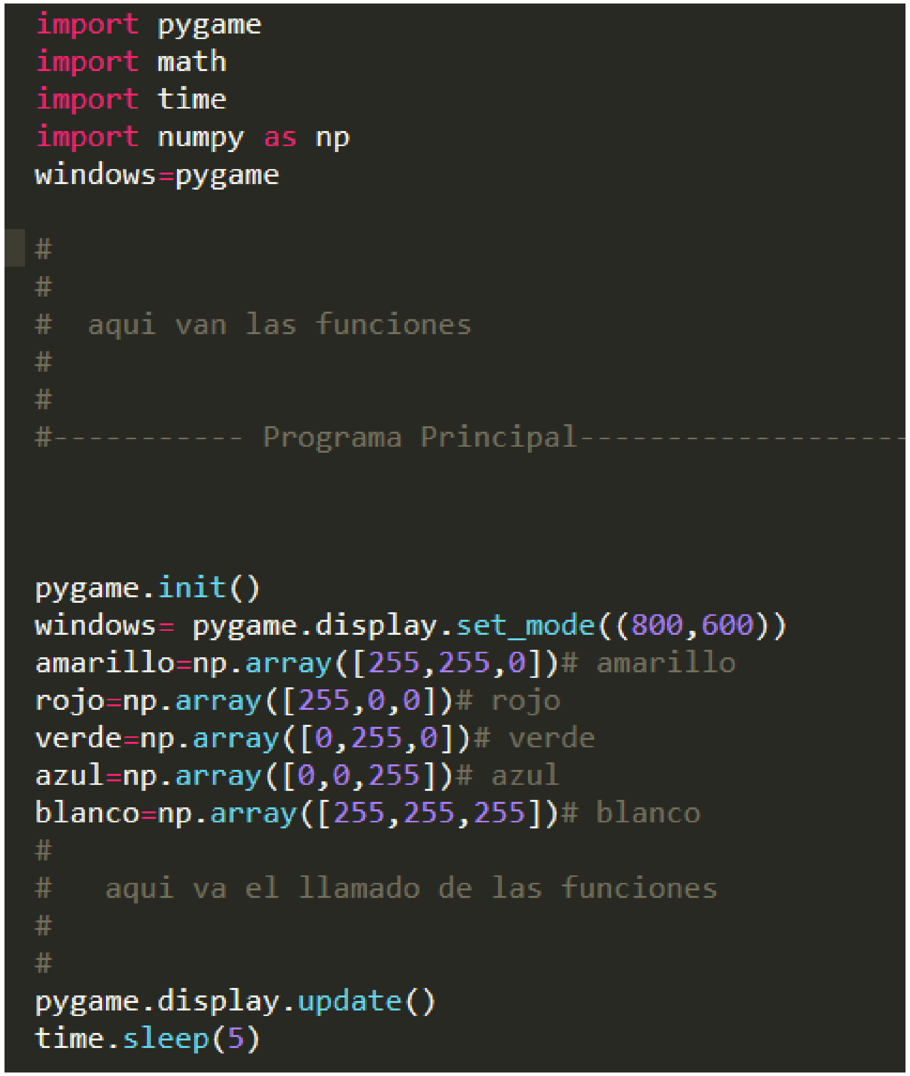
Taller de Primitivas Graficas

Cuerpo del programa



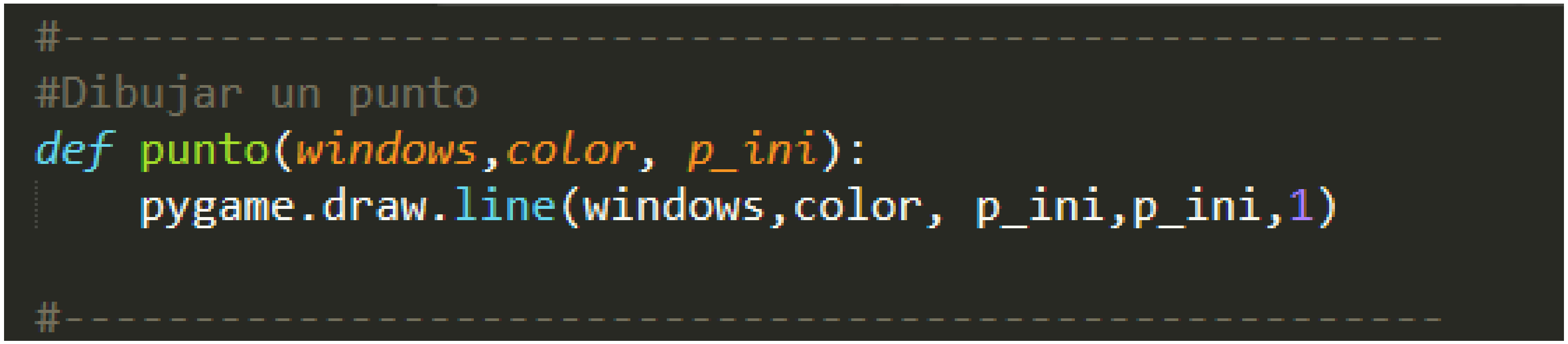
```
import pygame
import math
import time
import numpy as np
windows=pygame

#
#
#  aqui van las funciones
#
#
#----------- Programa Principal------------------



pygame.init()
windows= pygame.display.set_mode((800,600))
amarillo=np.array([255,255,0])# amarillo
rojo=np.array([255,0,0])# rojo
verde=np.array([0,255,0])# verde
azul=np.array([0,0,255])# azul
blanco=np.array([255,255,255])# blanco
#
#   aqui va el llamado de las funciones
#
#
pygame.display.update()
time.sleep(5)
```

Función Punto

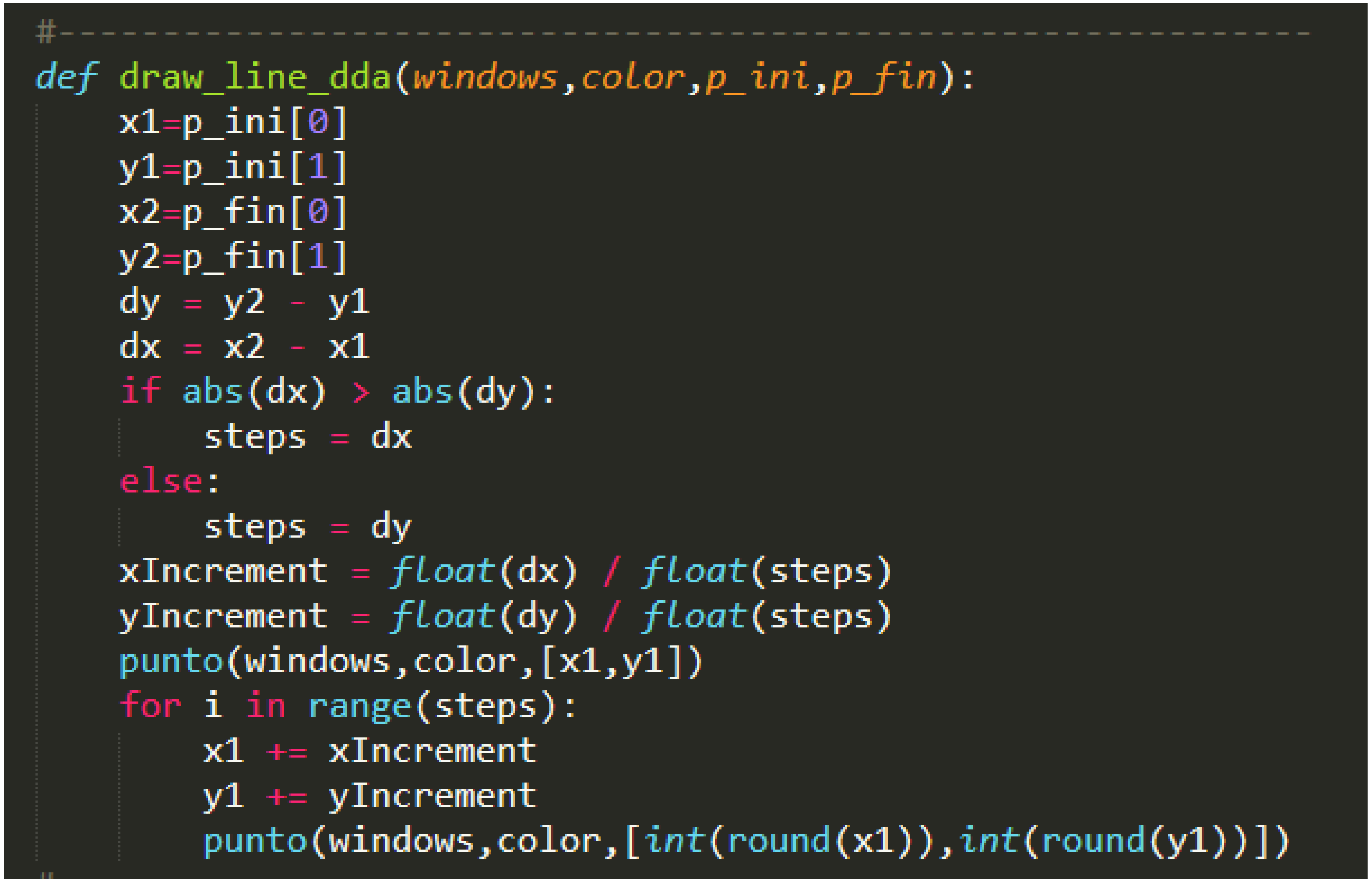


```
#-----------------------------------------------------------------
#Dibujar un punto
def punto(windows,color, p_ini):
    pygame.draw.line(windows,color, p_ini,p_ini,1)

#-----------------------------------------------------------------
```

punto(windows,azul,[200,200])

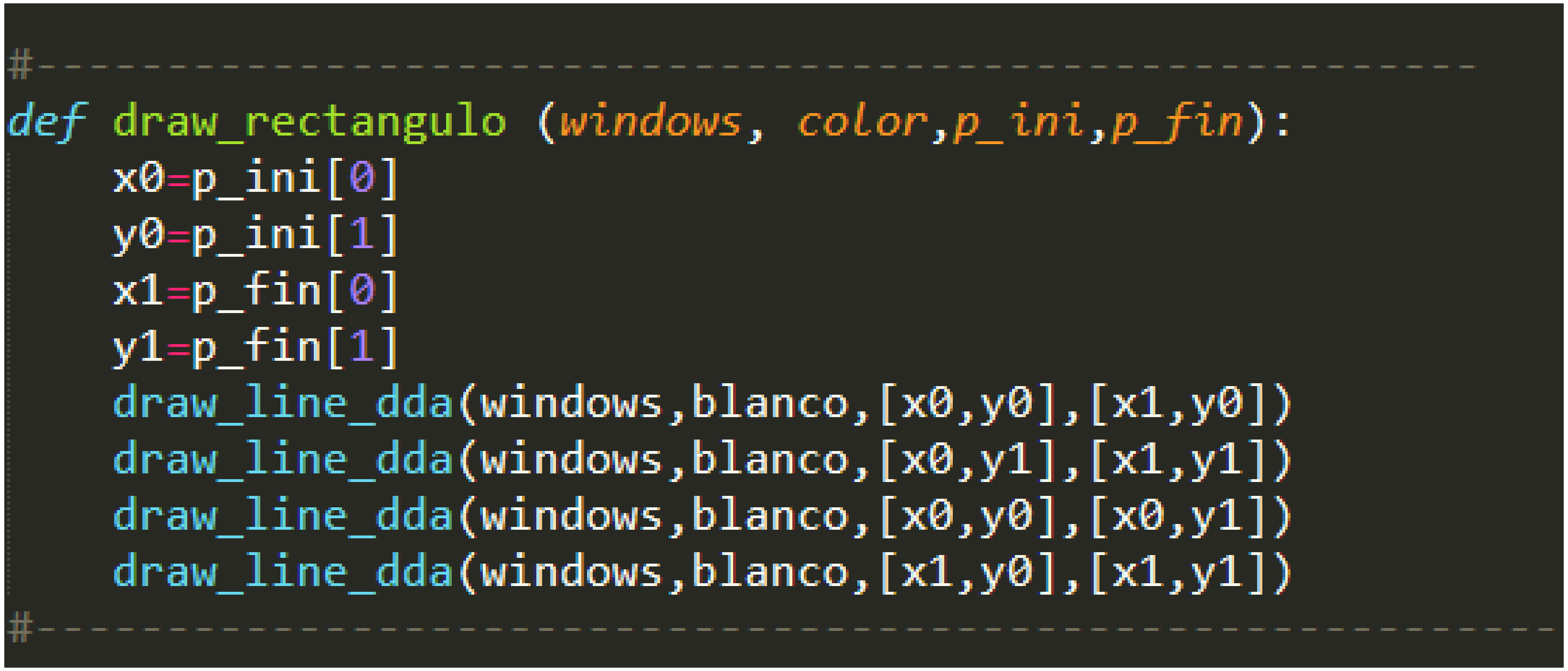
Función Línea (Algoritmo DDA)



```
#-----------------------------------------------------------------------------
def draw_line_dda(windows,color,p_ini,p_fin):
    x1=p_ini[0]
    y1=p_ini[1]
    x2=p_fin[0]
    y2=p_fin[1]
    dy = y2 - y1
    dx = x2 - x1
    if abs(dx) > abs(dy):
        steps = dx
    else:
        steps = dy
    xIncrement = float(dx) / float(steps)
    yIncrement = float(dy) / float(steps)
    punto(windows,color,[x1,y1])
    for i in range(steps):
        x1 += xIncrement
        y1 += yIncrement
        punto(windows,color,[int(round(x1)),int(round(y1))])
```

draw_line_dda(windows,azul,[0,300],[800,300])

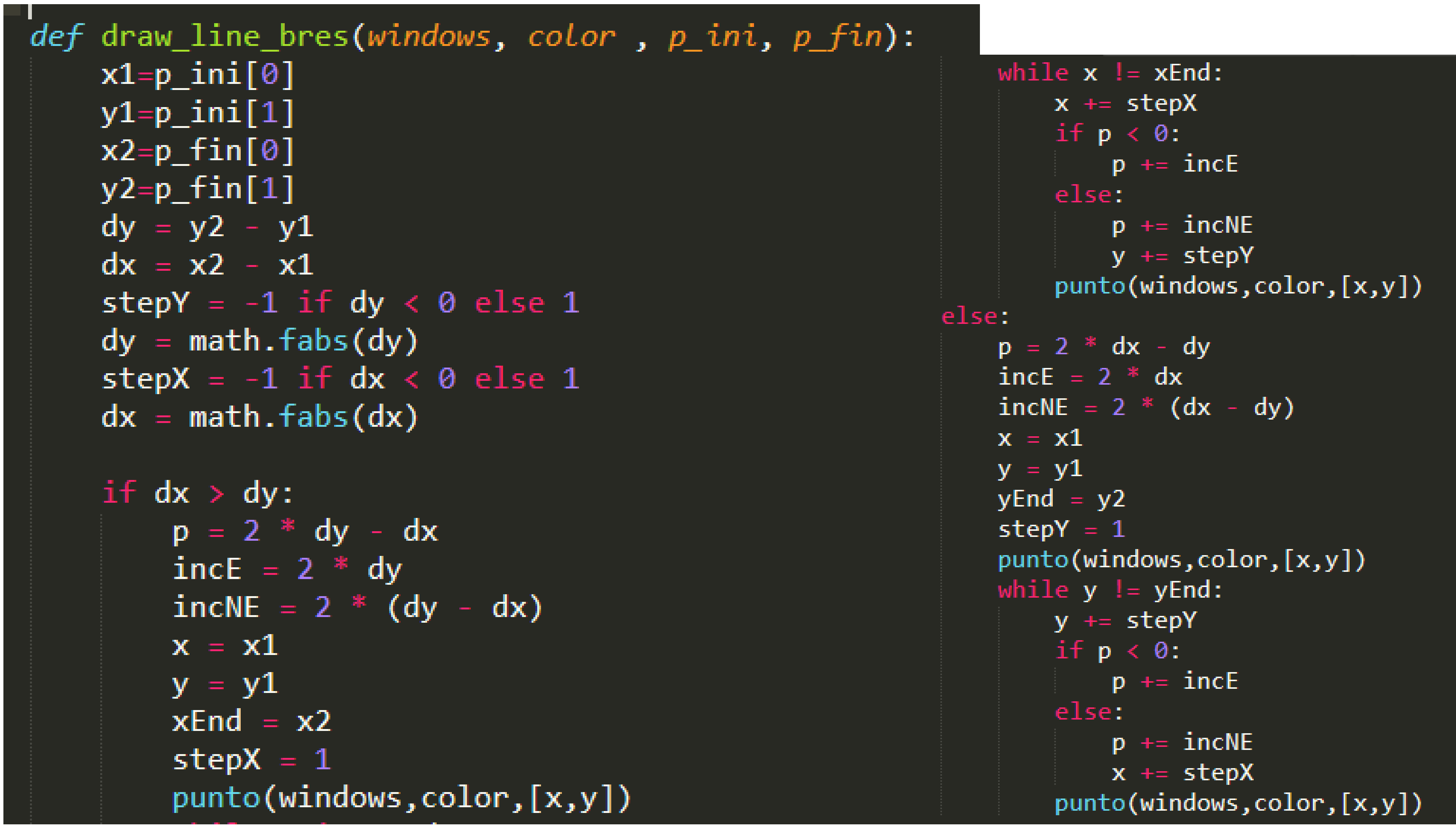
Función Rectángulo (basado en DDA)



```
#---------------------------------------------------------------
def draw_rectangulo (windows, color,p_ini,p_fin):
    x0=p_ini[0]
    y0=p_ini[1]
    x1=p_fin[0]
    y1=p_fin[1]
    draw_line_dda(windows,blanco,[x0,y0],[x1,y0])
    draw_line_dda(windows,blanco,[x0,y1],[x1,y1])
    draw_line_dda(windows,blanco,[x0,y0],[x0,y1])
    draw_line_dda(windows,blanco,[x1,y0],[x1,y1])
#------------------------------------------------------------------
```

draw_rectangulo(windows,verde,[50,50],[350,250])

Función línea (Algoritmo de Bresenham)



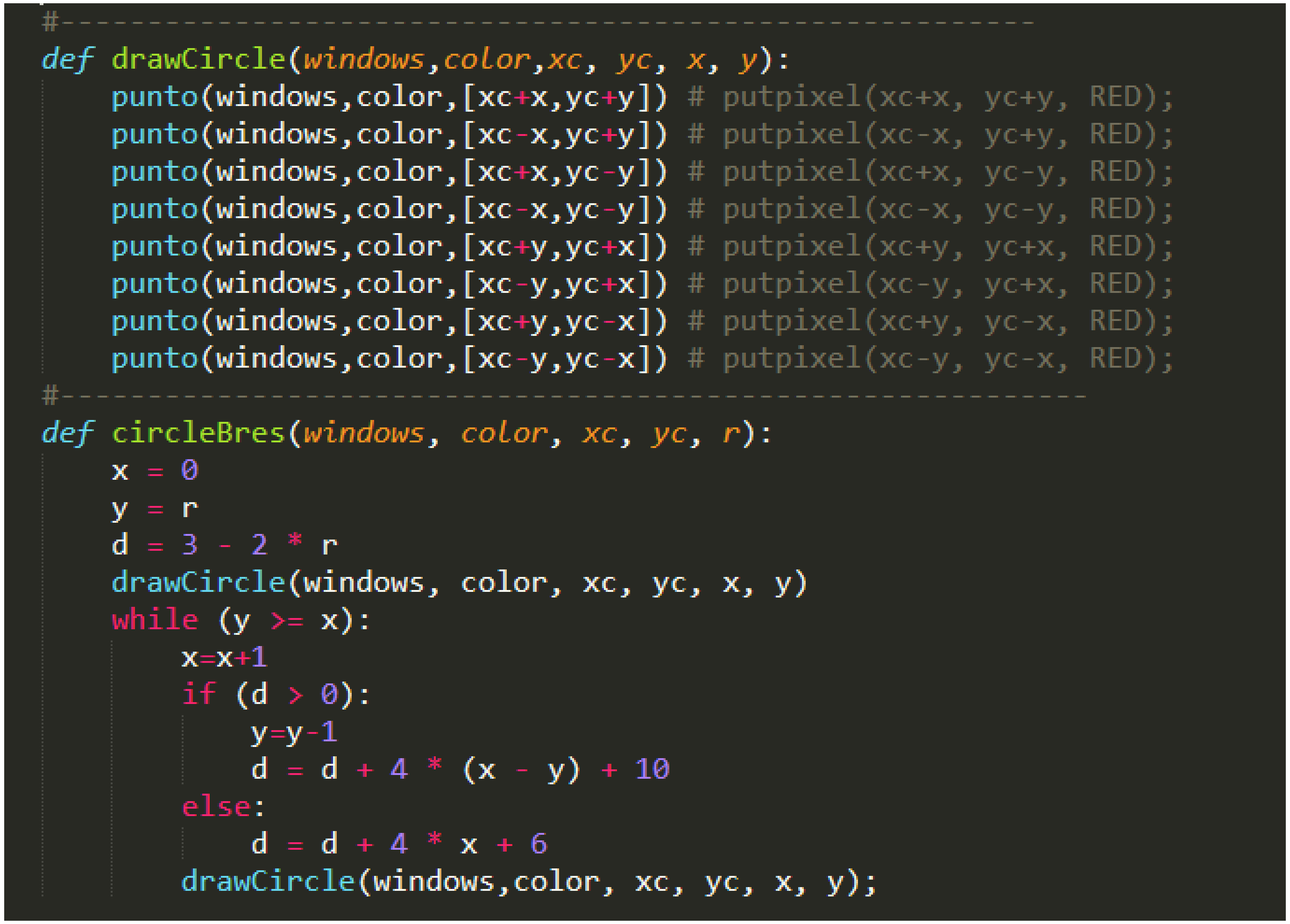
```python
def draw_line_bres(windows, color , p_ini, p_fin):
    x1=p_ini[0]
    y1=p_ini[1]
    x2=p_fin[0]
    y2=p_fin[1]
    dy = y2 - y1
    dx = x2 - x1
    stepY = -1 if dy < 0 else 1
    dy = math.fabs(dy)
    stepX = -1 if dx < 0 else 1
    dx = math.fabs(dx)

    if dx > dy:
        p = 2 * dy - dx
        incE = 2 * dy
        incNE = 2 * (dy - dx)
        x = x1
        y = y1
        xEnd = x2
        stepX = 1
        punto(windows,color,[x,y])
```

```python
        while x != xEnd:
            x += stepX
            if p < 0:
                p += incE
            else:
                p += incNE
                y += stepY
            punto(windows,color,[x,y])
    else:
        p = 2 * dx - dy
        incE = 2 * dx
        incNE = 2 * (dx - dy)
        x = x1
        y = y1
        yEnd = y2
        stepY = 1
        punto(windows,color,[x,y])
        while y != yEnd:
            y += stepY
            if p < 0:
                p += incE
            else:
                p += incNE
                x += stepX
            punto(windows,color,[x,y])
```

draw_line_bres(windows,verde,[0,300],[800,300])

Función Circulo Algoritmo de Bresenham



```
#------------------------------------------------------
def drawCircle(windows,color,xc, yc, x, y):
    punto(windows,color,[xc+x,yc+y]) # putpixel(xc+x, yc+y, RED);
    punto(windows,color,[xc-x,yc+y]) # putpixel(xc-x, yc+y, RED);
    punto(windows,color,[xc+x,yc-y]) # putpixel(xc+x, yc-y, RED);
    punto(windows,color,[xc-x,yc-y]) # putpixel(xc-x, yc-y, RED);
    punto(windows,color,[xc+y,yc+x]) # putpixel(xc+y, yc+x, RED);
    punto(windows,color,[xc-y,yc+x]) # putpixel(xc-y, yc+x, RED);
    punto(windows,color,[xc+y,yc-x]) # putpixel(xc+y, yc-x, RED);
    punto(windows,color,[xc-y,yc-x]) # putpixel(xc-y, yc-x, RED);
#---------------------------------------------------------
def circleBres(windows, color, xc, yc, r):
    x = 0
    y = r
    d = 3 - 2 * r
    drawCircle(windows, color, xc, yc, x, y)
    while (y >= x):
        x=x+1
        if (d > 0):
            y=y-1
            d = d + 4 * (x - y) + 10
        else:
            d = d + 4 * x + 6
        drawCircle(windows,color, xc, yc, x, y);
```

circleBres(windows, rojo, 400, 500, 100)

Figura 1

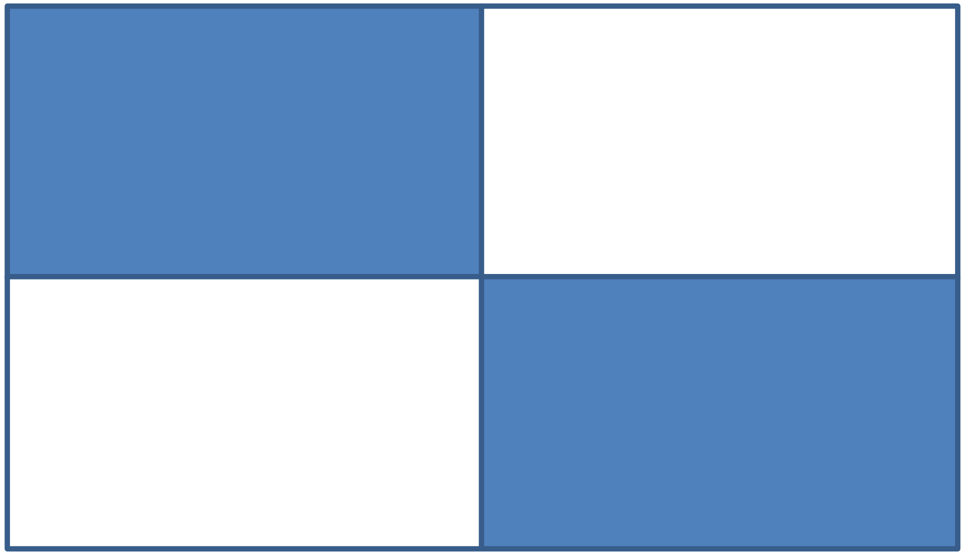


Figura 2

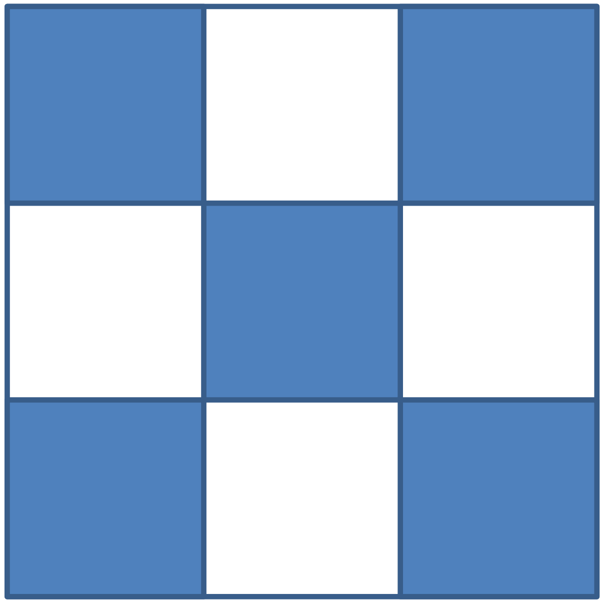


Figura 3

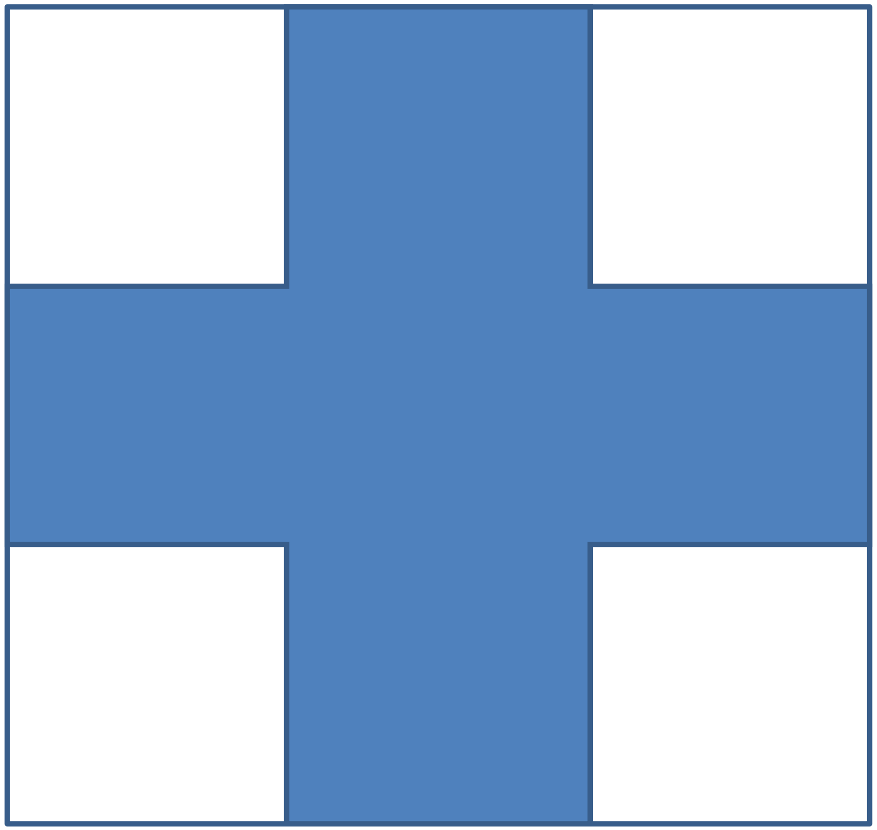


Figura 4

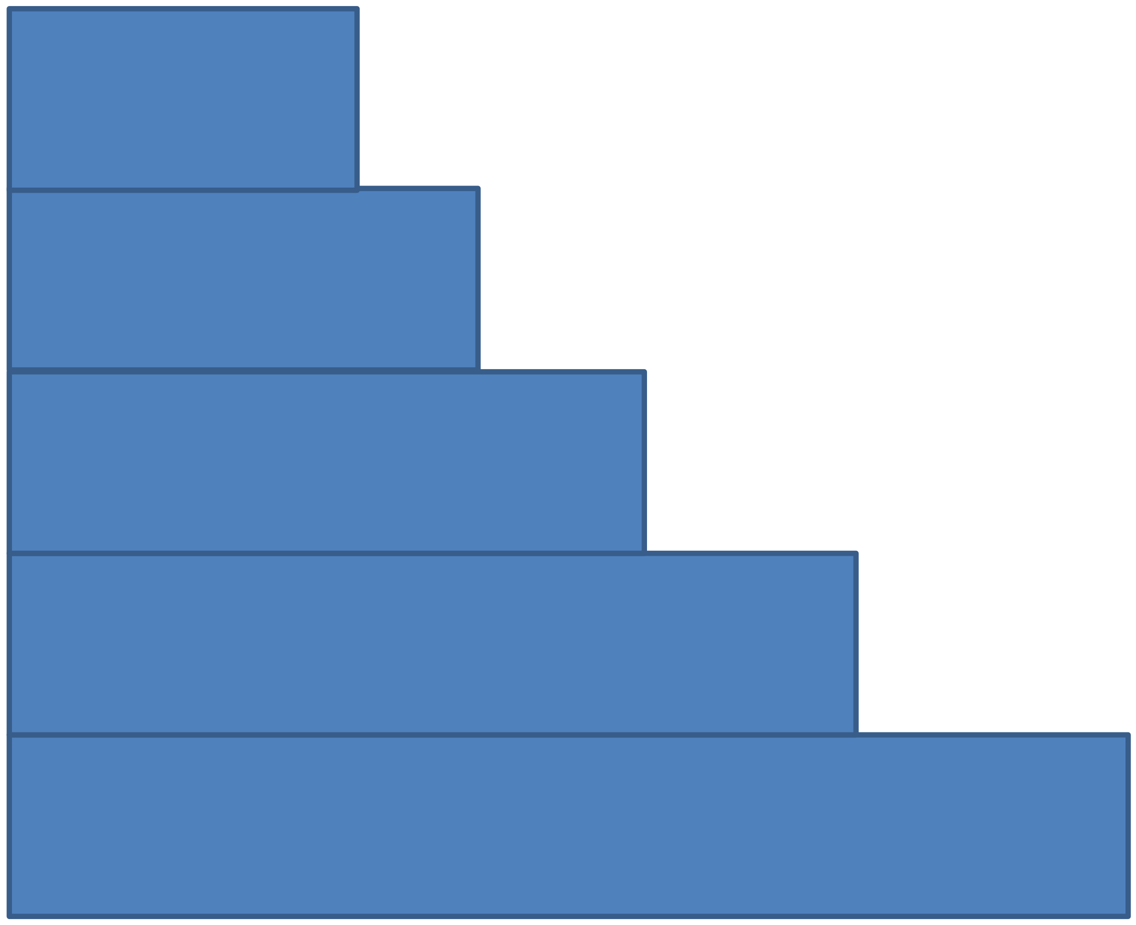


Figura 5

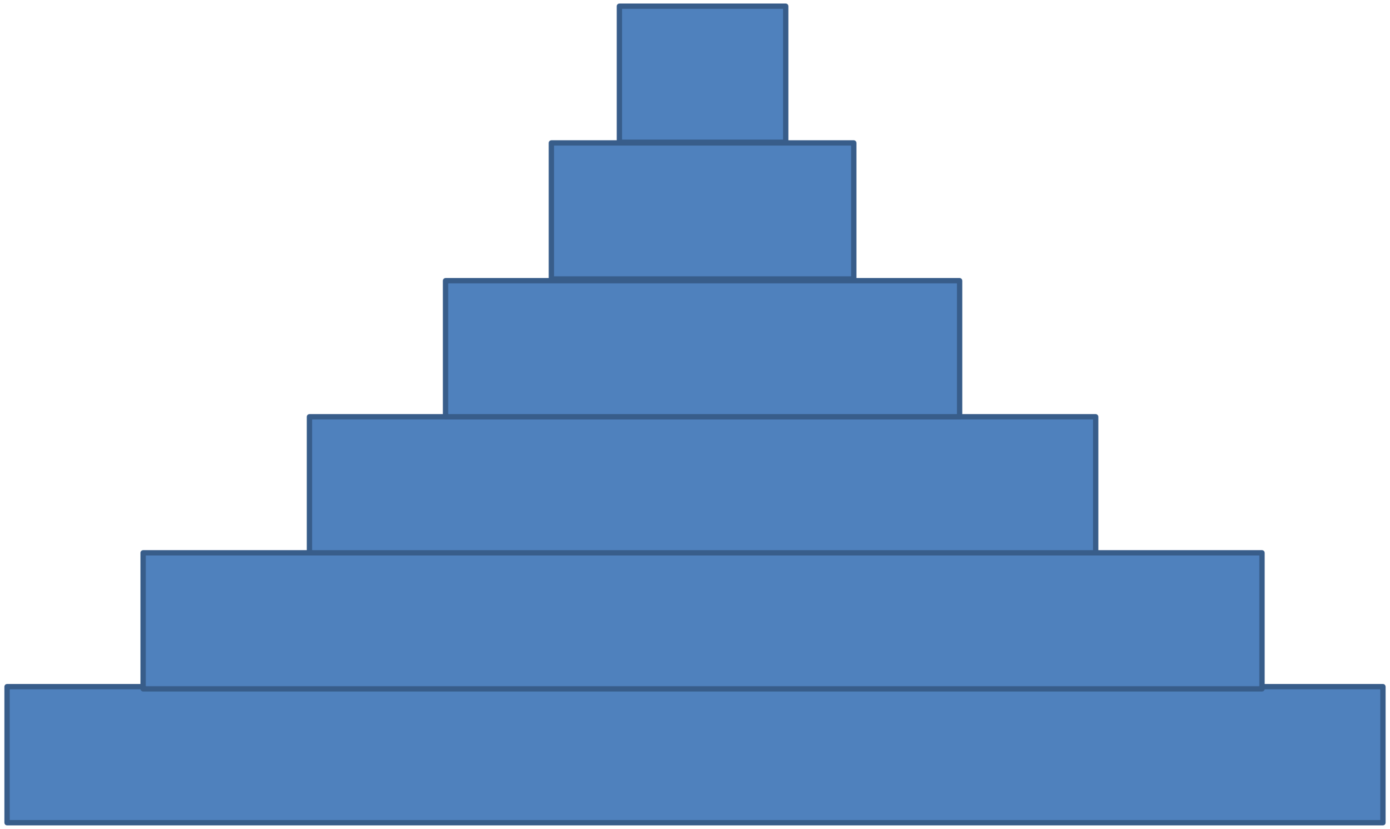


Figura 6

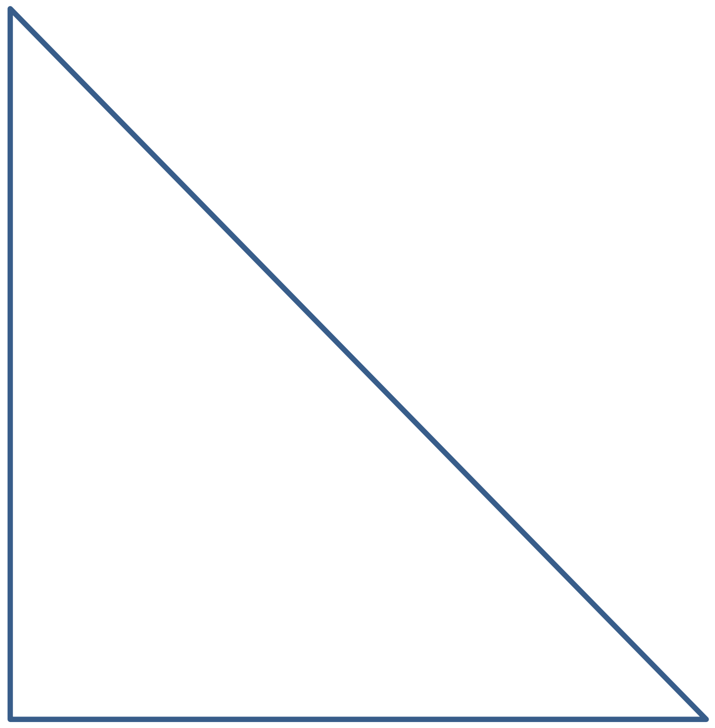


Figura 7

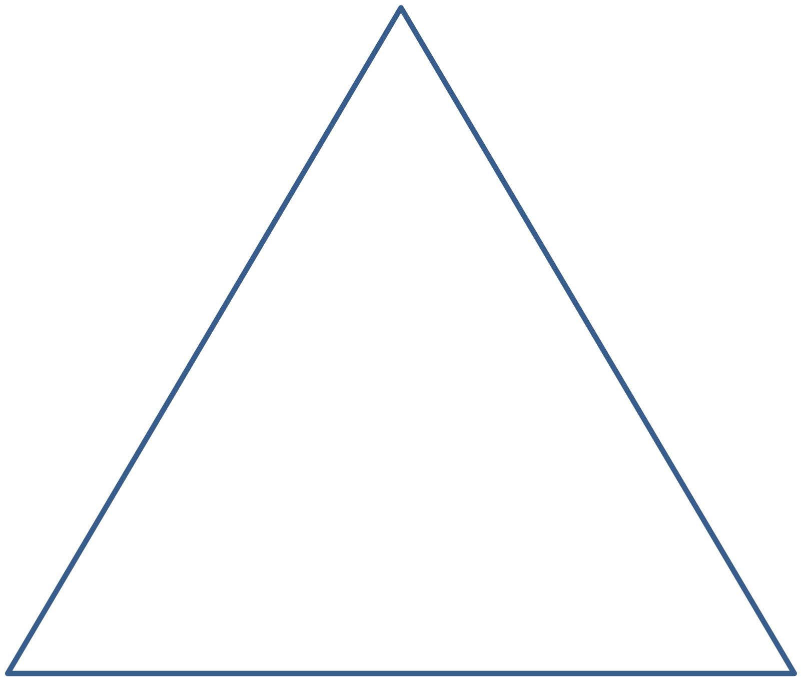


Figura 8

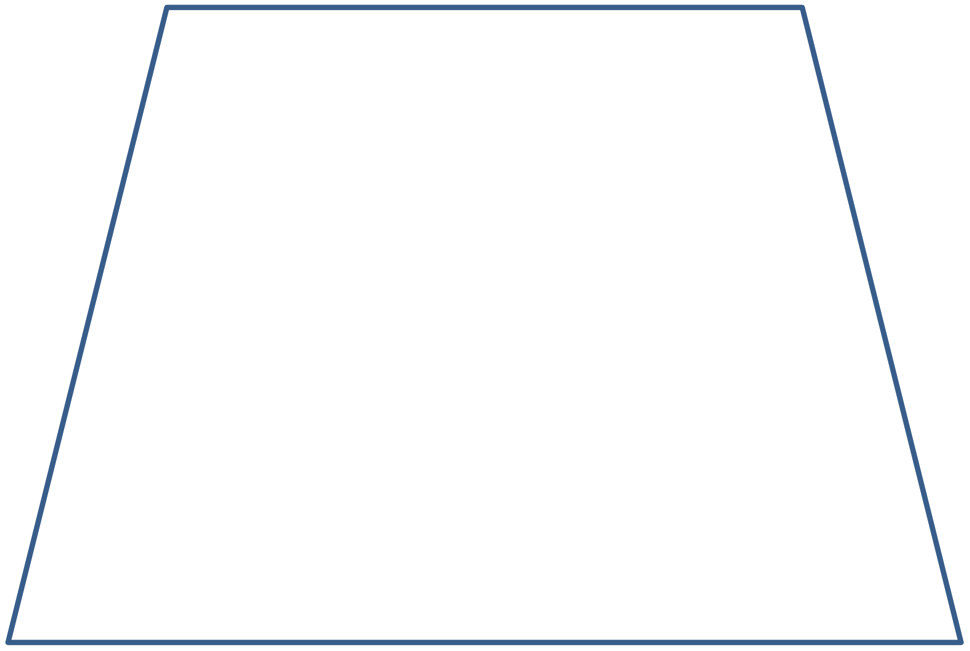


Figura 9

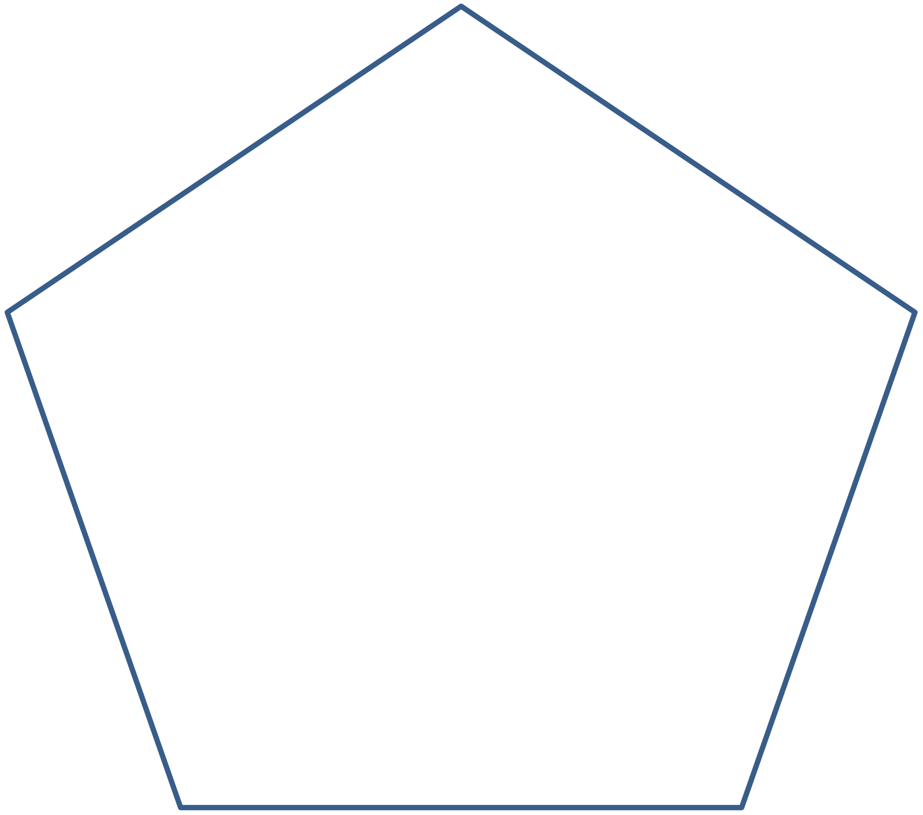


Figura 10