Inicio

Início\_Classes

de matemática, importe raíz quadrada, número Pi

classe Ponto:

função Parâmetros(molde, x, y):

molde.x = x

molde.y = y

molde.CoordenadasDoPonto = (molde.x, ,molde.y)

classe Reta:

função Parâmetros(molde, ax, ay, bx, by):

molde.ax = ax

molde.ay = ay

molde.bx = bx

molde.by = by

função EqdaReta(molde):

Coeficiente Angular = (molde.by – molde.ay) / (molde.bx - molde.ax)

Coeficiente Linear = molde.ay – (Coeficente Angular) \* molde.ax

Equação da Reta = formatado"{Coeficiente Angular} \* x + {Coeficiente Linear}"

retorne Equação da Reta

função Distancia(molde):

distancia = raiz quadrada((molde.bx - molde.ax)\*\*2 + (molde.by - molde.ay)\*\*2)

retorne distancia

classe Quadrado(Retangulo):

função Parâmetros(molde, b):

super().Parâmetros(molde, b)

molde.base = b

função Area(molde):

area = molde.base\*molde.base

mostre(formatado’Área: {molde.base}² = {area}.’)

retorne

função Diagonal(molde):

diagonal = raíz quadrada(((molde.base)\*\*2)+(molde.base)\*\*2)

mostre(formatado’Diagonal: raíz de 2 \* lado = raíz de 2 \* {molde.base} =

{diagonal:.2flutuantes}.’)

retorne

função Perimetro(molde):

perimet = 4\*(molde.base)

print(f'Perímetro: 4 x {molde.base} = {perimet}.')

classe Circulo:

função Parâmetros(molde, raio, centro\_x, centro\_y):

molde.centro\_x = centro\_x

molde.centro\_y = centro\_y

molde.raio = raio

função TesteDoRaio(self, ponto\_x, ponto\_y):

distancia = raíz quadrada((ponto\_x - self.centro\_x)\*\*2 + (ponto\_y -

self.centro\_y)\*\*2)

se distancia > molde.raio:

mostre("O ponto está fora do círculo.")

senão se distancia == molde.raio:

mostre("O ponto está em cima da circunferência do círculo.")

senão:

mostre("O ponto está dentro do círculo.")

area = pi \* (molde.Raio)\*\*2

mostre(formatado'Área: π x r² = {pi} x {molde.Raio}² = {area}.')

classe Triangulo:

função Parâmetros(molde, ax, ay, bx, by, cx, cy):

molde.ax = ax

molde.ay = ay

molde.bx = bx

molde.by = by

molde.cx = cx

molde.cy = cy

função LadosDoTriangulo(molde):

lado1 = raíz quadrada((molde.ax - molde.bx)\*\*2 + (molde.ay - molde.by)\*\*2)

lado2 = raíz quadrada((molde.bx - molde.cx)\*\*2 + (molde.by - molde.cy)\*\*2)

lado3 = raíz quadrada((molde.cx - molde.ax)\*\*2 + (molde.cy - molde.ay)\*\*2)

l = [lado1, lado2, lado3]

mostre(formatado'''

Distancia do primeiro ponto até o segundo = {lado1}

Distancia do segundo ponto até o terceiro = {lado2}

Distancia do terceiro ponto até o primeiro = {lado3}''')

função Area(molde):

lado1 = raíz quadrada((molde.ax - molde.bx)\*\*2 + (molde.ay - molde.by)\*\*2)

lado2 = raíz quadrada((molde.bx - molde.cx)\*\*2 + (molde.by - molde.cy)\*\*2)

lado3 = raíz quadrada((molde.cx - molde.ax)\*\*2 + (molde.cy - molde.ay)\*\*2)

l = [lado1, lado2, lado3]

se (lado1 + lado2 > lado3) e (lado1 + lado3 > lado2) e (lado2 + lado3 > lado1):

mostre('Elas conseguem formar um triâgulo.')

se lado1 igual a lado2 igual a lado3:

mostre('E esse triângulo é equilatero.')

area = ((lado1)\*\*2)\*(raíz quadrada(3))/4

mostre(formatado'Área do triângulo equilatero: {area}')

senão se lado1 igual a lado2 ou lado1 igual a lado3 ou lado2 igual a lado3:

mostre('E esse triângulo é isóceles.')

maior = [ ]

maior.adicionar(número máximo(l))

h = raíz quadrada(lado1\*\*2 + (maior/2)\*\*2)

area = (maior \* h)/2

mostre(formatado'Área do triângulo isóceles: {area}')

senão se lado1 diferente de lado2 diferente de lado3:

mostre('E esse triângulo é escaleno.')

#Fórmula de Heron para calcular a área do triângulo escaleno.

semiperimetro = (lado1 + lado2 + lado3)/2

area = raíz quadrada(semiperimetro\*(semiperimetro -

lado1)\*(semiperimetro-lado2)\*(semiperimetro-lado3))

mostre(formatado'Área do triângulo escaleno: {area}')

senão:

mostre('Não conseguem formar um triângulo.')

Fim\_Classes

Início\_Main

de reta, importe Reta

l = [ ]

para c no intervalo(0, 4):

p = inteiro(entrada('Digite um número que complete as coordenadas A(Xa, Ya) eB(Xb, Yb):

'))

l.adicione(p)

r1 = Reta(l[0], l[1], l[2], l[3])

r1.Distancia()

r1.EquacaodaReta()

de quadrado, importe Quadrado

#Diferentemente do retângulo, o quadrado não tem base e altura diferentes. Portanto, basta

saber um dos dados para aplicar as funções.

lado = inteiro(entrada('Qual o lado do quadrado? '))

l1 = Quadrado(lado)

l1.Area()

l1.Perimetro()

l1.Diagonal()

de circulo, importe Circulo

c1 = Circulo(5, 0, 0)

c1.TesteDoRaio(3, 4)

c1.Area()

de triangulo, importe Triangulo

#O init está na respectiva ordem: ax, ay, bx, by, cx, cy.

x = [ ]

para c no intervalo(0, 6):

p = inteiro(entrada('Digite um número que complete as coordenadas A(x, y), B(x, y) e C(x,

y): '))

x.adicionar(p)

t1 = Triangulo(x[0], x[1], x[2], x[3], x[4], x[5])

t1.LadosDoTriangulo()

t1.Area()

de retangulo importe Retangulo

b2 = inteiro(entrada('Qual a base do Retângulo? '))

h2 = inteiro(entrada('Qual a altura do Retângulo? '))

rg = Retangulo(b2, h2)

rg.Area()

rg.Diagonal()

Fim\_Main

Fim