Dibujando Funciones con Python

Materia: Algoritmos y Programación

Prof. D.Sc. BARSEKH-ONJI Aboud
Facultad de Ingeniería
Universidad Anáhuac México

23 de octubre de 2025

Agenda

1. Introducción y Herramientas
2. Nuestro Primer Gráfico
3. Tarea 1

¿Por qué Dibujar Funciones?

Más allá de `print()`

Hasta ahora, la única forma de ver un resultado es con la función `print()`. Esto funciona para texto y números, pero es muy poco intuitivo para entender una ecuación.

Ver para Entender

Para comprender realmente una función, como $y = x^2$, necesitamos **ver su gráfico**. En programación, a esto se le llama **visualización** o **"plotteo"**.

Aplicaciones

- Visualizar datos de un experimento.
- Analizar el rendimiento de un algoritmo.
- Simular un fenómeno físico.
- ¡Entender las matemáticas que estamos programando!

Las Herramientas Esenciales

Python no puede hacerlo solo

Python estándar no incluye herramientas de graficación. Para esto, usamos **bibliotecas externas**. Dos de las más importantes en el mundo de la ciencia y la ingeniería son:

NumPy (Numerical Python)

Es la biblioteca para cálculos numéricos avanzados.

- Su objeto principal es el **array**.
- La usaremos para **crear los datos** (nuestros puntos en el eje X) de forma rápida y eficiente.

Matplotlib (Plotting Library)

Es la biblioteca más popular para dibujar y crear gráficos en Python.

- Usaremos su módulo `pyplot`.
- La usaremos para tomar nuestros datos y **crear el dibujo** (el gráfico).

Instalación (Un solo paso)

Instalando las Bibliotecas

Como `numpy` y `matplotlib` son bibliotecas externas, debemos instalarlas en nuestra computadora **una sola vez**.

Usando `pip`

Abrimos una terminal o símbolo del sistema (no es el script de Python) y escribimos el siguiente comando:

```
# Esto descarga e instala ambas bibliotecas
pip install numpy matplotlib

```

Examples

Nota Esto solo se hace una vez. Una vez instaladas, ya puedes usarlas en todos tus programas de Python simplemente con `import`.

Agenda

1. Introducción y Herramientas
2. Nuestro Primer Gráfico
3. Tarea 1

El "Hola, Mundo"del Ploteo

Objetivo: Dibujar $y = x^2$

Vamos a crear un gráfico simple para la función $y = x^2$ en el rango de $x = -10$ a $x = 10$.

El "Hola, Mundo"del Ploteo

Código Completo

Este es el código mínimo para crear un gráfico:

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  # 1. Preparar los datos
5  x = np.linspace(-10, 10, 100) # 100 puntos entre -10 y 10
6  y = x**2                      # Calcula el cuadrado de CADA punto
7
8  # 2. Dibujar el grafico
9  plt.plot(x, y) # Crea el grafico en memoria
10
11 # 3. Mostrar el grafico
12 plt.show()     # Abre la ventana con el resultado
13
```

Análisis (Paso 1): Los Datos con NumPy

`import numpy as np`

Importamos la biblioteca y le damos un .ªlias.º apodo (‘np‘) para escribir menos. Es una convención estándar.

`x = np.linspace(-10, 10, 100)`

Este es el comando clave de NumPy para nosotros.

- **`linspace`** significa .ªespacio lineal".
- Genera un array (similar a una lista) con **100** números, espaciados uniformemente entre **-10** y **10**.
- Estos 100 números serán nuestros "puntos X".

Examples

`y = x**2` ¡Aquí está la magia de NumPy! No necesitamos un ciclo `for`. Al elevar el *array* `x` al cuadrado, NumPy aplica la operación a **cada uno de los 100 elementos** automáticamente y nos devuelve un nuevo array `y` con los resultados.

Análisis (Paso 2): El Dibujo con Matplotlib

`import matplotlib.pyplot as plt`

De forma similar, importamos el módulo `pyplot` de `matplotlib` y le damos el alias estándar `plt`.

`plt.plot(x, y)`

Este es el comando de dibujo principal. Le decimos a Matplotlib: "Toma el array `x` como eje horizontal y el array `y` como eje vertical, y dibuja una línea que los conecte".

Examples

`plt.show()` Este comando le dice a Python: "He terminado de configurar mi gráfico. Ahora, por favor, abre la ventana y muéstraselo al usuario". Sin `plt.show()`, el gráfico se crea en memoria pero nunca se muestra.

Agenda

1. Introducción y Herramientas
2. Nuestro Primer Gráfico
3. Tarea 1

Tarea 1: Dibuja una Función Trigonométrica

Objetivo

Usando el código que acabamos de ver como plantilla, crea un programa que dibuje la función **Seno** ($y = \sin(x)$).

Tarea 1: Dibuja una Función Trigonométrica

Instrucciones

Debes graficar la función en el rango de $x = 0$ a $x = 2\pi$.

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

# Pista: NumPy tiene su propia constante 'pi'
# 1. Crear datos para x:
#    (Usa np.linspace para ir de 0 a 2*np.pi)
x = ...

# 2. Crear datos para y:
#    (Pista: NumPy tiene su propia funcion 'sin')
y = np.sin(x)

```