1.3. Generación de Figuras Geométricas Básicas en Python.

En el diseño técnico, las figuras geométricas básicas (puntos, segmentos, polígonos y circunferencias) son los elementos fundamentales a partir de los cuales se construyen piezas más complejas.
En este módulo aprenderemos a **programar y visualizar** estas figuras en Python, utilizando librerías de cálculo y trazado.

Herramientas que vamos a usar:

- **numPy** → para generar coordenadas y realizar cálculos.
- **matplotlib** → para representar las figuras en pantalla.
- (Opcional) **ezdxf** → para exportar los resultados a formato DXF y abrirlos en AutoCAD.

1.3.1. Representación de un punto

Un punto se representa como un par ordenado (x, y) en el plano.

```
import matplotlib.pyplot as plt

# Coordenadas del punto
x = [3]
y = [4]

plt.figure()
plt.plot(x, y, 'ro', label="Punto (3,4)") # 'ro' = red circle marker
plt.axis('equal')
plt.legend()
plt.title("Representación de un punto")
plt.show()
```

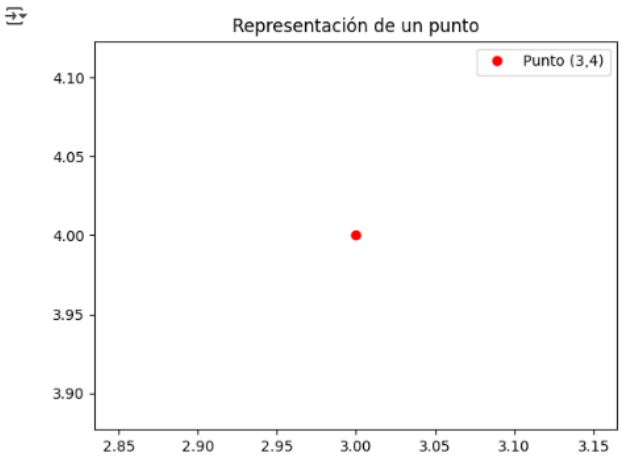

Figura 22. Salida esperada de un punto.

Ejercicio Propuesto 1.3.1.1.

- Cambiar los datos en las instrucciones para que el punto tenga otra posición.
- Agregar varios puntos en la misma gráfica.

1.3.2. Dibujo de un segmento

Un segmento se define por dos puntos (x1, y1) y (x2, y2).

```
x = [1, 5]
y = [2, 6]

plt.figure()
plt.plot(x, y, 'b-', linewidth=2, label="Segmento AB")
plt.scatter(x, y, color='red') # puntos extremos
plt.axis('equal')
plt.legend()
plt.title("Segmento AB")
plt.show()
```

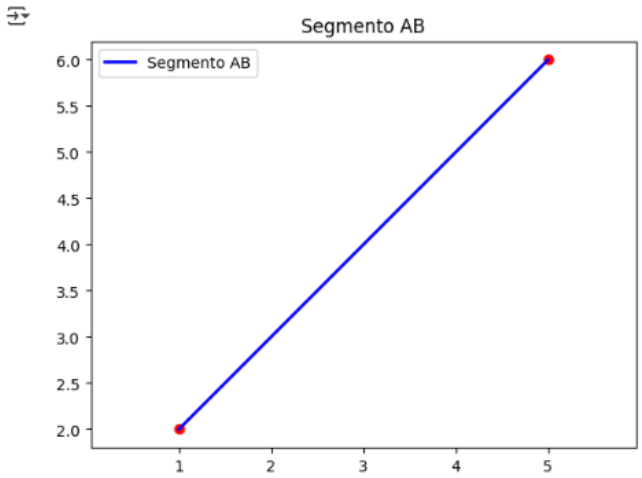

Figura 23. Salida esperada de un segmento.

Ejercicio Propuesto 1.3.2.1.

- Cambiar los datos en las instrucciones para que el segmento sea diferente.
- Usa el script del Ejercico Propuesto 1.3.1.1. y unir los puntos.

1.3.3. Creación de polígonos

Un polígono se genera uniendo varios puntos en orden y cerrando la figura repitiendo el primero al final.

Ejemplo: **Pentágono regular.**

```
import numpy as np

# Número de lados y radio
n = 5
r = 4
angulos = np.linspace(0, 2*np.pi, n+1)

x = r * np.cos(angulos)
y = r * np.sin(angulos)

plt.figure()
plt.plot(x, y, 'g-', linewidth=2, label="Pentágono")
plt.axis('equal')
plt.legend()
plt.title("Pentágono regular")
plt.show()
```

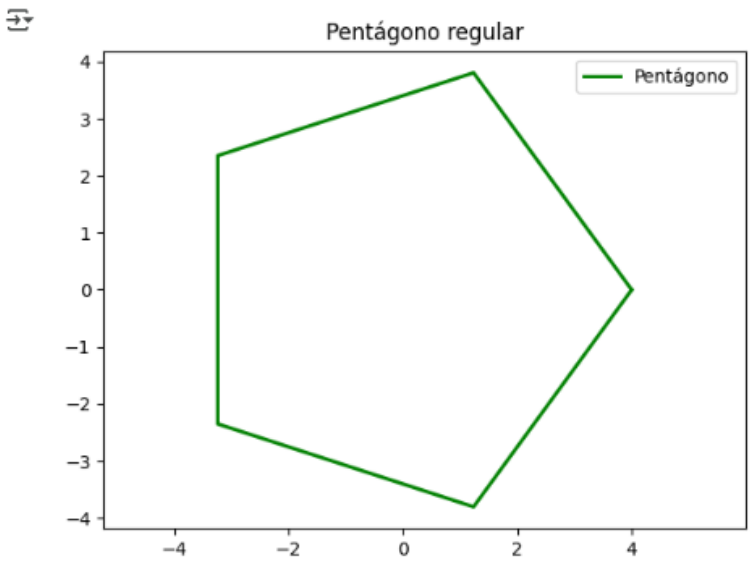

Figura 24. Salida esperada de un pentágono regular.

Ejercicio Propuesto 1.3.3.1.

- Cambiar los datos en las instrucciones para formar otro polígono regular, de 7 y 8 lados.
- Modificar el color del polígono.

1.3.4. Dibujo de circunferencias

La circunferencia se representa con ecuaciones paramétricas en el primer cuadrante (Ecuaciones 1-2):

$$x = r \cdot \cos(t) \quad (1)$$

$$y = r \cdot \text{sen}(t) \quad (2)$$

```
import numpy as np
import matplotlib.pyplot as plt

r = 5
t = np.linspace(0, 2*np.pi, 300)

x = r * np.cos(t)
y = r * np.sin(t)

plt.figure()
plt.plot(x, y, 'm-', linewidth=2, label="Circunferencia")
plt.axis('equal')
plt.legend()
plt.title("Circunferencia")
plt.show()
```

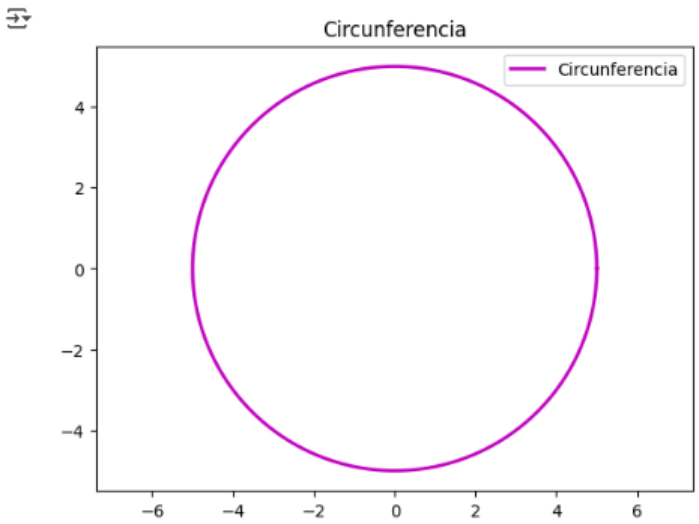

Figura 25. Salida esperada de un círculo.

Explicación de este código:

Ese código con **numPy** está creando **los puntos (x, y) de una circunferencia completa de radio 5** centrada en el origen.

```
r = 5 # Radio de la circunferencia = 5 unidades.
```

```
t = np.linspace(0, 2*np.pi, 300) # Genera 300 valores de t igualmente espaciados desde 0 hasta 2π radianes. Esto cubre toda la vuelta alrededor del círculo completo(360°).

x = r * np.cos(t)
y = r * np.sin(t)

# Aplica las ecuaciones paramétricas de la circunferencia para cada valor de t. x e y serán arrays con 300 puntos que, al graficarlos, forman un círculo perfecto.

# Interpretación: Si quisiéramos solo la parte de la circunferencia del primer cuadrante, cambiaríamos la línea de t a:

t = np.linspace(0, np.pi/2, 300)
```

```
import numpy as np
import matplotlib.pyplot as plt

r = 50  # radio
t = np.linspace(0, np.pi/2, 300)
x = r * np.cos(t)
y = r * np.sin(t)

plt.figure()
plt.plot(x, y)
plt.axis('equal')
plt.title("Circunferencia técnica generada con Python")
plt.show()
```

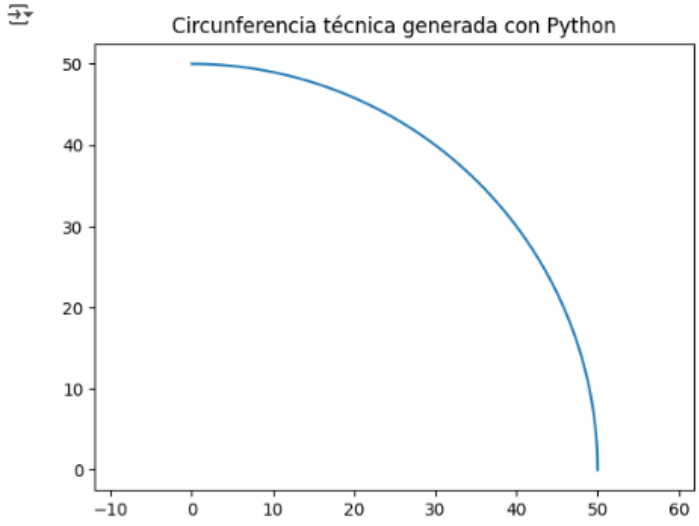

Figura 26. Salida esperada.

Ejercicio Propuesto 1.3.4.1.

- Dibujar en la misma figura:
  - Un punto en (2, 3).
  - Un segmento de (0,0) a (4,5).
  - Un triángulo equilátero de lado 5.
  - Una circunferencia de radio 3 centrada en el origen.
- Usar diferentes colores y estilos de línea.
- Mantener proporciones reales (plt.axis('equal')).

Ejercicio Propuesto Avanzado 1.3.4.2.

- Dibujar en Crear una función en Python que reciba:
  - Tipo de figura ("punto", "segmento", "polígono", "circunferencia").
  - Parámetros (coordenadas, número de lados, radio…).
- La función deberá dibujar la figura correspondiente.
- Opcional: añadir la opción de exportar a DXF para su uso en AutoCAD.