

Activités élaborées par le groupe PLP de l'IRES de Montpellier
2023-2025.

TP Python-Géogebra 1 : un peu de dessin

Le but de ce TP est de se familiariser à la programmation en utilisant l'outil Python-Geogebra, disponible en ligne sur le site

<https://geogebra.org/python/index.html>

- 1) **Commencer** par recopier les lignes suivantes dans l'outil.
Ces lignes permettent d'importer les bibliothèques python dont nous aurons besoin.

```
from math import *  
import time
```

- 2) On place un point aux coordonnées (x,y) grâce au code suivant.

```
Point(x,y)
```

Ajouter un point de coordonnées (1,1) puis un autre de coordonnées (0,1).

- 3) Nous aimerions maintenant tracer le segment entre ces deux points.
Pour cela, **modifier** le code précédent pour nommer A le point de coordonnées (1,1) et B le point aux coordonnées (0,1).

Tracer ensuite le segment [A;B] à l'aide de la commande suivante.

```
Segment(A,B)
```

- 4) **Compléter** votre dessin pour tracer le triangle ABC, où C est le point de coordonnées (0,0).

Remarque : Pour ne plus voir un point bleu à l'emplacement des points, on pourra choisir l'option « **is_visible=False** » lors de la création des points, avec la commande suivante.

```
Point(x,y, is_visible=False)
```

- 5) La distance entre deux points A et B peut être mesurée grâce à la fonction suivante :

```
AB=Distance(A,B)
```

Si l'on veut voir afficher le résultat du calcul, on utilisera la fonction “**print**”.

```
print("AB=", AB))
```

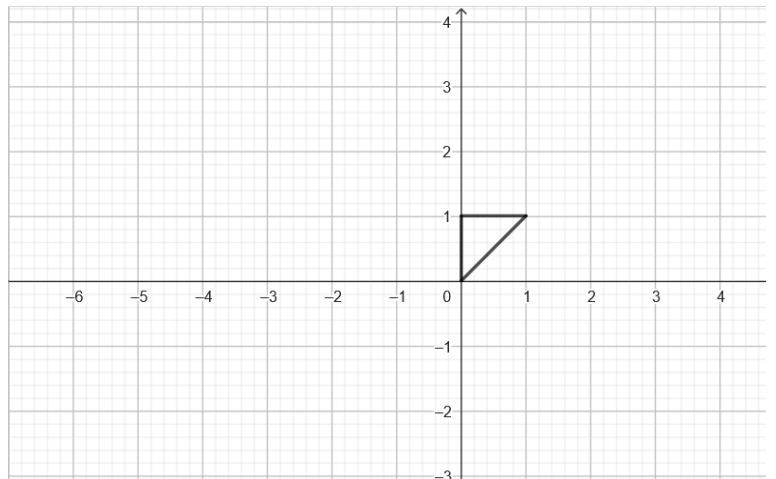
- 5.1) **Compléter** le programme afin qu'il permette d'afficher la longueur de chacun des 3 côtés du triangle

- 5.2) **Calculer** le périmètre du triangle ABC en utilisant la commande suivante :

```
print(("Le périmètre du triangle ABC est :", AB+ AC+ BC))
```

- 5.3) En utilisant ce qui précède, **vérifier** le théorème de Pythagore dans le triangle ABC.

```
1 from math import*
2 import time
3 A=Point(1,1,is_visible=False)
4 B=Point(0,1,is_visible=False)
5 C=Point(0,0,is_visible=False)
6
7 Segment(A,B)
8 Segment(A,C)
9 Segment(B,C)
10
11 AB=Distance(A,B)
12 AC=Distance(A,C)
13 BC=Distance(B,C)
14
15 print("AB=",AB)
16 print("AC=",AC)
17 print("BC=",BC)
18 print("le périmètre du triangle ABC est ",AB+AC+BC)
19
20 print("AB²+BC²=",AB**2+BC**2)
21 print("AC²=",AC**2)
```



```
AB= 1.0
AC= 1.414213562373095
BC= 1.0
le périmètre du triangle ABC est  3.414213562373095
AB²+BC²= 2.0
AC²= 2.0
```

On constate bien que : $AB^2+BC^2=AC^2$. Le théorème de Pythagore est bien vérifié dans le triangle

```
print("BC=",BC)
print("le périmètre du triangle ABC est ",AB+AC+BC)

print("AB2+BC2=",AB**2+BC**2)
print("AC2=",AC**2)
```

- 6) Nous aimerions maintenant créer une fonction « triangle » qui prend en entrée une longueur x et trace le triangle formé des points (0,0), (0,x) et (x,x).

Pour créer une fonction en python, il faut commencer comme suit :

```
def triangle(x):
    ...
```

Où ... représente le reste du code qui devra être indenté par rapport à l'en-tête.

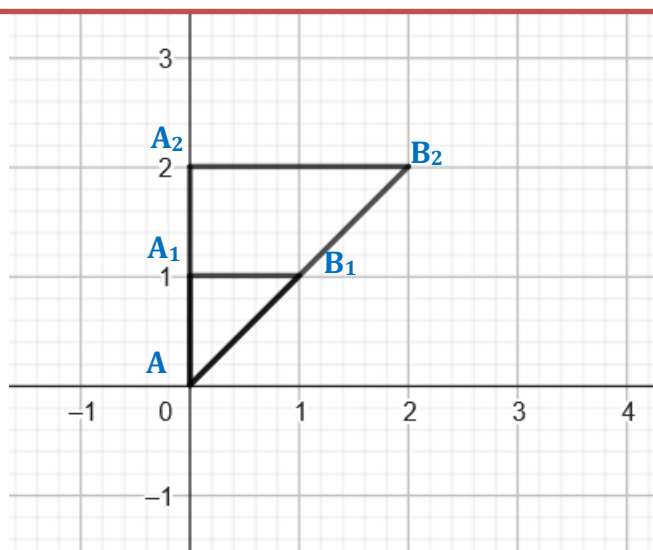
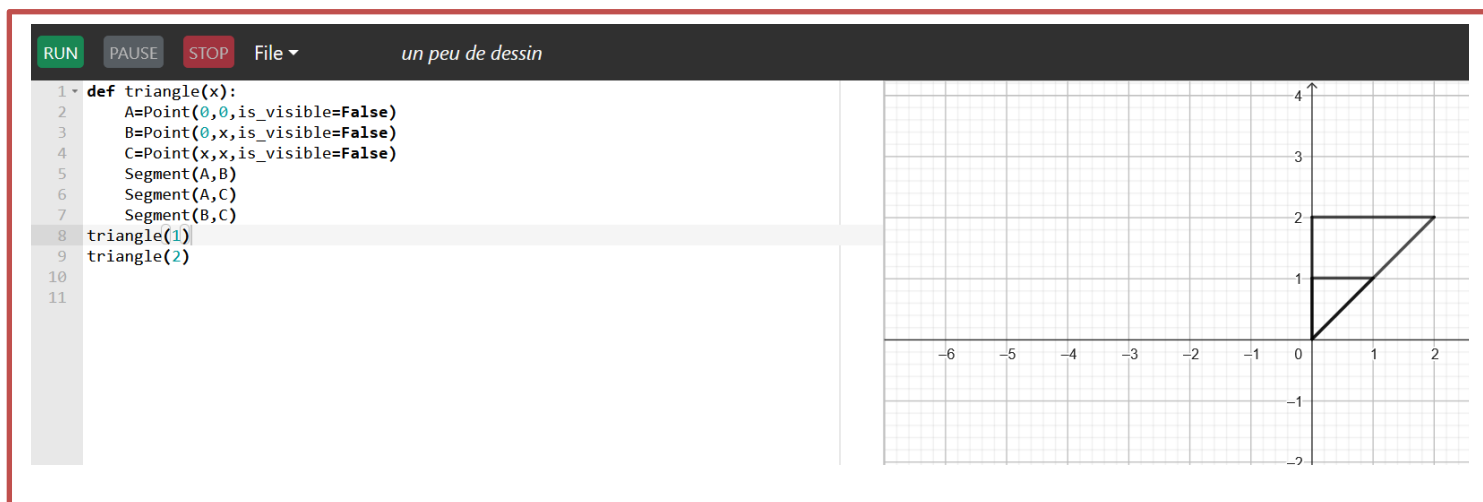
- 6.1) **Copier** puis **compléter** le code ci-dessous sur geogebra-python afin de de définir la fonction triangle :

```
def triangle(x):
    A=Point(...,..., is_visible=False)
    B=Point(...,..., is_visible=False)
    C=Point(...,..., is_visible=False)
    Segment(A,B)
    Segment(A,C)
    Segment(B,C)
```

- 6.2) **Exécuter** le programme en cliquant sur « Run », Que remarque-t-on ?
On constate que rien ne se passe. En effet, ce programme permet simplement de définir un triangle et non de le tracer

- 6.3) **Ajouter** au programme précédent la commande : **triangle(1)**
Que constate-t-on ?
On obtient un triangle dont les 3 sommets on pour coordonnées (0,0) ,(0,1) et (1,1)

- 6.4) **Ajouter** au programme précédent la commande : **triangle (2)**
Que constate-t-on ?
On obtient un deuxième triangle dont les 3 sommets on pour coordonnées (0,0) ,(0,2) et (2,2)



7.1) Donner les coordonnées des points A, A₁, A₂, B₁ et B₂

Les coordonnées des points A, A₁, A₂, B₁ et B₂ sont :

A (0 ;0)

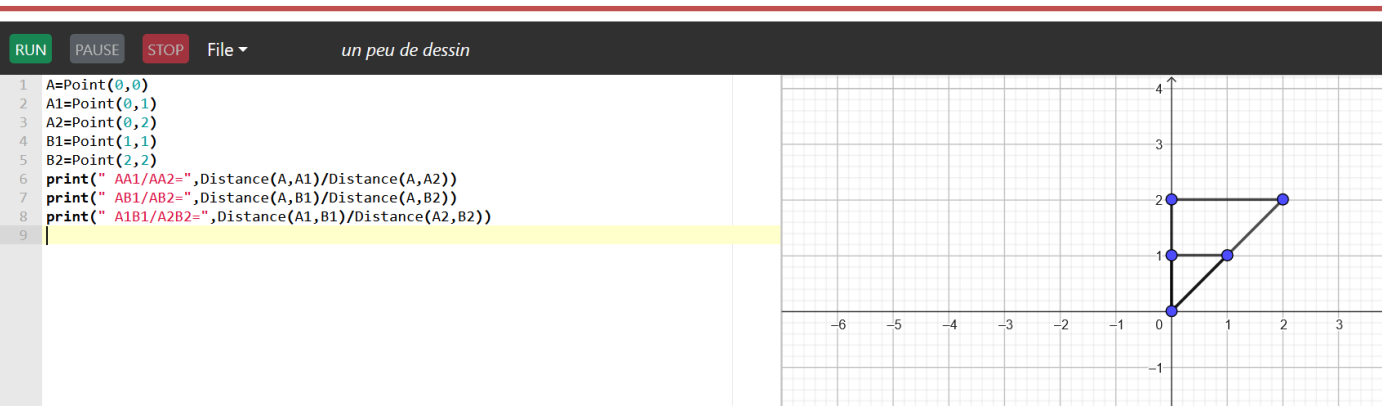
A₁(0 ;1)

A₂(0 ;2)

B₁(1 ;1)

B₂(2 ;2)

7.2) Ecrire un code permettant de vérifier le théorème de Thalès dans la configuration obtenue



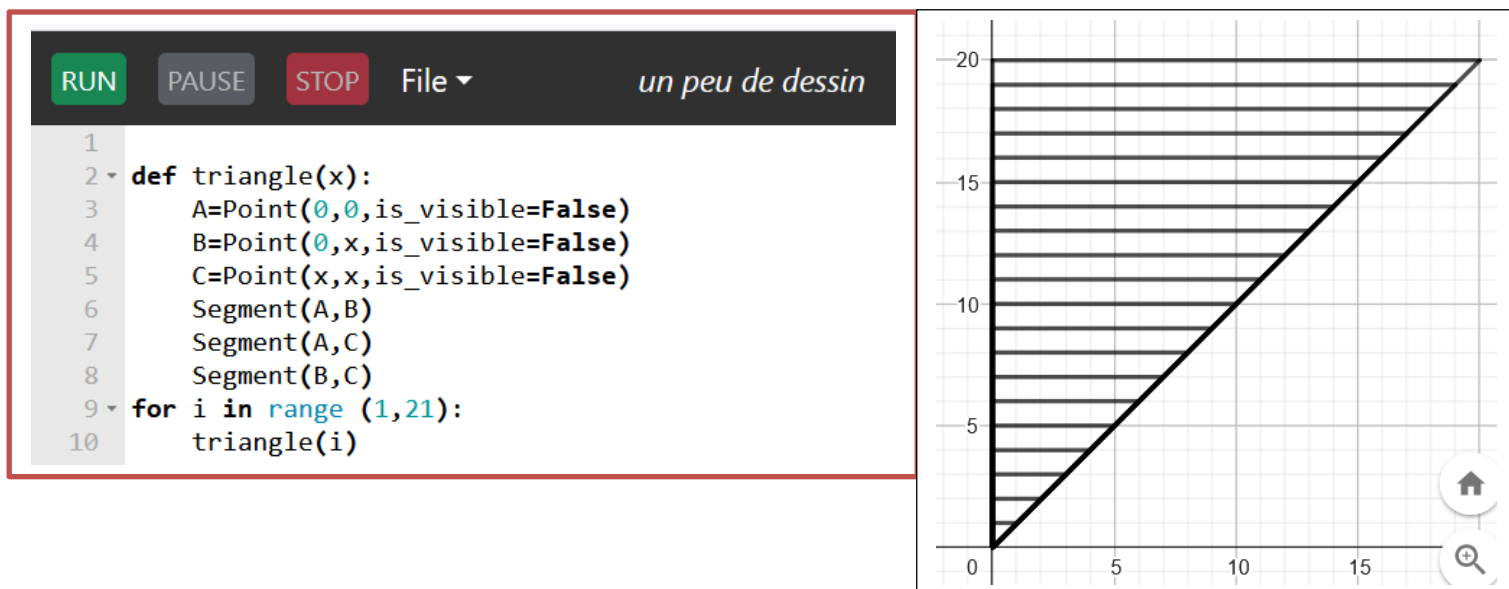
On est bien dans une configuration de Thalès :

- Les points A, A₁, A₂ et les points A, B₁ et B₂ sont alignés dans le même ordre
- Les droites (A₁, B₁) et (A₂, B₂) sont parallèles

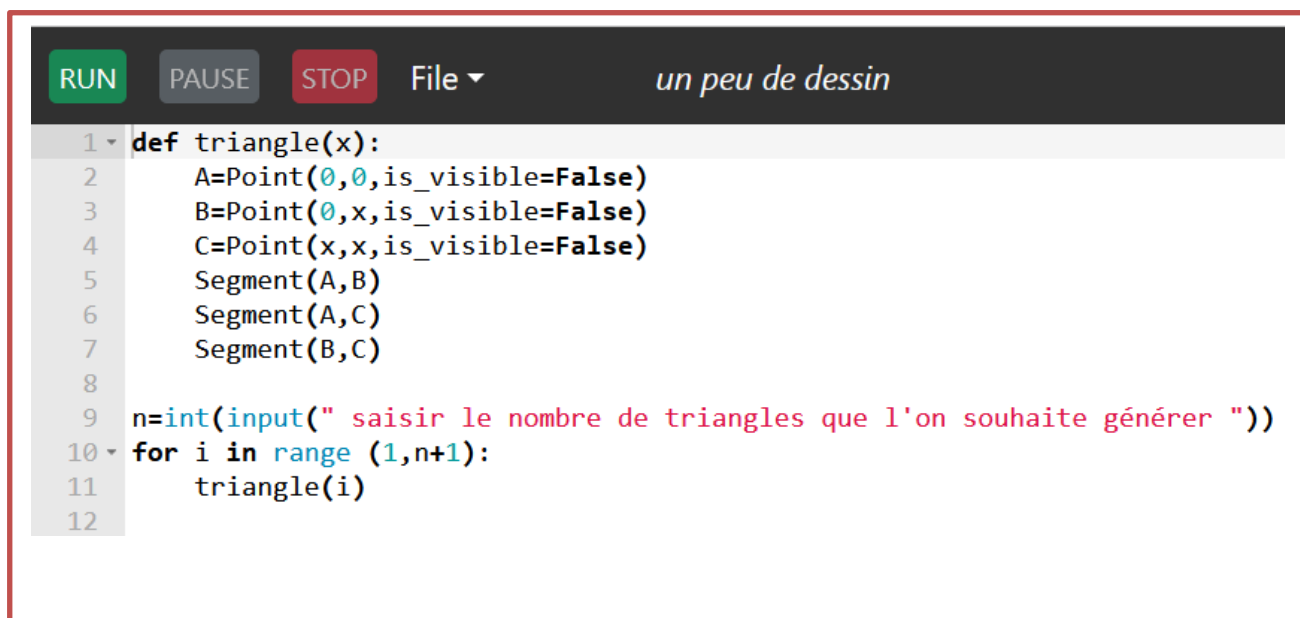
La relation $\frac{AA_1}{AA_2} = \frac{AB_1}{AB_2} = \frac{A_1B_1}{A_2B_2}$ est bien vérifiée

8) On souhaiterait générer 20 triangles

8.1) **Proposer** un code permettant de générer 20 triangles à partir de la fonction triangle définie à la question 6

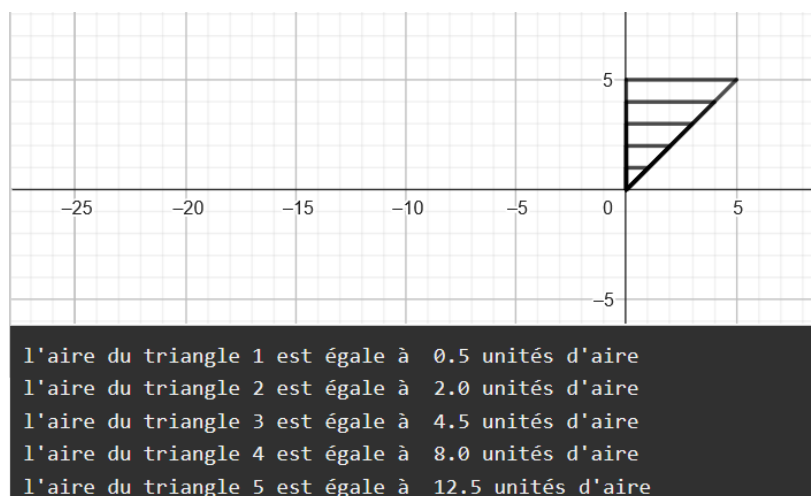


8.2) Modifier le programme précédent afin qu'il demande de saisir le nombre de triangles à générer avant de les tracer



8.3) Compléter le programme précédent afin qu'il permette de calculer (et d'afficher) l'aire des différents triangles générés

```
RUN PAUSE STOP File ▾ un peu de dessin
1 def triangle(x):
2     A=Point(0,0,is_visible=False)
3     B=Point(0,x,is_visible=False)
4     C=Point(x,x,is_visible=False)
5     Segment(A,B)
6     Segment(A,C)
7     Segment(B,C)
8
9     n=int(input(" saisir le nombre de triangles que l'on souhaite générer "))
10    for i in range(1,n+1):
11        triangle(i)
12        print("l'aire du triangle",i,"est égale à ",i*i/2,"unités d'aire")
13
14
```



9)

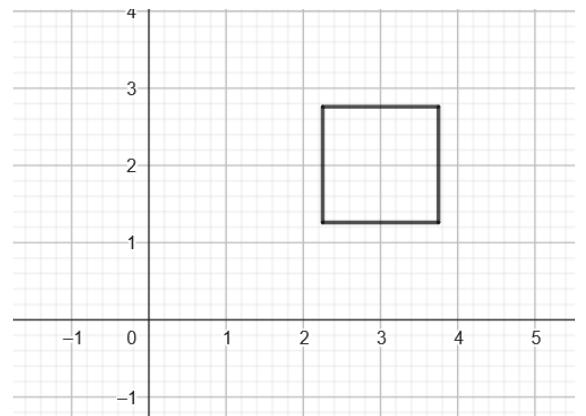
9.1) Proposer un programme permettant de définir la fonction « carre » permettant de tracer un carré :

- Centré au point de coordonnées (x, y)
- De côté de longueur c

9.2) Appliquer ce programme pour tracer un carré de centré au point de coordonnées (3,2) et dont la longueur d'un côté est 1,5

RUN PAUSE STOP File *un peu de dessin*

```
1 def carre(x,y,c):  
2     A=Point(x-c/2,y-c/2, is_visible=False)  
3     B=Point(x+c/2,y-c/2, is_visible=False)  
4     C=Point(x-c/2,y+c/2, is_visible=False)  
5     D=Point(x+c/2,y+c/2, is_visible=False)  
6     Segment(A,B)  
7     Segment(A,C)  
8     Segment(D,B)  
9     Segment(D,C)  
10    carre(3,2,1.5)
```



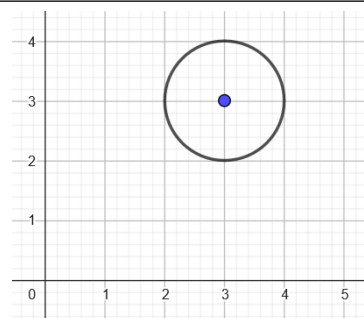
10) Voici le code permettant de tracer le cercle centré au point de coordonnées (x,y) et de diamètre d

```
def cercle(x,y,d):  
  
    Circle(x,y,d/2)
```

10.1) Exécuter ce programme après avoir ajouté une ligne permettant de tracer le cercle de centre I(3,3) et de rayon 2

RUN PAUSE STOP File

```
1 def cercle(x,y,d):  
2     Circle(x,y,d/2)  
3     cercle(3,3,2)  
4  
5  
6
```

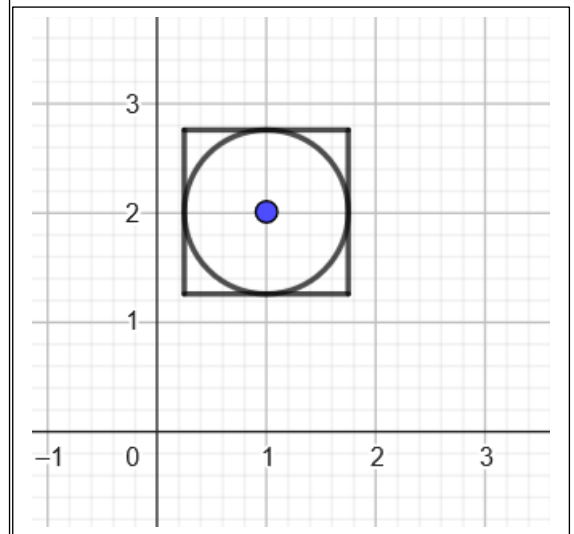


10.2) Proposer un programme permettant de tracer le cercle inscrit dans le carré centré au point de coordonnées (x,y) de côté c

```

RUN PAUSE STOP File ▾ un peu de dessin
1 def carre(x,y,c):
2     A=Point(x-c/2,y-c/2, is_visible=False)
3     B=Point(x+c/2,y-c/2, is_visible=False)
4     C=Point(x-c/2,y+c/2, is_visible=False)
5     D=Point(x+c/2,y+c/2, is_visible=False)
6     Segment(A,B)
7     Segment(A,C)
8     Segment(D,B)
9     Segment(D,C)
10
11 def cercle(x,y,d):
12     Circle(x,y,d/2)
13
14 carre(1,2,1.5)
15 cercle(1,2,1.5)

```

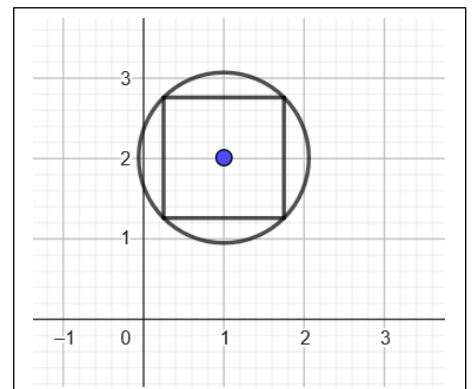


10.3) Proposer un programme permettant de tracer le carré inscrit dans cercle centré au point de coordonnées (x,y) de côté c

```

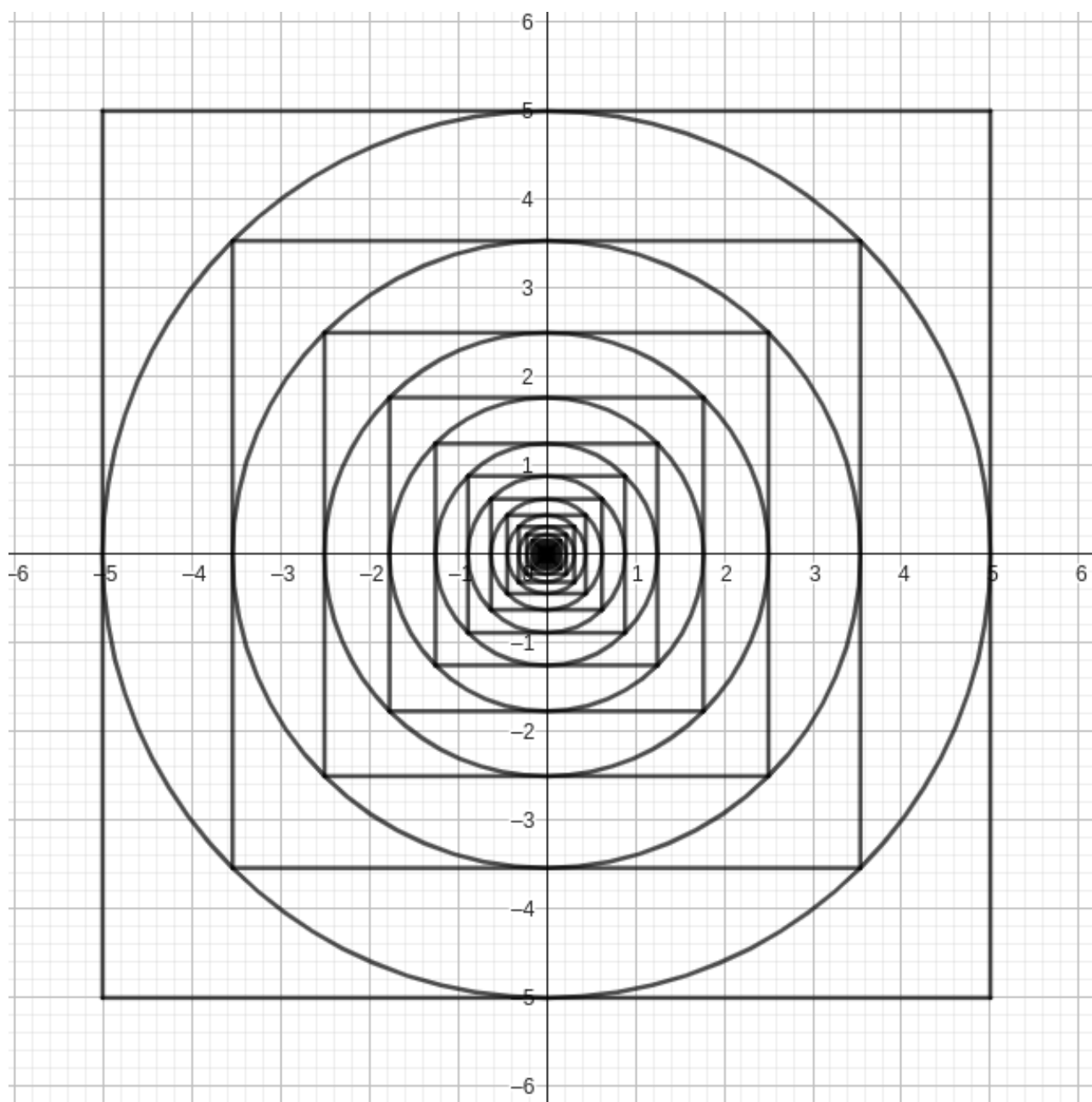
RUN PAUSE STOP File ▾ un peu de dessin
1 from math import*
2 def carre(x,y,c):
3     A=Point(x-c/2,y-c/2, is_visible=False)
4     B=Point(x+c/2,y-c/2, is_visible=False)
5     C=Point(x-c/2,y+c/2, is_visible=False)
6     D=Point(x+c/2,y+c/2, is_visible=False)
7     Segment(A,B)
8     Segment(A,C)
9     Segment(D,B)
10    Segment(D,C)
11
12 def cercle(x,y,d):
13    Circle(x,y,sqrt(2)*d/2)
14
15 carre(1,2,1.5)
16 cercle(1,2,1.5)

```



carre(1,2,1.5)
cercle(1,2,1.5)

11) Proposer un programme permettant d'obtenir la figure ci-dessous :



TP Python-Géogebra 2 : PRINCIPE DE FERMAT

Le but de ce TP est de se familiariser à la programmation en utilisant l'outil Python-Geogebra, disponible en ligne sur le site et réinvestir t les notions travaillées en sciences physiques (Optique)

<https://geogebra.org/python/index.html>

Le principe de Fermat*, appelé aussi « principe du moindre temps », énonce que la lumière se propage entre deux points en suivant la trajectoire qui minimise son temps de parcours. Ce TP aborde la démonstration de l'équivalence entre la loi de Snell-Descartes et le principe de Fermat dans le cas de la réfraction

* Pierre de Fermat était un mathématicien et physicien français (1601-1665)

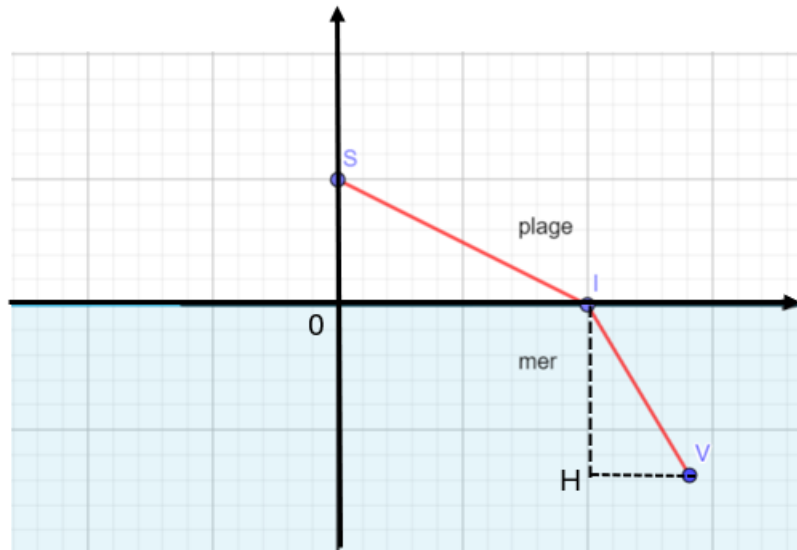
Un maître-nageur est situé en S (sauveteur) sur la plage, alors qu'un vacancier situé en V est sur le point de se noyer. Le maître-nageur peut courir avec la vitesse v_1 et nager avec la vitesse v_2 . Il se déplace en ligne droite sur la plage comme dans l'eau. Il atteint l'eau au point I (repéré par l'abscisse x).

On rappelle que la vitesse moyenne est donnée par la formule :

$$v = \frac{d}{t} \text{ où } v \text{ est la vitesse exprimée en mètres par seconde (m/s)}$$

d la distance parcourue en mètres (m)

t la durée du parcours en secondes (s)

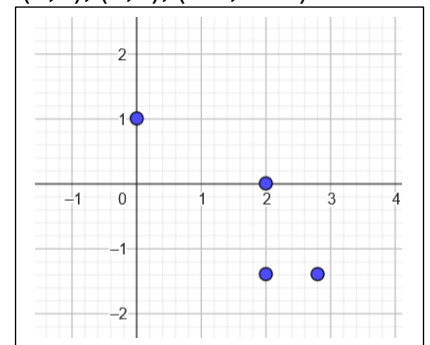


- 12) Commencer** par recopier la lignes suivante dans l'outil Python-géogebra
 Cette ligne permet d'importer les bibliothèques python dont nous aurons besoin.

```
from math import *
```

- 13) Ajouter** les points S, I, Vet H de coordonnées respectives (0,1), (2,0), (2.8, -1.4) et (2,-1.4)

```
1 from math import*
2 S=Point(0,1)
3 I=Point(2,0)
4 H=Point(2,-1.4,is_visible=False)
5 V=Point(2.8,-1.4)
```



- 14) Tracer** ensuite le segment [SI] et [IV]

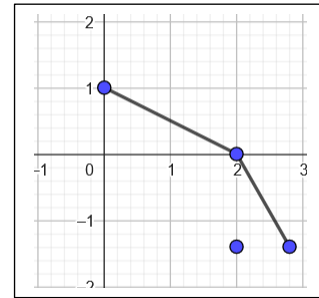
On rappelle que la commande permettant de tracer le segment [AB] est :

```
Segment(A,B)
```

```

1 from math import*
2 S=Point(0,1)
3 I=Point(2,0)
4 H=Point(2,-1.4)
5 V=Point(2.8,-1.4)
6 Segment(S,I)
7 Segment(V,I)

```



15) **Modifier** le programme de manière à ce que le point H ne soit plus visible

RAPPEL : Pour ne plus voir un point bleu à l'emplacement des points, on pourra choisir l'option « **is_visible=False** » lors de la création des points, avec la commande suivante.

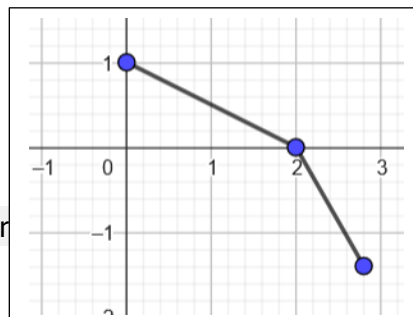
Point(x,y, is_visible=False)

```

1 from math import*
2 S=Point(0,1)
3 I=Point(2,0)
4 H=Point(2,-1.4,is_visible=False)
5 V=Point(2.8,-1.4)
6 Segment(S,I)
7 Segment(V,I)

```

peut être



AB=Distance(A,B)

Si l'on veut voir afficher le résultat du calcul, on utilisera la fonction “**print**” :

print("AB=", AB))

5.1) **Compléter** le programme afin qu'il permette d'afficher les longueurs SI puis IV

5.2) **Compléter** le programme afin qu'il permette d'afficher la distance totale parcourue par le maître-nageur dans le cas où il emprunterait le chemin matérialisé par les segments [SI] et [IV].

L'unité de longueur est le mètre

La distance sera arrondie à 10^{-2} près

```

1 from math import*
2 S=Point(0,1)
3 I=Point(2,0)
4 H=Point(2,-1.4,is_visible=False)
5 V=Point(2.8,-1.4)
6 Segment(S,I)
7 Segment(V,I)
8 SI=Distance(S,I)
9 IV=Distance(I,V)
10 print("la distance parcourue totale par le maitre-nageur est ",SI+IV,"m")
11

```

la distance parcourue totale par le maitre-nageur est 3.8485195271595 m

```
SI=Distance(S,I)
IV=Distance(I,V)
print("la distance parcourue totale par le maitre-nageur est ",SI+IV,"m")
```

Pour arrondir un résultat à deux chiffres après la formule on utilise le code suivant :

```
round(résultat,2)
```

En modifie pour cela la ligne 10 comme suit :

```
print("la distance parcourue totale par le maitre-nageur est ",round(SI+IV,2),"m")
```

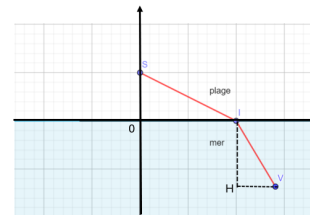
```
la distance parcourue totale par le maitre-nageur est 3.85 m
```

17) Nous aimerions maintenant créer la fonction « **temps** » qui prenne en entrée :

- L'abscisse x du point I (point d'entrée dans l'eau)
- L'ordonnée y_S du point S (définissant la position du vacancier)
- Les coordonnées x_V et y_V du point V (définissant la position du vacancier)
- La vitesse moyenne de déplacement du sauveteur v_1 en m/s sur la plage
- La vitesse moyenne de déplacement du sauveteur v_2 en m/s dans l'eau

Cette fonction permettra de déterminer le temps mis pour le sauveteur pour aller secourir le vacancier. Dans la fonction de temps, nous aurons les paramètres précédents précédemment pour différentes positions du sauveur et du vacancier moi

- De manière à ce qu'il, après exécution de celui-ci, den les 6 variables mentionnées précédemment
- En considérant que les coordonnées des points S, I, V
 - S(0, y_S)
 - I(x , 0)
 - V(x_V , y_V)
 - H(x , y_V)
- De manière à ce qu'il affiche les coordonnées des points S, I, V et H



Le code permettant de faire en sorte qu'une fois le programme lancé, on est amené à saisir la valeur d'un paramètre p est :

```
p = float(input (" saisir la valeur de x:  x = "))
```

Le code permettant d'afficher les coordonnées d'un point A (x_A , y_A) , dans le cas où les coordonnées ont été saisies suite à la demande du programme après exécutions, est :

```
print("les coordonnées du point A sont : V(",x_A,"",y_A,"")
```

6.3) On rappelle que la vitesse v (en m/s) d'un objet en fonction de la distance parcourue d (en m) et de la durée t (en s) est donnée par la formule :

$$v = \frac{d}{t}$$

Exprimer la durée t en fonction de la distance d et de la vitesse v

$$v = \frac{d}{t} \implies t = \frac{d}{v}$$

6.4) Exprimer le temps t_1 mis par le sauveteur pour parcourir la distance SI en fonction de la vitesse v_1

$$t_1 = \frac{SI}{v_1}$$

6.5) Exprimer le temps t_2 mis par le sauveteur pour parcourir la distance IV en fonction de la vitesse v_2

$$t_2 = \frac{IV}{v_2}$$

6.6) En déduire le temps t mis par le sauveteur pour aller secourir le vacancier en fonction des distances SI et IV et des vitesses v_1 et v_2

$$temps = \frac{SI}{v_1} + \frac{IV}{v_2}$$

6.7) Créer maintenant la fonction temps dépendant des 6 paramètres mentionnés précédemment.

6.8) Tester le programme dans le cas où :

- S(0,20) I(50,0) V(60,-100)
- La vitesse moyenne du sauveteur sur la plage est de 5m/s
- La vitesse moyenne du sauveteur dans est de 2m/s

```

1 from math import*
2 y_S=float(input("saisir la position (ordonnée) du sauveteur: y_S= "))
3 x=float(input("saisir la position (abscisse) d'entrée dans la mer: x= "))
4 x_V=float(input("saisir la position (abscisse) du vacancier: x_V= "))
5 y_V=float(input("saisir la position (ordonnée) du vacancier: y_V= "))
6 v_1=float(input("saisir la vitesse du sauveteur sur la plage:v_1= "))
7 v_2=float(input("saisir la vitesse du sauveteur sur la plage:v_2= "))
8 S=Point(0,y_S)
9 I=Point(x,0)
10 H=Point(x,y_V,is_visible=False)
11 V=Point(x_V,y_V)
12 Segment(S,I)
13 Segment(V,I)
14 SI=Distance(S,I)
15 IV=Distance(I,V)
16 print("les coordonnées du point S sont : S(0,",y_S,")")
17 print("les coordonnées du point V sont : V(",x_V,",",y_V,")")
18 print("les coordonnées du point H sont : V(",x,",",y_V,")")
19 print("les coordonnées du point I sont : V(",x,",0)")
20 print("la distance parcourue totale par le maitre-nageur est ",round(SI+IV,2),"m")
21
22 def temps(x,x_S,x_V,y_V,v_1,v_2):
23     return(sqrt(x**2+y_S**2)/v_1+sqrt((x_V-x)**2+y_V**2)/v_2)

```

```

print("les coordonnées du point H sont : V(",x,"",y_V,"")
print("les coordonnées du point I sont : V(",x,"",0) ")
print("la distance parcourue totale par le maitre-nageur est
",round(SI+IV,2),"m")

```

```

def temps(x,x_S,x_V,y_V,v_1,v_2):
    return(sqrt(x**2+y_S**2)/v_1+sqrt((x_V-x)**2+y_V**2)/v_2)

```

Exprimer le temps T en fonction de x , x_V , y_V , v_1 et v_2 mis par le sauveteur pour parcourir la distance SI+IV

$$v = \frac{d}{t} \implies t = \frac{d}{v}$$

$$T = \frac{SI}{v_1} + \frac{IV}{v_2}$$

$$T = \frac{\sqrt{x^2 + y_S^2}}{v_1} + \frac{\sqrt{(x_V - x)^2 + y_V^2}}{v_2}$$

6.9) On admet maintenant que :

- $T = \frac{\sqrt{x^2 + y_S^2}}{v_1} + \frac{\sqrt{(x_V - x)^2 + y_V^2}}{v_2}$
- La vitesse du sauveteur dans le sable est de 5m /s
- La vitesse du sauveteur dans l'eau est de 2m /s
- La position du sauveteur est S (0 ;4)
- La position du vacancier est V (2,-6)

Exprimer T en fonction de x

$$T = \frac{\sqrt{x^2 + y_S^2}}{v_1} + \frac{\sqrt{(x_V - x)^2 + y_V^2}}{v_2}$$

$$T = \frac{\sqrt{x^2 + 4^2}}{5} + \frac{\sqrt{(2 - x)^2 + (-6)^2}}{2}$$

$$T = \frac{\sqrt{x^2 + 16}}{5} + \frac{\sqrt{4 - 2x + x^2 + 36}}{2}$$

$$T = \frac{\sqrt{x^2 + 16}}{5} + \frac{\sqrt{x^2 - 2x + 40}}{2}$$