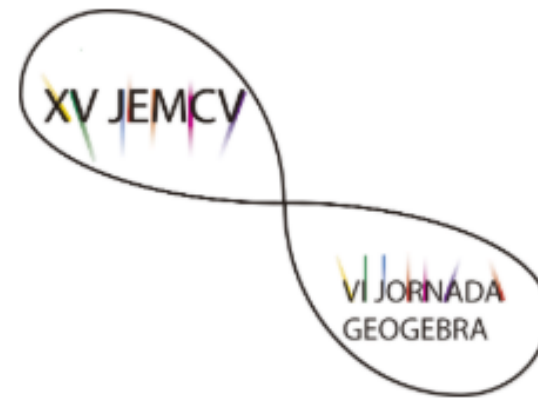




XV JORNADES D'EDUCACIÓ MATEMÀTICA DE LA COMUNITAT VALENCIANA
VI JORNADA GEOGEBRA DE LA COMUNITAT VALENCIANA
PLANTEJANT REPTES: MATEMÀTIQUES PER A PENSAR MILLOR

València, 17 i 18 d'octubre de 2025
Universitat Politècnica de València

PYTHON CON GEOGEBRA

Mónica Soler Montaner

`m.solermontaner@edu.gva.es`

`msolmon@mat.upv.es`

PyGgb WebApp

Plataforma online donde se puede programar en Python con objetos de GeoGebra y visualizar la salida en la ventana gráfica de GeoGebra.

ACCESO : <https://geogebra.org/python>

PyGgb WebApp Interface

RUN

PAUSE

STOP

File ▾

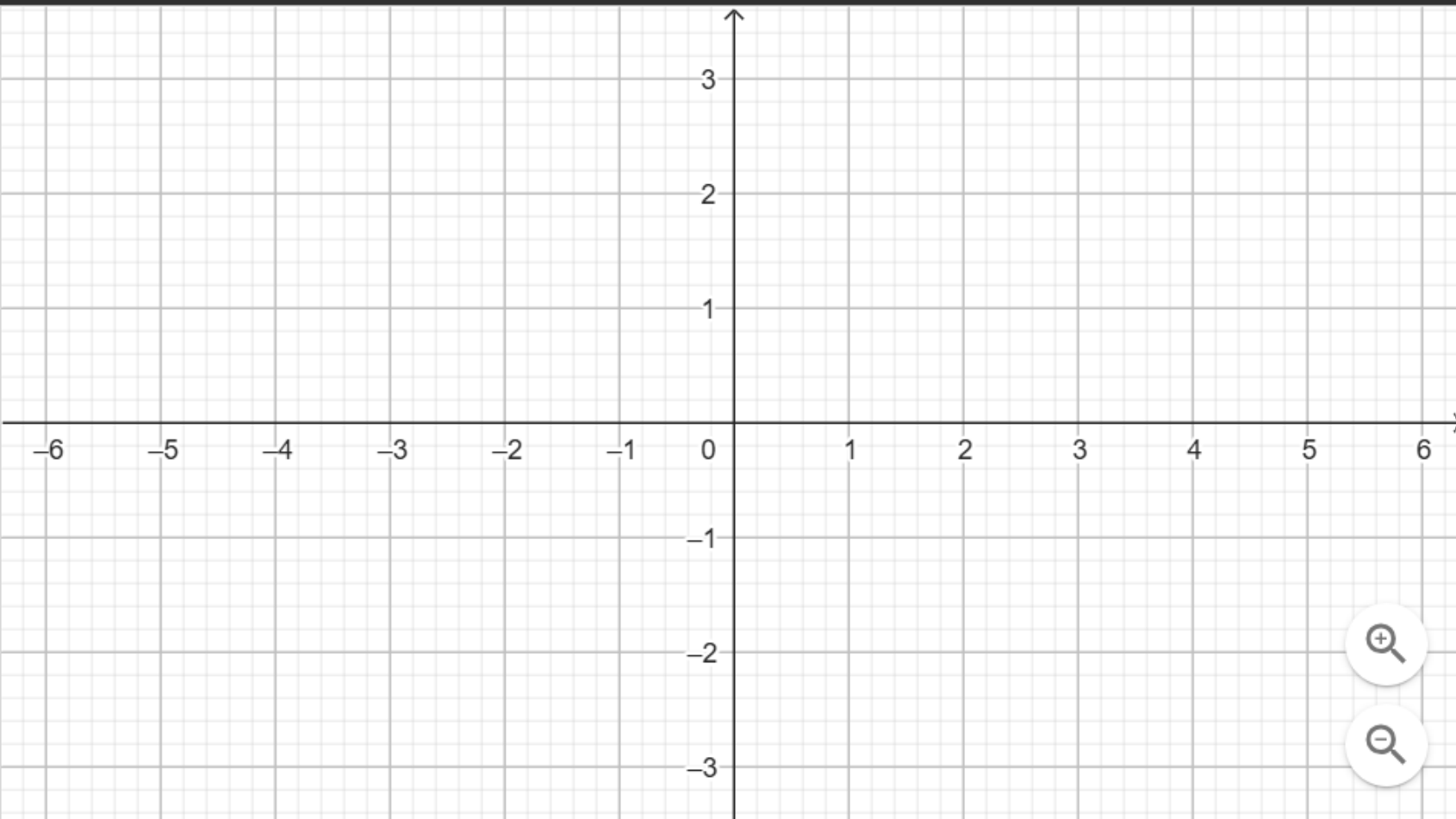
Programa 1

[Docs](#)

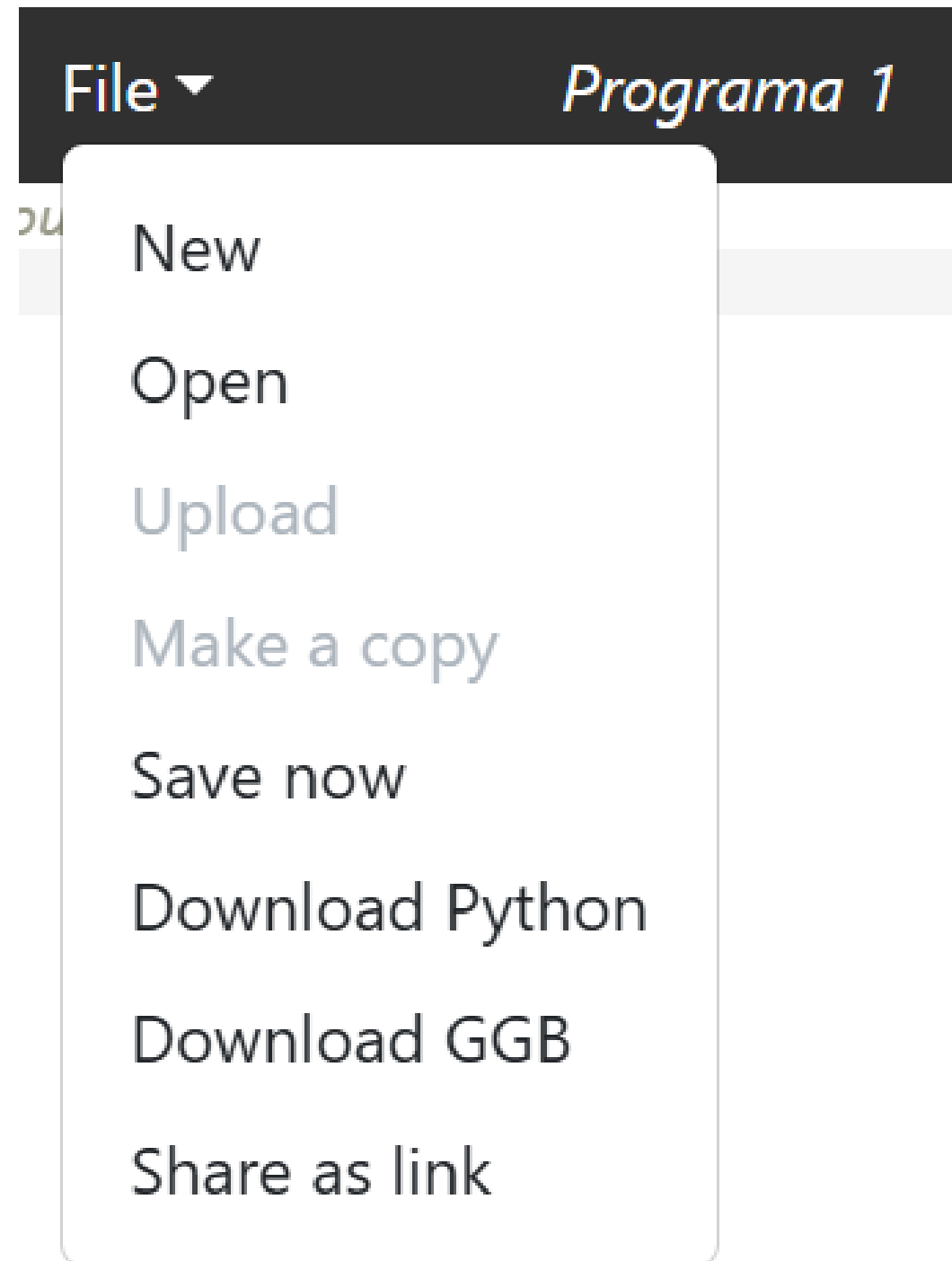
About

1 # Start writing your code!

2 |



PyGgb WebApp Interface



Objetos de GeoGebra

OBJETO	EXPRESIÓN EN GEOGEBRA
PUNTO	Point (coordenada x, coordenada y)
RECTA	Line (Punto, Punto); Line (pendiente, ordenada en el origen)
SEGMENTO	Segment (Punto, Punto)
CÍRCUNFERENCIA	Circle (Centro, radio); Circle (Centro, punto)
ELIPSE	Ellipse (Punto, Punto, semieje mayor)

Objetos de GeoGebra

OBJETO	EXPRESIÓN EN GEOGEBRA
PARÁBOLA	Parabolla(Origen, directriz); Parabolla (a, b, c)
POLÍGONO	Polygon(Puntos); Polygon(Punto, Punto, número de lados)
VECTOR	Vector(Punto, Punto); Vector(coordenada x, coordenada y)
DESLIZADOR	Slider(mín, máx, is_visible = True, label_visible = True)

Propiedades de los objetos

PROPIEDAD
<code>nombre.is_visible = True / False</code>
<code>nombre.is_independent = True / False (PUNTOS)</code>
<code>nombre.color = 'black'</code>
<code>nombre.color = (0.1, 0.2, 0.2)</code>

Propiedades de los objetos

PROPIEDAD
nombre.size = 2 (PUNTOS)
nombre.line_thickness = 2
nombre.opacity = 1

Funciones de GeoGebra

FUNCIÓN
Distance(obj1, obj2)
Intersect(obj1, obj2)
NumberOfObjects()
Rotate(obj, ángulo)

Funciones de Geogebra

FUNCIÓN
Rotate(obj, ángulo, punto)
ZoomIn()
ZoomIn(k)
ZoomIn(k, p)

Funciones de Geogebra

FUNCIÓN
ZoomIn(k, coordenadas)

Operadores de Python

OPERADOR	EXPRESIÓN
SUMA	+
RESTA	-
PRODUCTO	*
DIVISIÓN	/
POTENCIA	**

Operadores de Python

OPERADOR	EXPRESIÓN
DIVISIÓN ENTERA	//
RESTO DE LA DIVISIÓN	%
ASIGNAR VALORES	=
VALOR ABSOLUTO	abs()
COMENTARIOS	# O “””” “”””

Operadores lógicos de Python

OPERADOR	EXPRESIÓN
y	and / &
o	or /
negación	not / ~
igual	==
distinto	!=

Operadores lógicos de Python

OPERADOR	EXPRESIÓN
mayor	>
mayor o igual	>=
menor	<
menor o igual	<=

Sentencias if de Python

if *condición* :

....

if condición1:

...

elif condición2:

...

else:

...

Bucles en Python

while condición:

...

for n in range (0, 10):

...

for n in range (0, 10, 2):

...

Listas en Python

DEFINICIÓN	<code>nombre_lista=[A, B, C]</code>
LISTA VACÍA	<code>nombre_lista = []</code>
LONGITUD DE LA LISTA	<code>len(nombre_lista)</code>
ELEMENTO EN LA POSICIÓN CERO	<code>nombre_lista[0]</code>

Listas en Python

AÑADIR UN ELEMENTO AL FINAL DE LA LISTA	<code>nombre_lista.append(elemento)</code>
AÑADIR UN ELEMENTO EN UNA POSICIÓN DETERMINADA	<code>nombre_lista.append(elemento,'posicion')</code>

Entrada y salida de datos

ENTRADA DE DATOS Y ASIGNACIÓN A UNA VARIABLE	<code>edad = int(input('Introducir edad: '))</code>
SALIDA DE DATOS	<code>print('Edad: ', edad)</code>

Funciones en Python

```
def nombre_función (parámetro1, parámetro2, ...):
```

```
...
```

Módulos en Python

1) MÓDULO MATH

import math

math.pi

math.e

math.sqrt()

Módulos en Python

2) MÓDULO RANDOM

```
import random
```

```
random.randint(a, b)
```

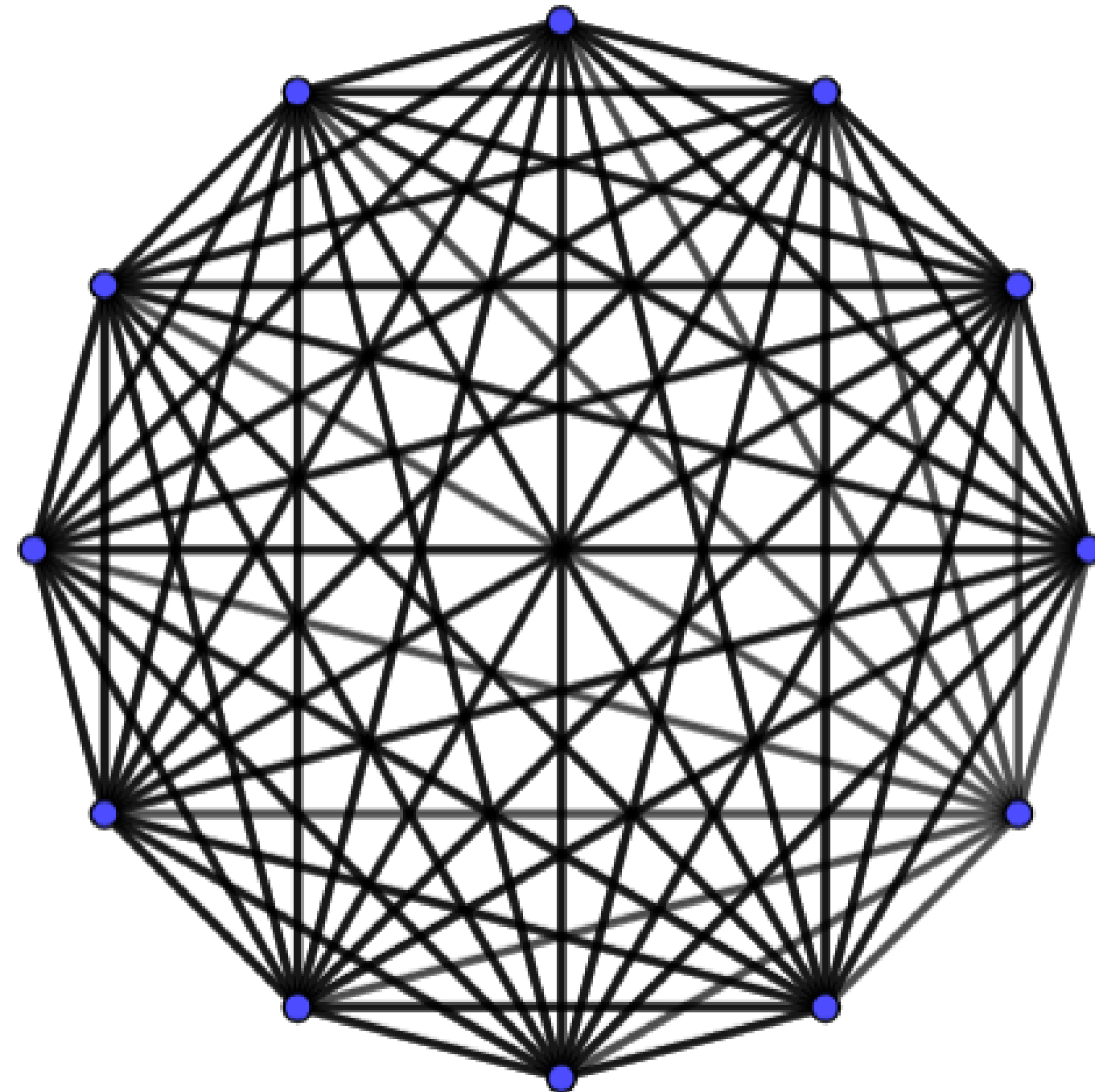
```
random.uniform(a, b)
```

Módulos en Python

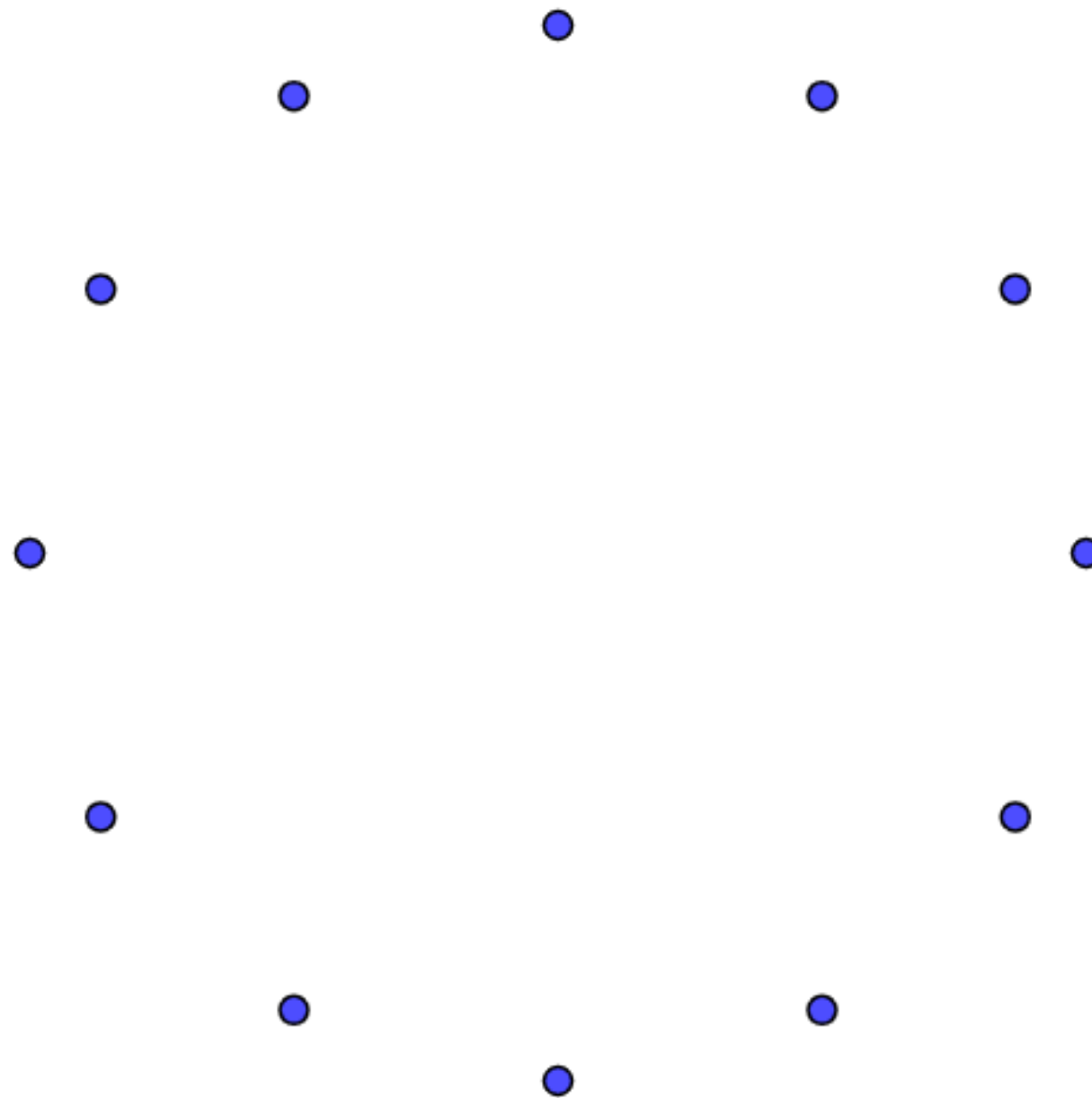
3) MÓDULO TIME

```
import time  
time.sleep(0.01)
```

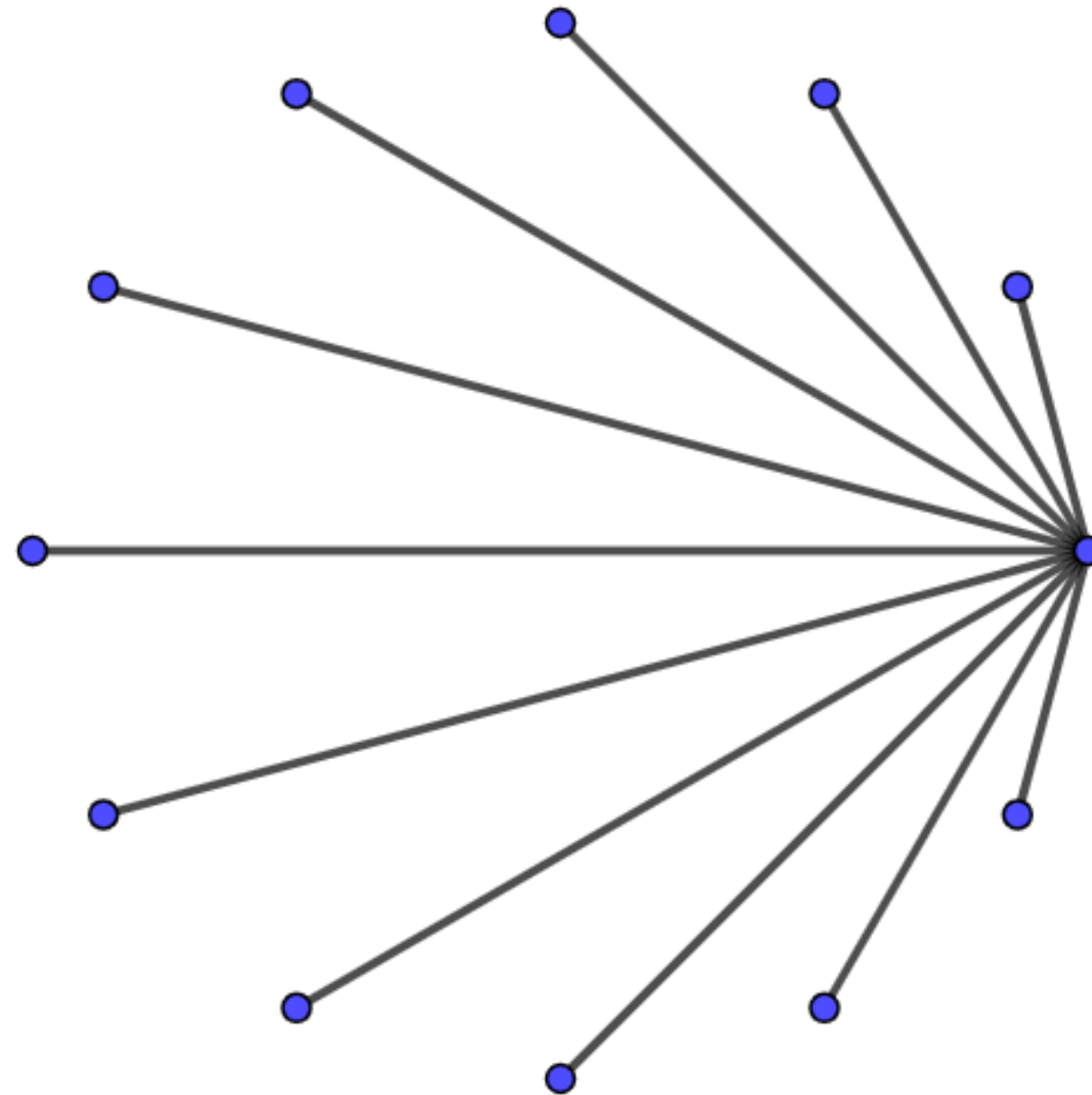
Construcción geométrica 1



Construcción geométrica 1



Construcción geométrica 1



Construcción geométrica 1

```
import time  
import math
```

Construcción geométrica 1

#función para dibujar los puntos

def Puntos(radio,num_puntos):

m=0

while m<=2*math.pi:

punto=Point(radio*math.cos(m), radio*math.sin(m))

punto.size=4

puntos.append(punto)

m=m+2*math.pi/num_puntos

time.sleep(0.1)

Construcción geométrica 1

```
#función para dibujar los segmentos
def Segmentos(num_puntos):
    for i in range(0,num_puntos-1):
        for j in range(i+1,num_puntos):
            segmento=Segment(puntos[i], puntos[j])
            segmento.color='black'
            segmentos.append(segmento)
            time.sleep(0.1)
```

Construcción geométrica 1

```
#programa principal
```

```
time.sleep(1)
```

```
#informamos en que consiste la actividad
```

```
print('Introduce los datos que te pide el programa')
```

```
time.sleep(3)
```

```
print('y responde a las preguntas que te formula sobre la construcción.')
```

```
time.sleep(3)
```

Construcción geométrica 1

#Dibujamos una circunferencia para que el fondo sea blanco

P=Point(0,0,is_visible=False)

C=Circle(P,10)

C.color='white'

C.opacity=1

time.sleep(0.1)

Construcción geométrica 1

#nos pide el radio y el número de puntos

r= float(input('Radio:'))

n=int(input('Número de puntos:'))

Construcción geométrica 1

#creamos una lista vacía donde la función Puntos acumulará los puntos

puntos=[]

#llamamos a la función Puntos

Puntos(r,n)

#creamos una lista vacía donde la función Segmentos acumulará los segmentos

segmentos=[]

#llamamos a la función Segmentos

Segmentos(n)

time.sleep(2)

Construcción geométrica 1

#definimos la variable D como el número de diagonales

D=len(segmentos)-n

#le preguntamos al usuario ¿Cuántas diagonales tiene la construcción?

#almacenamos su respuesta en otra variable que denominamos diagonales

diagonales=int(input('¿Cuántas diagonales hay?'))

time.sleep(1)

Construcción geométrica 1

#comparamos ambas variable

#resultado correcto, lo indica

if diagonales==D:

print ('RESPUESTA CORRECTA')

#resultado incorrecto, vuelve a preguntar

#nuevo error, da la respuesta correcta

else:

diagonales1=int(input('RESPUESTA INCORRECTA. Inténtalo de nuevo. ¿Cuántas diagonales hay? '))

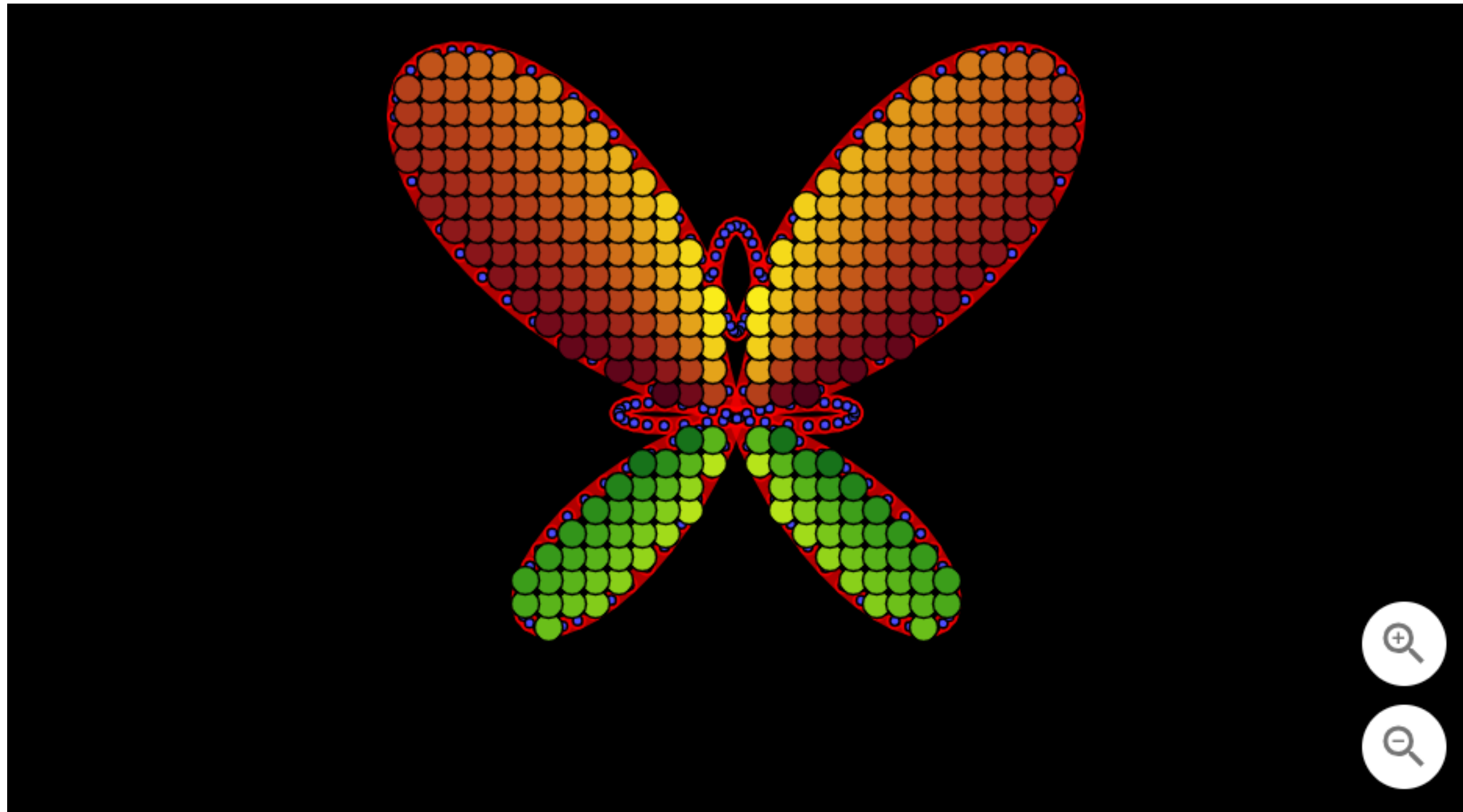
if diagonales1==D:

print ('RESPUESTA CORRECTA')

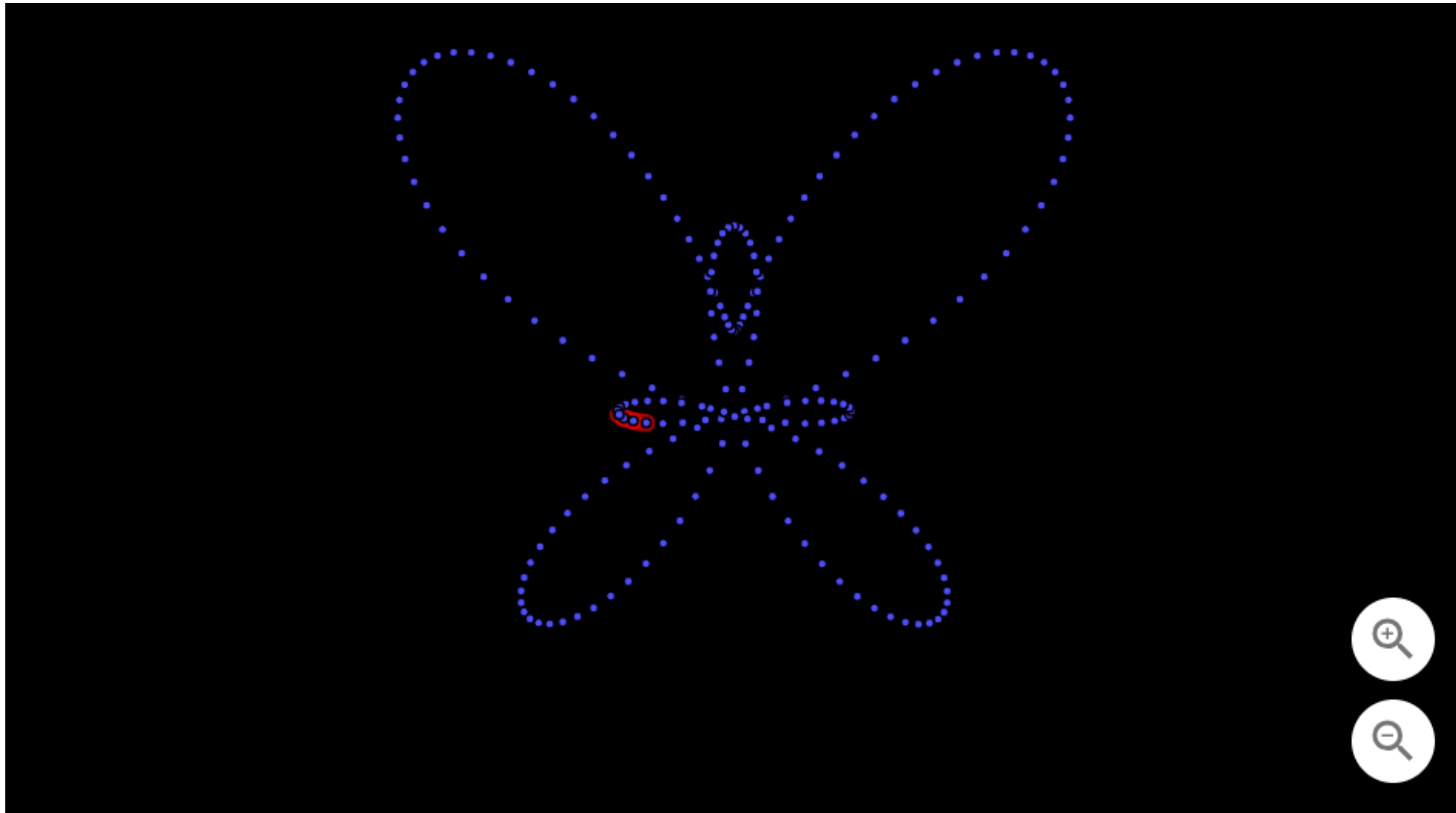
else:

print('RESPUESTA INCORRECTA. RESPUESTA CORRECTA:', D, 'diagonales.')

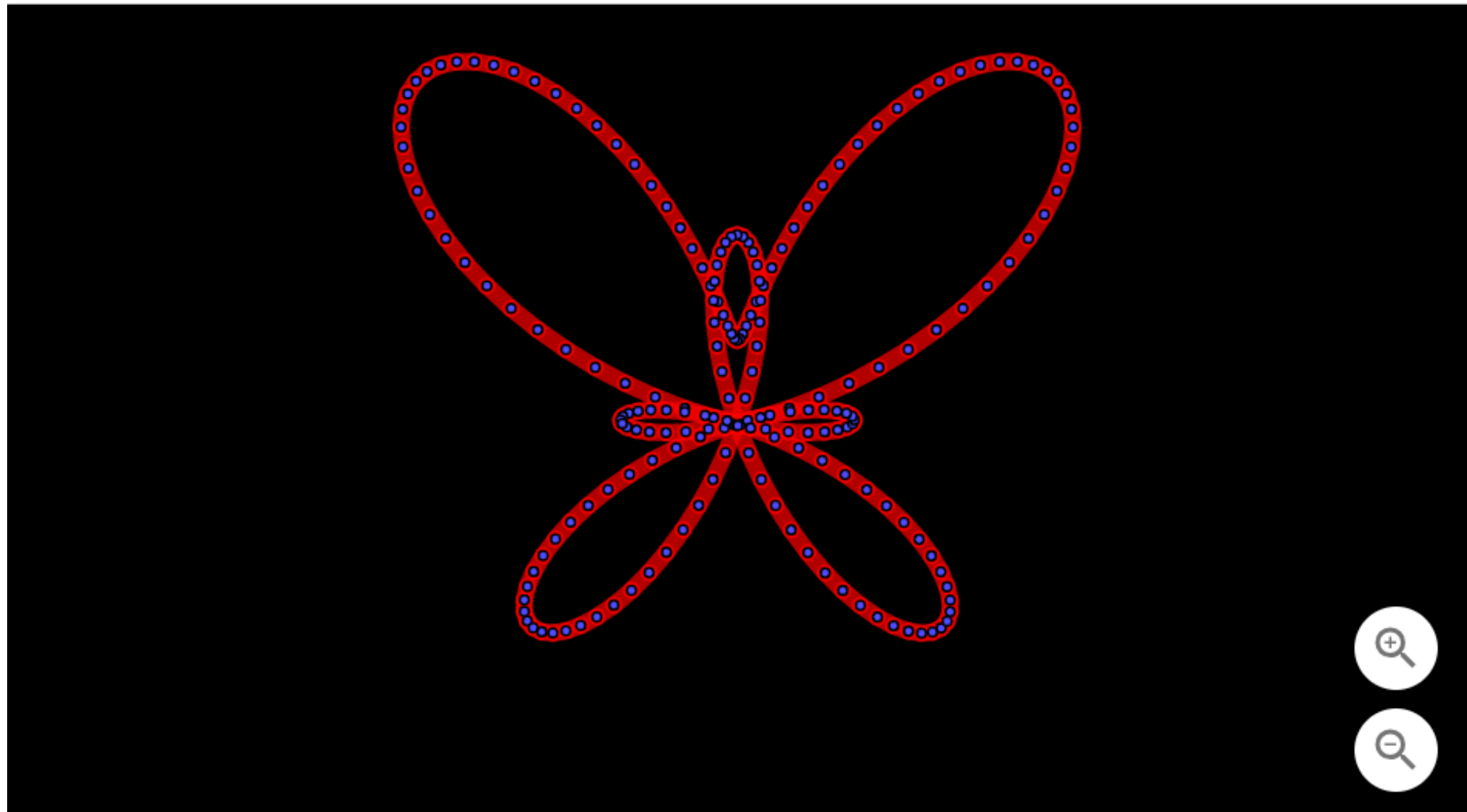
Construcción geométrica 2



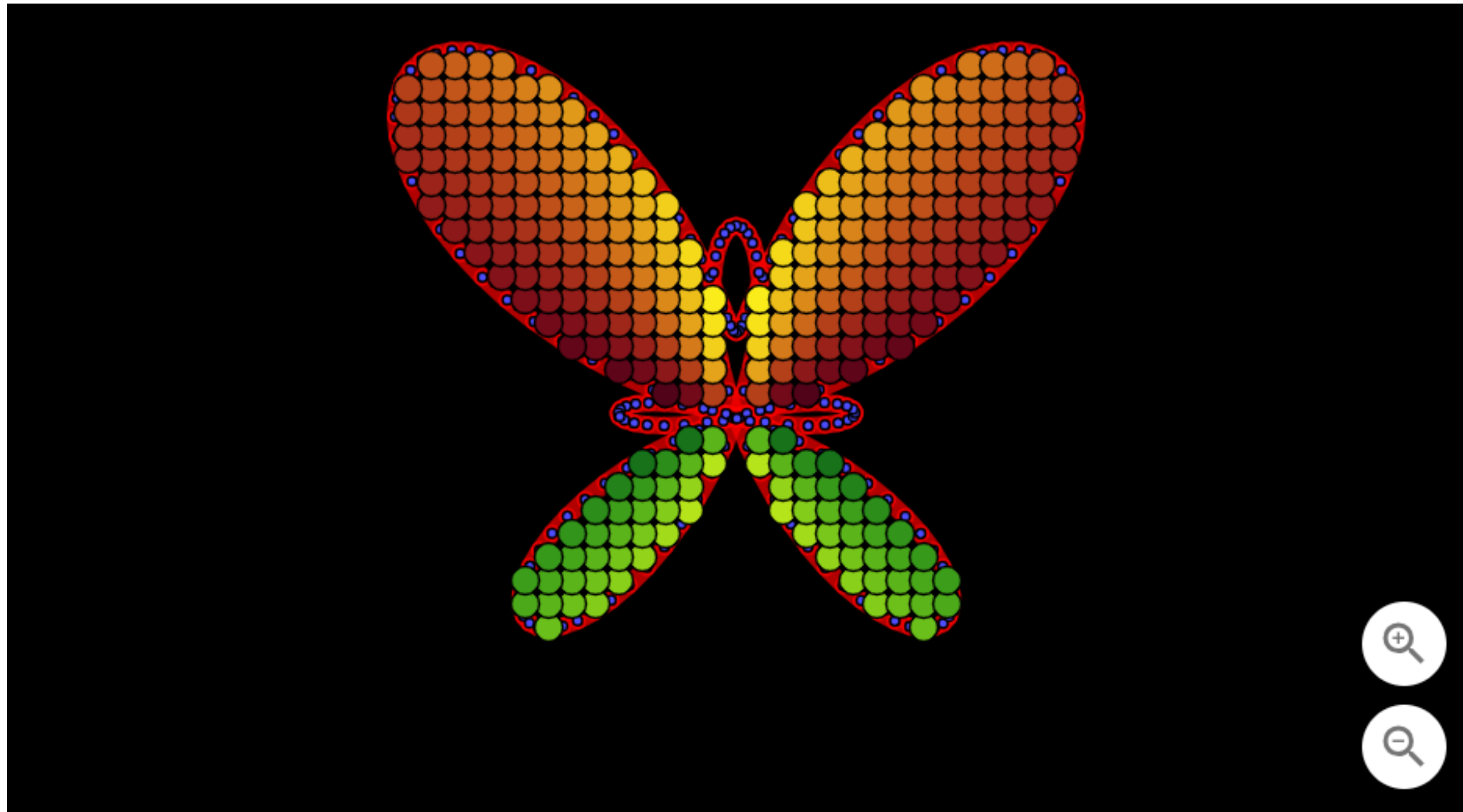
Construcción geométrica 2



Construcción geométrica 2



Construcción geométrica 2



Construcción geométrica 2

```
import time  
import math
```

Construcción geométrica 2

Ecuación polar de la mariposa $r = e^{\operatorname{sen}(\theta)} - 2\cos(4\theta) + \operatorname{sen}^5\left(\frac{2\theta - \pi}{24}\right)$

Ecuación cartesiana paramétrica de la mariposa

$$x(\theta) = \left(e^{\operatorname{sen}(\theta)} - 2\cos(4\theta) + \operatorname{sen}^5\left(\frac{2\theta - \pi}{24}\right) \right) \cos(\theta)$$

$$y(\theta) = \left(e^{\operatorname{sen}(\theta)} - 2\cos(4\theta) + \operatorname{sen}^5\left(\frac{2\theta - \pi}{24}\right) \right) \operatorname{sen}(\theta)$$

Construcción geométrica 2

#definimos una función que dibuja los puntos de la mariposa

def Puntos():

m=0

while m<=2*math.pi:

punto=Point(((math.e)(math.sin(m))-2*math.cos(4*m)+(math.sin((2*m-math.pi)/24))**5)*math.cos(m),((math.e)**(math.sin(m))-2*math.cos(4*m)+(math.sin((2*m-math.pi)/24))**5)*math.sin(m))**

punto.size=2

puntos.append(punto)

m=m+math.pi/100

time.sleep(0.01)

Construcción geométrica 2

#definimos una función que dibuja los segmentos entre puntos

def Segmentos():

for i in range(0,len(puntos)-1):

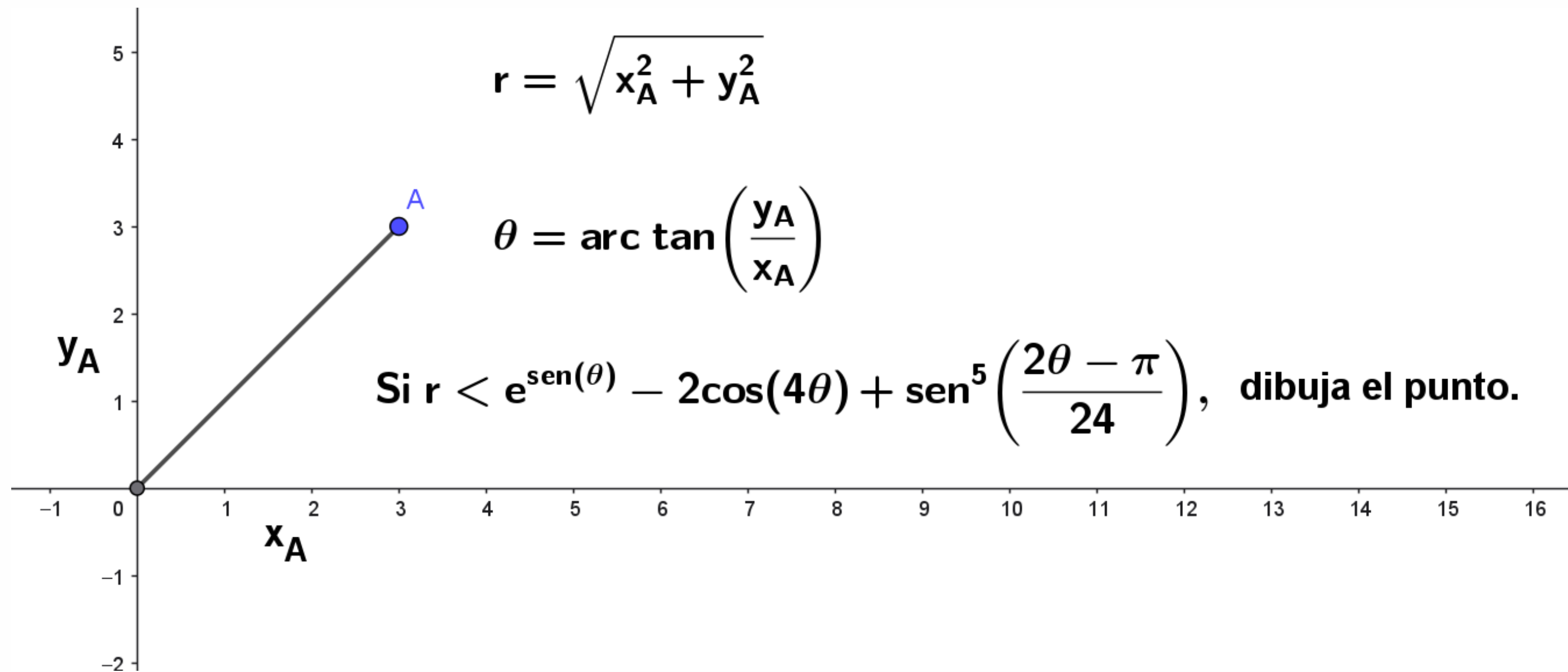
segmento=Segment(puntos[i], puntos[i+1])

segmento.color='red'

segmento.line_thickness=15

time.sleep(0.01)

Construcción geométrica 2



Construcción geométrica 2

#función que dibuja puntos en el interior de las alas superiores

def Interior1():

for i in range(1,20):

for j in range(1,20):

m=math.atan(j/i)

m1=math.pi-m

if math.sqrt((i/5)2+(j/5)**2)< ((math.e)**(math.sin(m))-2*math.cos(4*m)+(math.sin((2*m-math.pi)/24))**5):**

P=Point(i/5,j/5)

P.color=((math.sin(m)),(math.sin(m))4,0.1)**

P.size=6

time.sleep(0.01)

if math.sqrt((i/5)2+(j/5)**2)< ((math.e)**(math.sin(m1))-2*math.cos(4*m1)+(math.sin((2*m1-math.pi)/24))**5):**

P=Point(-i/5,j/5)

P.color=((math.sin(m1)),(math.sin(m1))4,0.1)**

P.size=6

time.sleep(0.01)

Construcción geométrica 2

#función que dibuja puntos en el interior de las alas inferiores

def Interior2():

for l in range(1,20):

for p in range(-20,0):

m2=math.atan(p/l)

m3=math.pi-m2

if math.sqrt((l/5)2+(p/5)**2)< ((math.e)**(math.sin(m2))-2*math.cos(4*m2)+(math.sin((2*m2-math.pi)/24))**5):**

P1=Point(l/5,p/5)

P1.color=(abs((math.sin(m2)))3,abs((math.sin(m2))),0.1)**

P1.size=6

time.sleep(0.01)

if math.sqrt((l/5)2+(p/5)**2)< ((math.e)**(math.sin(m3))-2*math.cos(4*m3)+(math.sin((2*m3-math.pi)/24))**5):**

P1=Point(-l/5,p/5)

P1.color=(abs((math.sin(m3)))3,abs((math.sin(m3))),0.1)**

P1.size=6

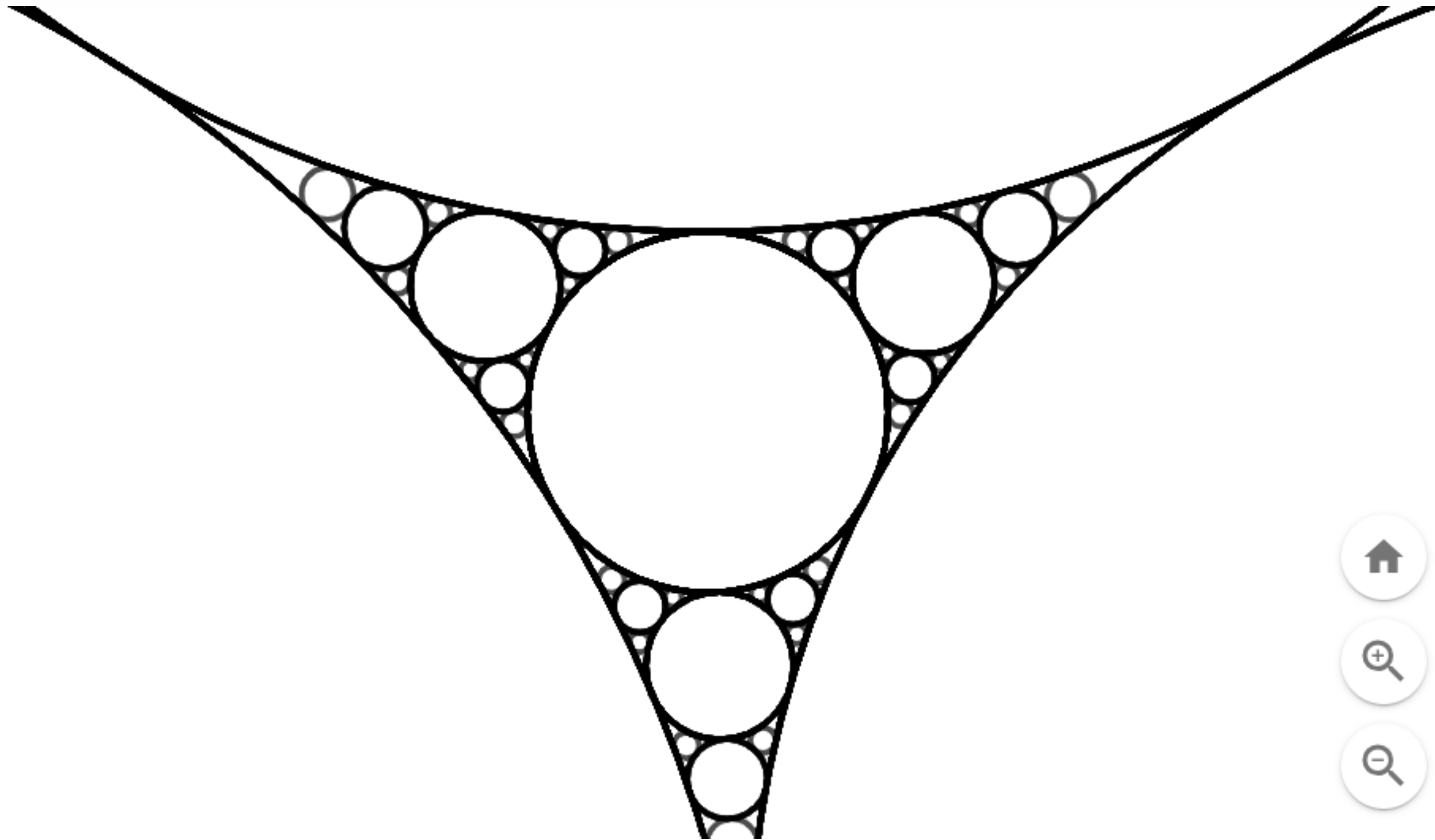
time.sleep(0.01)

Construcción geométrica 2

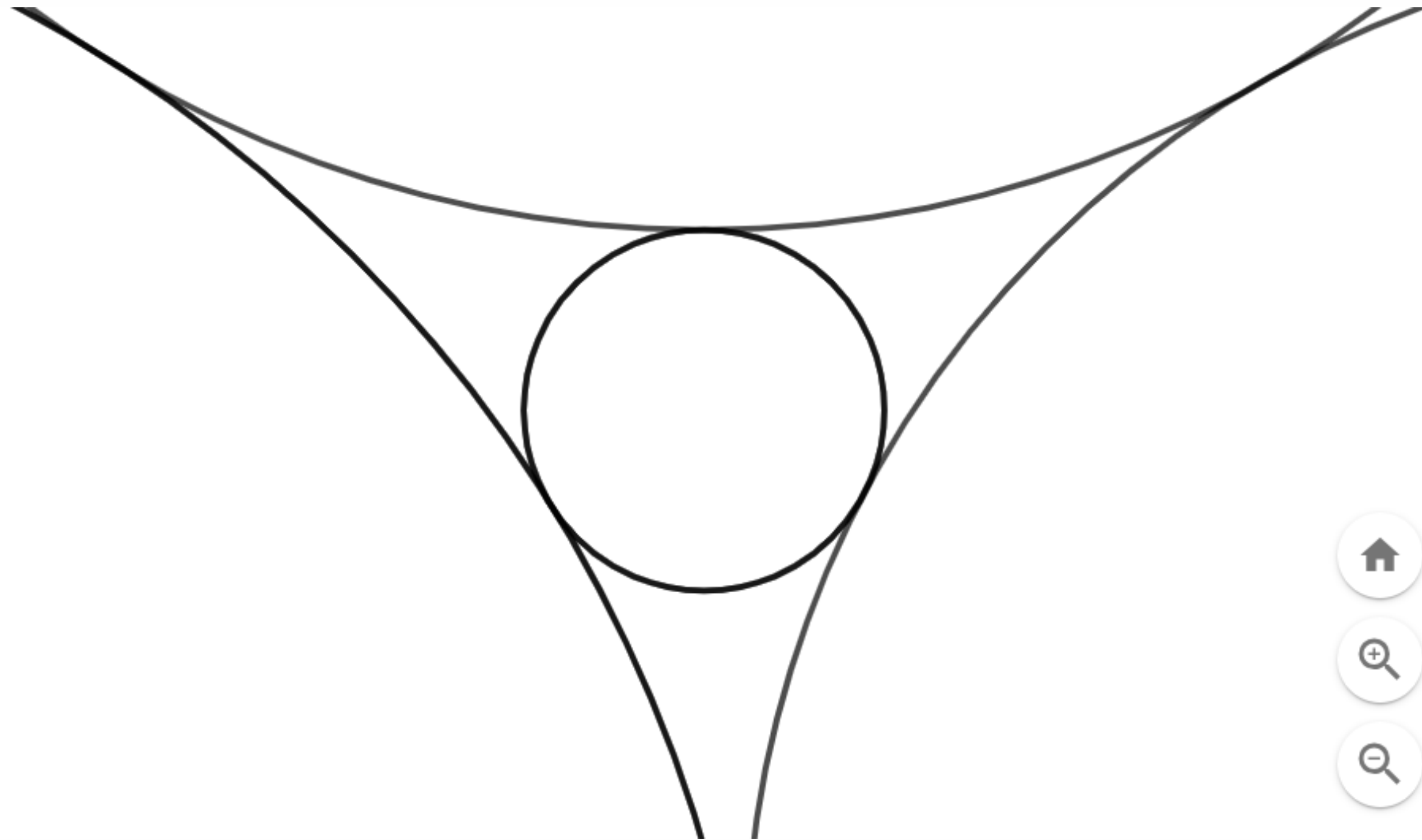
```
#programa principal
#Dibujamos una circunferencia en el fondo
P=Point(0,0,is_visible=False)
C=Circle(P,10)
C.color='black'
C.opacity=1
time.sleep(0.01)

puntos=[]
Puntos()
Segmentos()
Interior1()
Interior2()
```

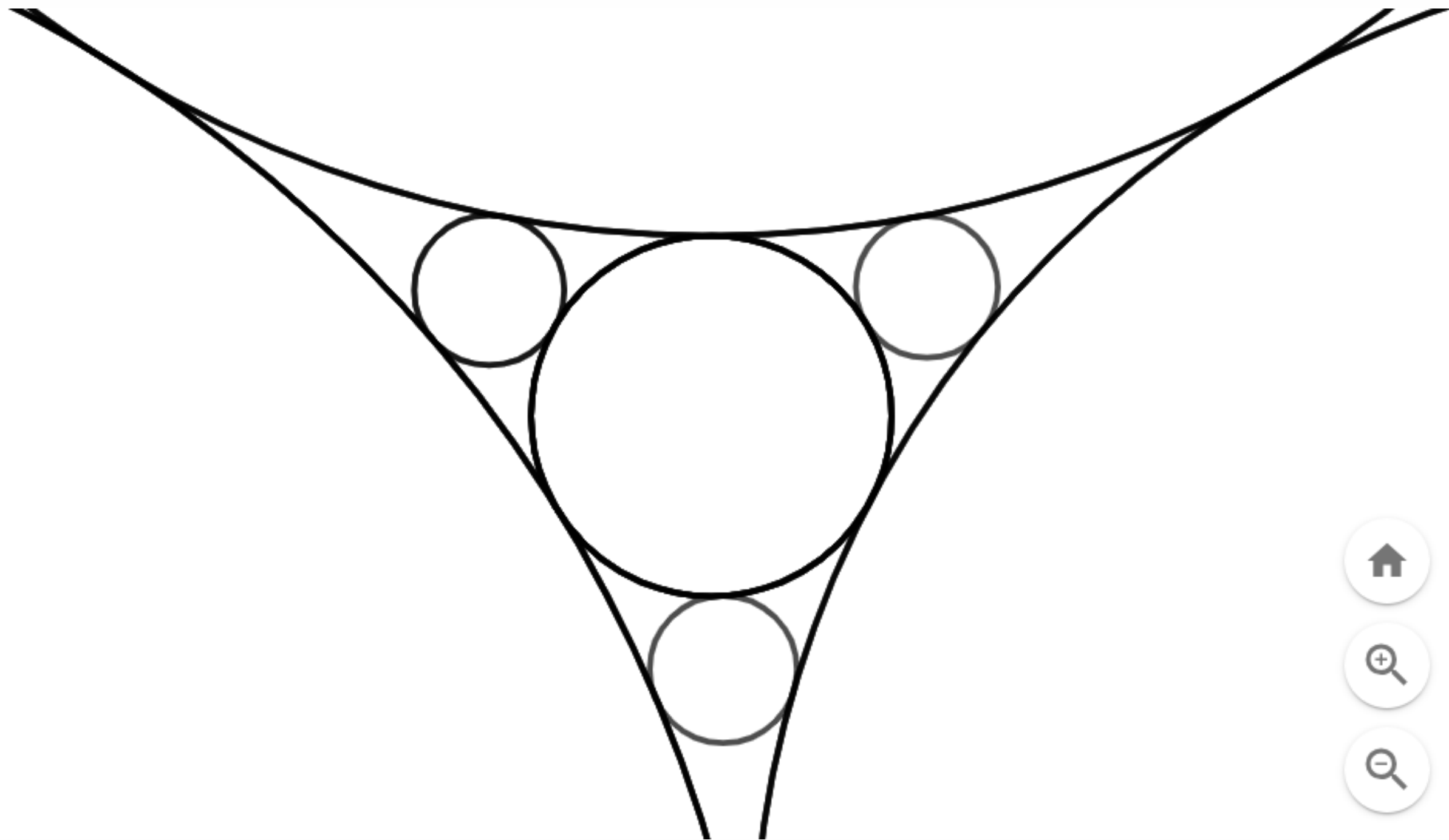
Construcción geométrica 3



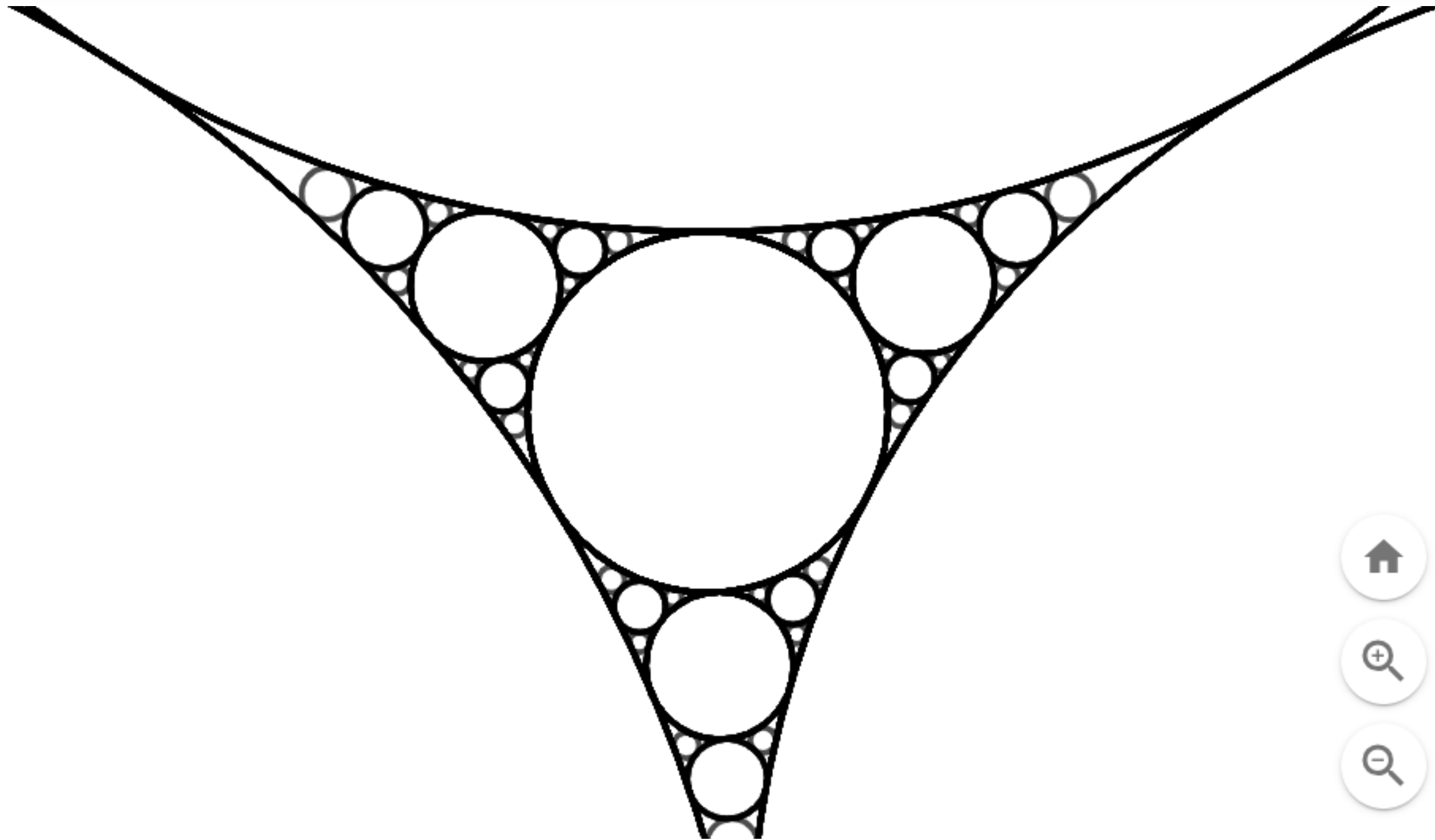
Construcción geométrica 3



Construcción geométrica 3



Construcción geométrica 3



Construcción geométrica 3

```
import time  
import math
```

Construcción geométrica 3

#definimos una función que dibuja la circunferencia tangente a las otras tres
def apolonio(A,B,C):

#calculamos la distancia entre los puntos

a=Distance(B,C)

b=Distance(A,C)

c=Distance(A,B)

#calculamos el radio de las circunferencias centradas en A, B y C

r1=(-a+b+c)/2

r2=(a-b+c)/2

r3=(a+b-c)/2

Construcción geométrica 3

#dibujamos las circunferencias

C1=Circle(A,r1)

C1.opacity=0

time.sleep(0.1)

C2=Circle(B,r2)

C2.opacity=0

time.sleep(0.1)

C3=Circle(C,r3)

C3.opacity=0

time.sleep(0.1)

Construcción geométrica 3

#Dibujamos la circunferencia tangente a las anteriores

#radio de la circunferencia tangente

$$r = \frac{r_1 * r_2 * r_3}{r_1 * r_2 + r_2 * r_3 + r_1 * r_3 + 2 * \text{math.sqrt}(r_1 * r_2 * r_3 * (r_1 + r_2 + r_3))}$$

Construcción geométrica 3

#centro de la circunferencia tangente

$$a0=2*(A.x-B.x)$$

$$a1=2*(A.x-C.x)$$

$$b0=2*(A.y-B.y)$$

$$b1=2*(A.y-C.y)$$

$$c0=2*(r1-r2)$$

$$c1=2*(r1-r3)$$

$$d0=(A.x**2+A.y**2-r1**2)-(B.x**2+B.y**2-r2**2)$$

$$d1=(A.x**2+A.y**2-r1**2)-(C.x**2+C.y**2-r3**2)$$

$$x=(b1*d0-b0*d1-b1*c0*r+b0*c1*r)/(a0*b1-b0*a1)$$

$$y=(-a1*d0+a0*d1+a1*c0*r-a0*c1*r)/(a0*b1-b0*a1)$$

$$S=Point(x,y,is_visible=False)$$

Construcción geométrica 3

#circunferencia tangente, centro S y radio r

C4=Circle(S,r)

C4.opacity=0

time.sleep(0.1)

lista.append(S)

time.sleep(1)

Construcción geométrica 3

#definimos una función que dibuja tres circunferencias tangentes

def apolonioapolonio(A,B,C,D):

apolonio(D,A,B)

apolonio(D,B,C)

apolonio(D,C,A)

Construcción geométrica 3

```
#programa principal  
#informamos en que consiste la actividad  
print('Introduce las coordenadas de tres puntos: A, B y C.')  
time.sleep(3)
```

Construcción geométrica 3

#Dibujamos una esfera para que el fondo sea blanco

P=Point(0,0,is_visible=False)

C=Circle(P,10)

C.color='white'

C.opacity=1

time.sleep(0.1)

Construcción geométrica 3

#introducimos las coordenadas de los puntos

xA= int(input('Coordenada x de A:'))

time.sleep(1)

yA= int(input('Coordenada y de A:'))

time.sleep(1)

xB= int(input('Coordenada x de B:'))

time.sleep(1)

yB= int(input('Coordenada y de B:'))

time.sleep(1)

xC= int(input('Coordenada x de C:'))

time.sleep(1)

yC= int(input('Coordenada y de C:'))

time.sleep(1)

Construcción geométrica 3

#definimos los puntos y aplicamos las funciones

A=Point(xA,yA,is_visible=False)

B=Point(xB,yB,is_visible=False)

C=Point(xC,yC,is_visible=False)

lista=[A,B,C]

apolonio(lista[0],lista[1],lista[2])

Construcción geométrica 3

apolonioapolonio(lista[0],lista[1],lista[2],lista[3])

apolonioapolonio(lista[3],lista[0],lista[1],lista[4])

apolonioapolonio(lista[3],lista[1],lista[2],lista[5])

apolonioapolonio(lista[3],lista[2],lista[0],lista[6])

apolonioapolonio(lista[4],lista[3],lista[0],lista[7])

apolonioapolonio(lista[4],lista[0],lista[1],lista[8])

apolonioapolonio(lista[4],lista[3],lista[1],lista[9])

Construcción geométrica 3

apolonioapolonio(lista[5],lista[3],lista[1],lista[10])

apolonioapolonio(lista[5],lista[1],lista[2],lista[11])

apolonioapolonio(lista[5],lista[2],lista[3],lista[12])

apolonioapolonio(lista[6],lista[3],lista[2],lista[13])

apolonioapolonio(lista[6],lista[2],lista[0],lista[14])

apolonioapolonio(lista[6],lista[0],lista[3],lista[15])

Bibliografía

Cook, T.A. (1979). *The curves of life*. Dover Publications.

es.python.org/aprende-python/ (Último acceso 6 de octubre de 2025)

Ibáñez, R. (2023). *Las matemáticas como herramienta de creación artística*. Catarata Editorial.

Kaliaguin, I.M., Sarkisión, A.A. (2010). *Breve introducción a la topología*. Editorial URSS.

Ferréol, R. (s.f.). <https://mathcurve.com> (Último acceso 6 de octubre de 2025)

