TP5_VecteurSecondeEleve

February 7, 2021

1 TP 5 : Distance entre deux points, lunes et tâches des girafes

Document issu d'une proposition de TP de l'académie de Strasbourg.

1°) Un algorithme de calcul de distance * Dans un repère orthonormé, on donne les points $A(-2;3)$ et $B(5;7)$. Calculer la valeur de la distance AB.

Réponse :

- Compléter le programme ci-dessous pour qu'il calcule la distance entre deux points dans un repère orthonormé puis exécuter-le avec les coordonnées des points de la question.

[ ]:
```
from math import *

def distance(xA,yA,xB,yB):
    d=sqrt((xB-xA)**2+ ............)   #à compléter
    return d

print(distance(.................))  #à compléter
```

- Quel est le résultat affiché ? Que peut-on dire de ce résultat ?

Réponse :

2°) Un autre algorithme de calcul de distance * On conserve dans cette partie les points $A(-2;3)$ et $B(5;7)$. Calculer les coordonnées du vecteur $\overrightarrow{AB}$.

Réponse :

- Compléter le programme ci-dessous pour qu'il calcule les coordonnées d'un vecteur dans un repère orthonormé puis exécuter-le avec les coordonnées des points de la question.

[ ]:
```
from math import *

def CoordVect(xA,yA,xB,yB):
    X=...........
    Y=...........  #à compléter
    return X,Y

#print (CoordVect(.......................))   #à compléter
```

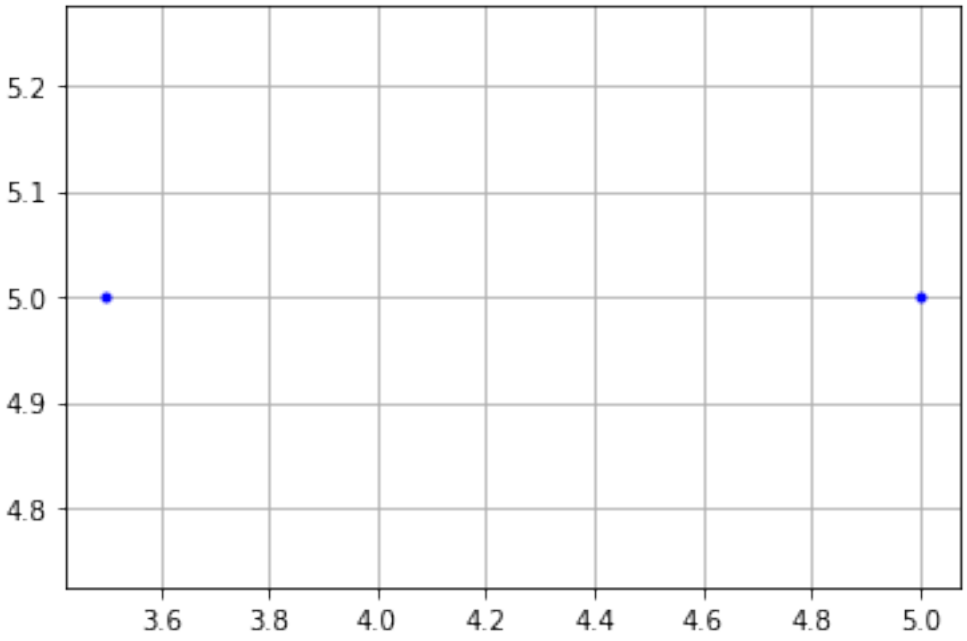
- Compléter le programme suivant pour qu'il calcule la distance entre deux points à partir des coordonnées du vecteur formé par ces deux points.

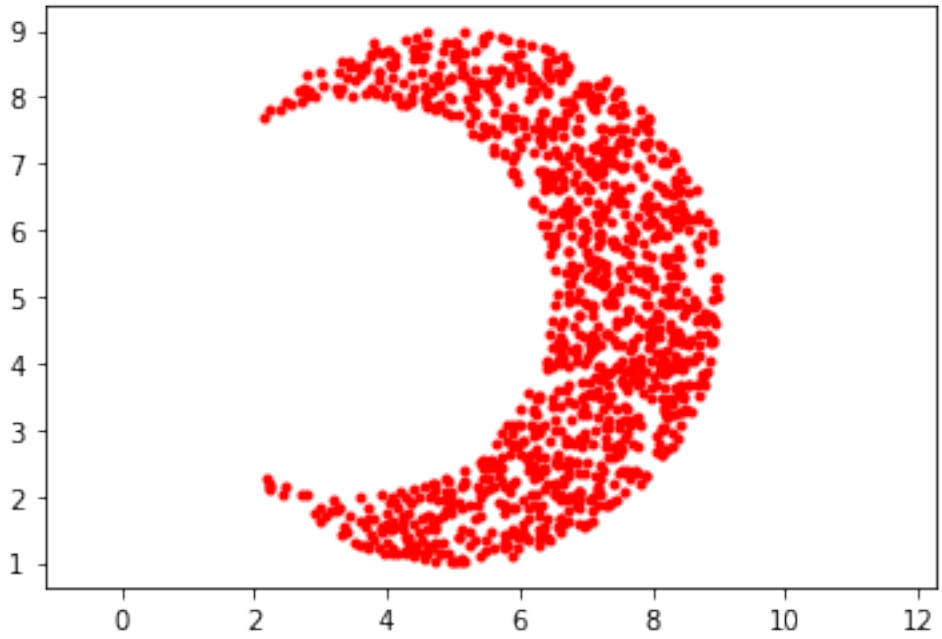
[7]:
```
from math import *

def DistVect(xA,yA,xB,yB):
    X,Y=..... #à compléter
    d=..... #à compléter
    return d

print (DistVect(-2,3,5,7))
```

```
8.06225774829855
```

3°) Des lunes * Exécuter le programme ci-dessous.

[1]:
```
from math import *
from random import *
import matplotlib.pyplot as plt


def distance(xA,yA,xB,yB):
    d=sqrt((xB-xA)**2+ (yB-yA)**2)
    return d

xA=3.5
yA=5
xB=5
yB=5

plt.plot([xA,xB],[yA,yB],'b.')
plt.grid()
plt.show()
plt.axis("equal")

for i in range(1,5000):
    x1=random()*10
    y1=random()*10
    dA=distance(x1,y1,xA,yA)
    dB=distance(x1,y1,xB,yB)
    if dA>3 and dB<4:
        plt.plot(x1,y1,'r.')
```



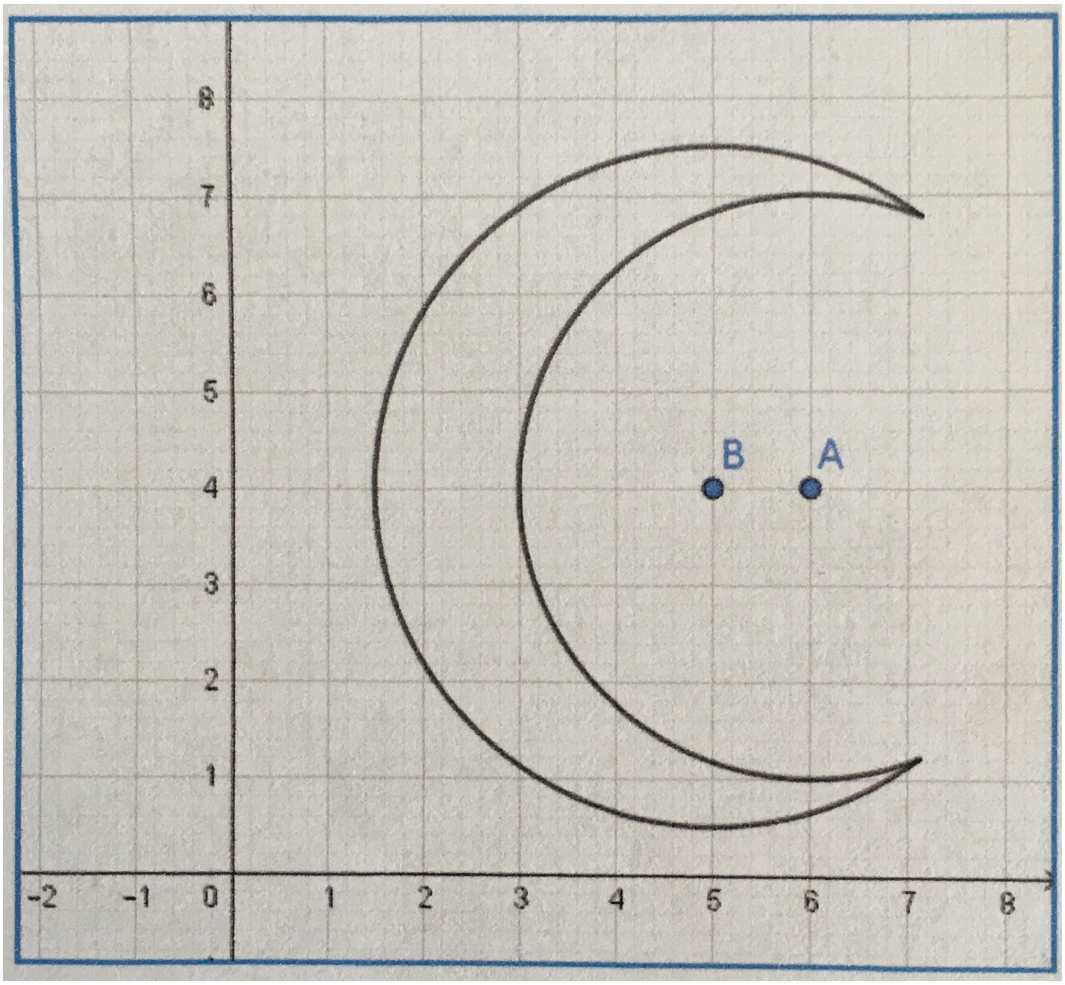


- Justifier le résultat obtenu

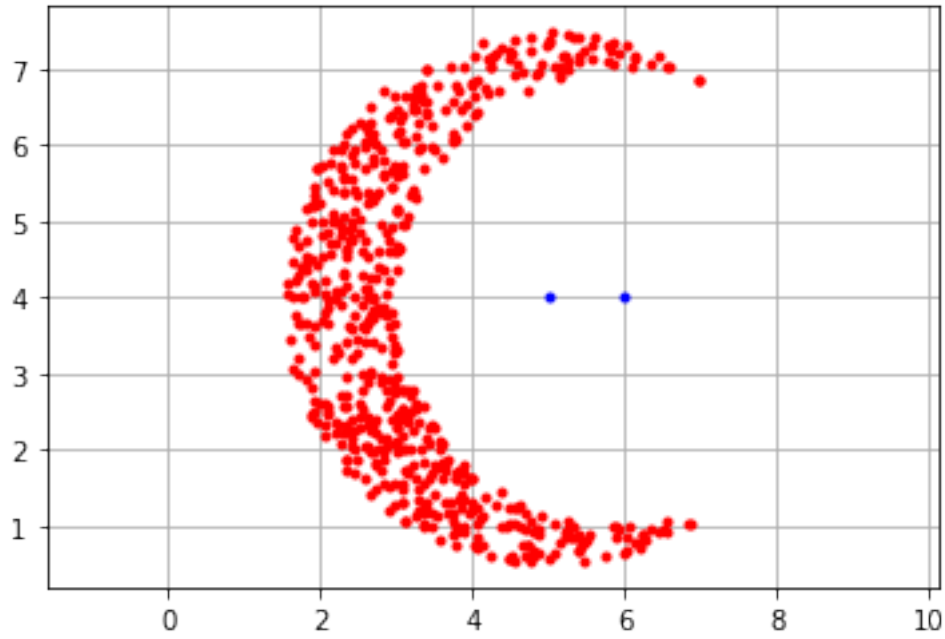
Réponse :

- Modifier le résultat pour obtenir un ensemble de points colorés dans la lune de l'image ci-

dessous.



[3]:
```
from math import *
from random import *
import matplotlib.pyplot as plt

def distance(xA,yA,xB,yB):
    d=sqrt((xB-xA)**2+ (yB-yA)**2)
    return d

xA=6
yA=4
xB=5
yB=4


à Compléter...
```

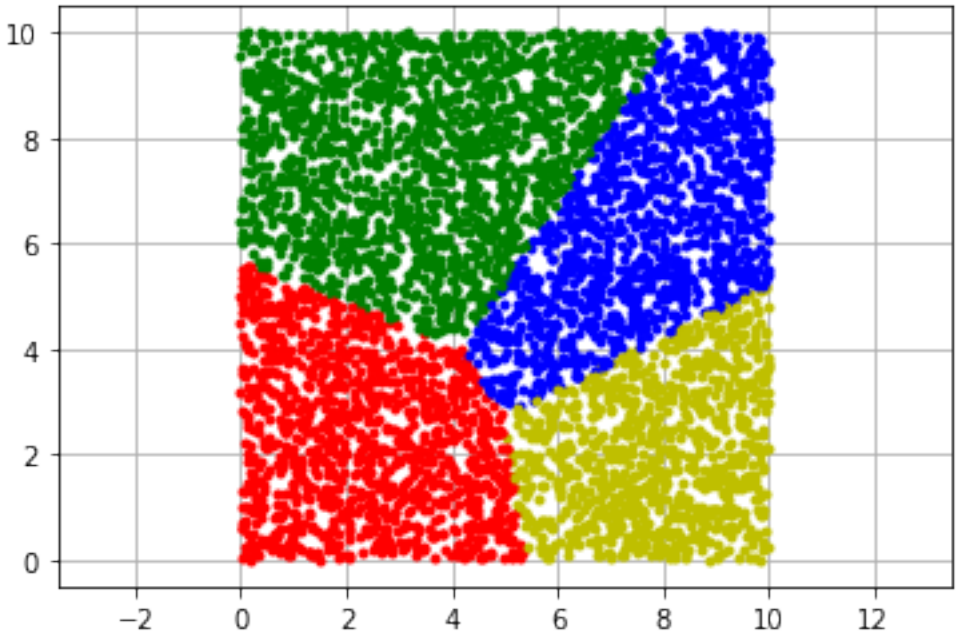


4°) À vous de programmer Dans un repère orthonormé, on donne les points: A(2;2), B(8;3), C(4;7) et D(7;5). On considère les points dont chaque coordonnée est comprise entre 0 et 10. (10 exclu)

- Créer un programme qui fait apparaître en couleur des points plus proches de A que de B, C et D.
- Faire le même travail avec trois autres couleurs pour des points plus proches de B, puis de C puis de D.

[6]:
```
from math import *
from random import *
import matplotlib.pyplot as plt

def distance(xA,yA,xB,yB):
    d=sqrt((xB-xA)**2+ (yB-yA)**2)
    return d

xA=2
yA=2
xB=8
yB=3
xC=4
yC=7
xD=7
yD=5
```

```
à compléter
```



Cela s'appelle le diagramme de Voronoï et constitue un modèle de répartition de la pigmentation du pelage des girages.