points[1][1], fill='black',arrow='last')

vecteurDeplacement=fond.create_line(Points[0][0],Points[0][1],Points[1][0],Points[1][1],
fill='red',arrow='last',width=2)
racine.mainloop()
```

On exécute le programme 5 modifié : on fait apparaitre le vecteur déplacement :



II.V.Plaçons un grand nombre de points



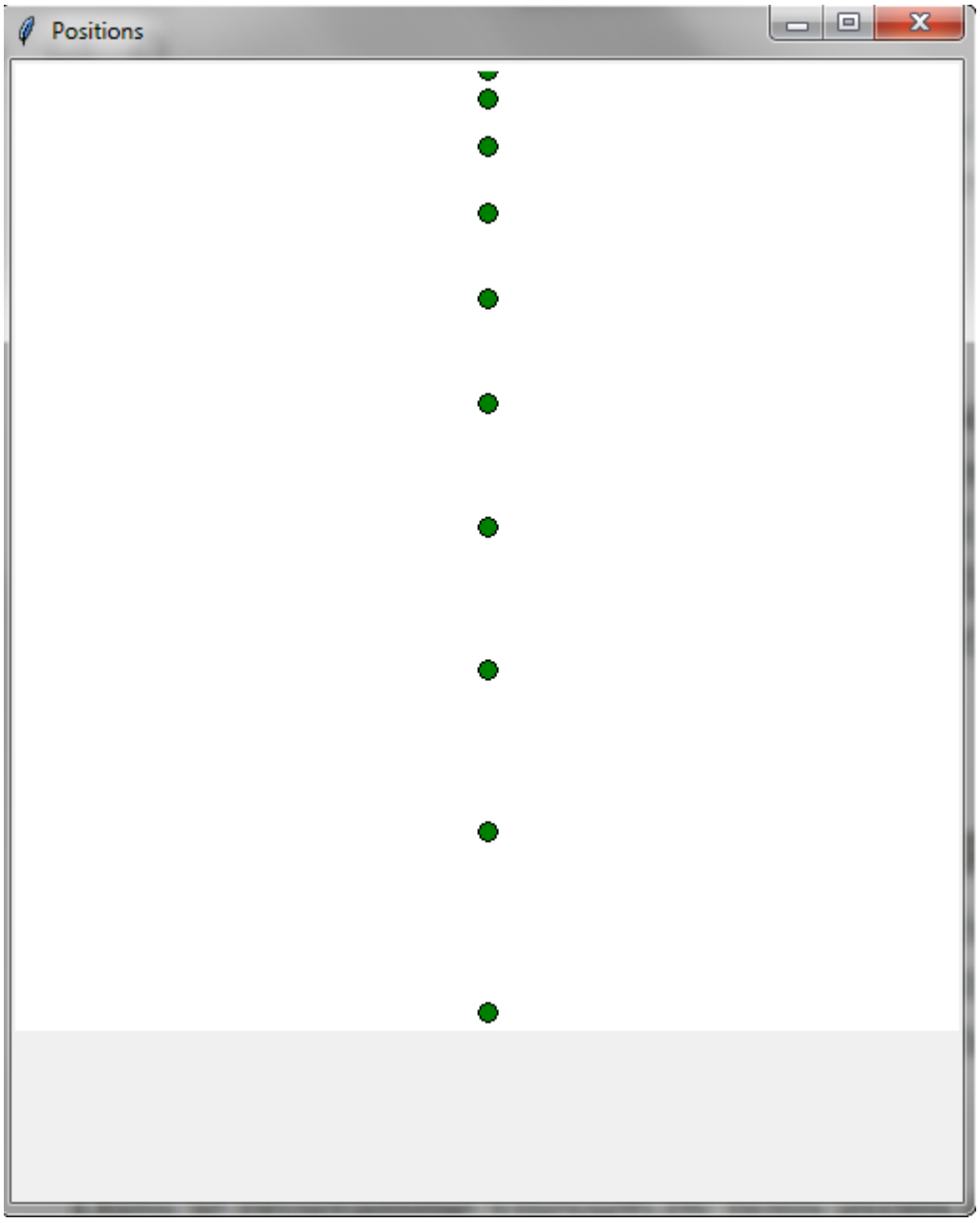
Vous avez constaté, dans les exemples précédents, qu'il suffisait de changer d'indice pour passer d'un point à un autre. Points[0] pour le premier point, Points[1] pour le second… et on généralise Points[2] pour le troisième… etc. Dans le langage de programmation Python, on peut utiliser une boucle pour changer cet indice sans réécrire une ligne par point.

Dans le programme vitesse6.py nous avons écrit la liste de position suivante :
Points=[[250, 0.0, 0], [250, 5.0, 1], [250, 20.0, 2], [250, 45.0, 3], [250, 80.0, 4], [250, 125.0, 5], [250, 180.0, 6], [250, 245.0, 7], [250, 320.0, 8], [250, 405.0, 9], [250, 500.0, 10]]

On exécute le programme 6 :

```
Points=[[250, 0.0, 0], [250, 5.0, 1], [250, 20.0, 2], [250, 45.0, 3], [250, 80.0, 4], [250, 125.0, 5], [250, 180.0, 6], [250, 245.0, 7], [250, 320.0, 8], [250, 405.0, 9], [250, 500.0, 10]]

racine=Tk()
racine.geometry("500x600")
racine.title("Positions")
fond=Canvas(racine, bg='white',width=500,height=500,bd=4)
fond.pack()
for i in range(len(Points)):
   fond.create_oval(Points[i][0]-5,Points[i][1]-5,Points[i][0]+5,Points[i][1]+5,fill='green')
racine.mainloop()
```

Résultat du programme 6 :



Position du 1er point : x = 250 , y = 0
Position du dernier point : x = 250, y = 500

III.I.Première étape : calculer la distance entre un point et son successeur

On rappelle que :
Points=[[250, 0.0, 0], [250, 5.0, 1], [250, 20.0, 2], [250, 45.0, 3], [250, 80.0, 4], [250, 125.0, 5], [250, 180.0, 6], [250, 245.0, 7], [250, 320.0, 8], [250, 405.0, 9], [250, 500.0, 10]]
et on fait l'hypothèse que les coordonnées sont données en mètres.

⇒Quelle est la distance entre les points 7 et 8 ? M_7M_8 =

Nous allons créer une fonction **Distance(i)** calculant la distance entre le i^{EME} point de la liste et le suivant.
⇒ Rappelez la formule donnant le distance entre deux points $M_1(x_1,y_1)$ et $M_2(x_2,y_2)$ vue en cours de mathématiques.
⇒ Ouvrez le programme vitesse7.py Comment la formule donnant la distance entre deux points est-elle écrite dans le programme ?
⇒ Exécutez le programme vitesse7.py. Dans la console, demandez la distance M_7M_8 et vérifiez la valeur trouvée précédemment.
Exemple pour la distance M_4M_5

```
>>> Distance(4)
La distance entre le point  4  et le suivant est :  45.0  m
```

On complète le programme 7 :

```
def Distance(i):#donne la distance entre un point d'indice i et son successeur
    x1=Points[i][0]
    y1=Points[i][1]
    x2=Points[i+1][0]
    y2=Points[i+1][1]
    d=sqrt((x2-x1)**2+(y2-y1)**2)
    print("La distance entre le point ",i," et le suivant est : ",d," m")

########################
# PROGRAMME PRINCIPAL #
########################
Points=[[250, 0.0, 0], [250, 5.0, 1], [250, 20.0, 2], [250, 45.0, 3], [250, 80.0, 4], [250, 125.0, 5], [250, 180.0, 6], [250, 245.0, 7], [250, 320.0, 8], [250, 405.0, 9], [250, 500.0, 10]]
Distance(7)
```

On execute le programme 7 :

La distance entre le point 7 et le suivant est : 75.0 m

III.II.Deuxième étape : calculer la vitesse entre un point et son successeur

Pour déterminer cette vitesse, il faut diviser la distance précédente par l'intervalle de temps nécessaire pour la parcourir.
⇒ Quel intervalle de temps s'est-il écoulé entre ces deux positions ? Δt =
⇒ Calculez v_7= M_7M_8 / Δt
Nous allons créer une fonction **Vitesse(i)** calculant la vitesse instantanée du système au i^{EME} point.
⇒ Exécutez le programme vitesse8.py. Dans la console, demandez les valeurs des vitesses v_4, v_5, v_6,v_7 et v_8 selon l'exemple ci-dessous.

```
>>> Vitesse(4)
La distance entre le point  4  et le suivant est :  45.0  m
La vitesse instantanée au point 4 vaut  45.0 m/s
```

⇒ A la lecture de ces valeurs, comment pouvez-vous décrire le mouvement du système ?

On complète le programme 8 :

```
from tkinter import *
from math import sqrt
##############
# Fonctions #
##############
def Distance(i):#donne la distance entre un point d'indice i et son successeur
    x1=Points[i][0]
    y1=Points[i][1]
    x2=Points[i+1][0]
    y2=Points[i+1][1]
    d=sqrt((x2-x1)**2+(y2-y1)**2)
    print("La distance entre le point ",i," et le suivant est : ",d," m")
```

```
    return d

def Vitesse(i):
    DeltaT=1 #il s'écoule une seconde entre chaque position
    v=Distance(i)/DeltaT
    print("La vitesse instantanée au point", i, "vaut ",v, "m/s")


########################
# PROGRAMME PRINCIPAL #
########################
Points=[[250, 0.0, 0], [250, 5.0, 1], [250, 20.0, 2], [250, 45.0, 3], [250, 80.0, 4], [250, 125.0, 5], [250, 180.0, 6], [250, 245.0, 7], [250, 320.0, 8], [250, 405.0, 9], [250, 500.0, 10]]

Vitesse(4)
Vitesse(5)
Vitesse(6)
Vitesse(7)
```

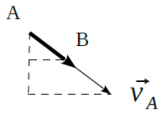
On execute le programme 8 : Le résultat (affiché en bleu) est dans la console :

```
La distance entre le point  4  et le suivant est :  45.0  m
La vitesse instantanée au point 4 vaut  45.0 m/s
La distance entre le point  5  et le suivant est :  55.0  m
La vitesse instantanée au point 5 vaut  55.0 m/s
La distance entre le point  6  et le suivant est :  65.0  m
La vitesse instantanée au point 6 vaut  65.0 m/s
La distance entre le point  7  et le suivant est :  75.0  m
La vitesse instantanée au point 7 vaut  75.0 m/s
```

Le mouvement est un mouvement rectiligne accéléré.

III.III.Troisième étape: représenter le vecteur vitesse

Le **vecteur vitesse** au point A est colinéaire au **vecteur déplacement** entre les points A et B.
Pour tracer le vecteur nous allons donc utiliser ces deux points.

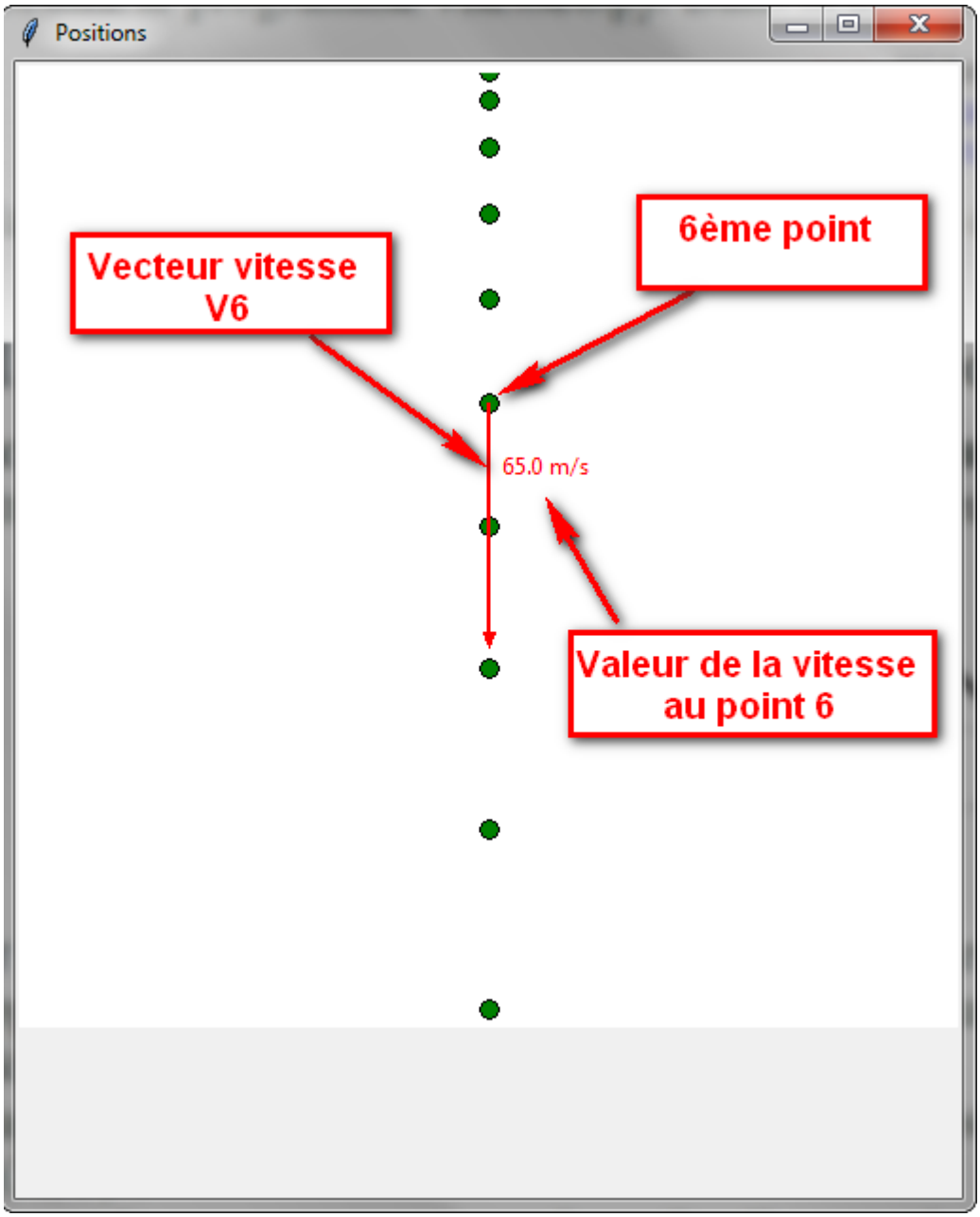


⇒ Sur le schéma ci-dessus, situez le **vecteur vitesse** et le **vecteur déplacement**
On appelle k le facteur de colinéarité entre les vecteurs déplacement et vitesse. (par exemple si k=2, le vecteur vitesse est 2 fois plus grand que le vecteur déplacement).
⇒ D'après le théorème de Thalès, quelle relation existe-t-il entre les coordonnées des points A (x_A,y_A) et B(x_B,y_B) et celles de l'extrémité du vecteur vitesse (X,Y) ?
⇒ Ouvrez le programme vitesse9.py. Quelles sont les deux lignes calculant ces coordonnées ?
⇒ Exécutez le programme 9 et observez le résultat.
⇒ Modifiez le programme 9 pour qu'il dessine le vecteur vitesse du sixième point.

On ouvre le programme 9 :

```
##############
# Fonctions #
##############
def Distance(i):#donne la distance entre un point d'indice i et son successeur
    x1=Points[i][0]
    y1=Points[i][1]
    x2=Points[i+1][0]
    y2=Points[i+1][1]
    d=sqrt((x2-x1)**2+(y2-y1)**2)
    print("La distance entre le point ",i," et le suivant est : ",d," m")
    return d

def Vitesse(i):
    DeltaT=1 #il s'écoule une seconde entre chaque position
    v=Distance(i)/DeltaT
    print("La vitesse instantanée au point", i, "vaut ",v, "m/s")
    return v

def Vecteur_Vitesse(i,k): #trace le iEME vecteur vitesse avec un facteur k
    xA=Points[i][0]
    yA=Points[i][1]
    xB=Points[i+1][0]
    yB=Points[i+1][1]
    X=(xB-xA)*k+xA
    Y=(yB-yA)*k+yA
    return (X,Y)

########################
# PROGRAMME PRINCIPAL #
########################
Points=[[250, 0.0, 0], [250, 5.0, 1], [250, 20.0, 2], [250, 45.0, 3], [250, 80.0, 4], [250, 125.0, 5], [250, 180.0, 6], [250,
245.0, 7], [250, 320.0, 8], [250, 405.0, 9], [250, 500.0, 10]]
n=6 #le vecteur dont on veut dessiner la vitesse
racine=Tk()
racine.geometry("500x600")
racine.title("Positions")
fond=Canvas(racine, bg='white',width=500,height=500,bd=4)
fond.pack()
for i in range(len(Points)):
    fond.create_oval(Points[i][0]-5,Points[i][1]-5,Points[i][0]+5,Points[i][1]+5,fill='green')

#on trace le vecteur vitesse
vecteurVitesse=fond.create_line(Points[n][0],Points[n][1],Vecteur_Vitesse(n,2)[0],Vecteur_Vitesse(n,2)[1],
fill='red',arrow='last',width=2)

#bonus : on affiche sa valeur
fond.create_text((Points[n][0]+Points[n+1][0])*0.5+30,(Points[n][1]+Points[n+1][1])*0.5,fill='red',text=(str(Vitesse
(n))+" m/s"))
racine.mainloop()
```

On exécute le programme 9 :



ON A TRACÉ UN VECTEUR VITESSE A L'AIDE D'UN LANGUAGE DE PROGRAMMATION !!!