Introducción

Victor Daniel Yeme Morfin

Hello World

In [1]:

```
print('Hello World')
```

```
Hello World
```

Importar Librerias

In [11]:

```
import numpy as np
import sympy as sym
import matplotlib.pyplot as plt
import pandas as pd
import scipy.optimize as opt
import yfinance as yf
import geopandas as gpd
from sklearn.preprocessing import StandardScaler
print("Todas las librerías están instaladas correctamente.")
```

```
Todas las librerías están instaladas correctamente.
```

1. Numpy: Operaciones numéricas

In [4]:

```
import numpy as np
# Crear un array y calcular su suma
array = np.array([1, 2, 3, 4, 5])
print("Array:", array)
print("Suma del array:", np.sum(array))
```

```
Array: [1 2 3 4 5]
Suma del array: 15
```

2. Sympy: Cálculo simbólico

In [7]:

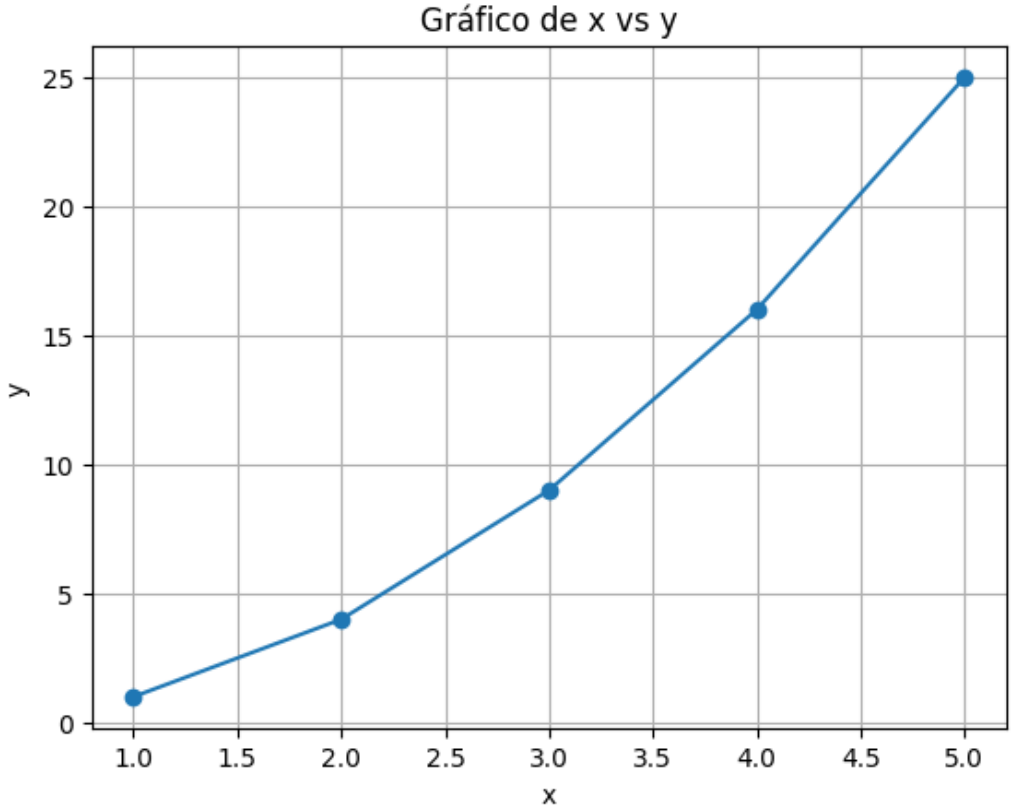
```
import sympy as sym
# Definir una variable simbólica y calcular la derivada
x = sym.Symbol('x')
func = x**2 + 3*x + 2
derivada = sym.diff(func, x)
print("Función:", func)
print("Derivada:", derivada)
```

```
Función: x**2 + 3*x + 2
Derivada: 2*x + 3
```

3. Matplotlib: Gráficos

In [8]:

```
import matplotlib.pyplot as plt
 # Crear un gráfico simple
 x = [1, 2, 3, 4, 5]
 y = [1, 4, 9, 16, 25]
 plt.plot(x, y, marker='o')
 plt.title("Gráfico de x vs y")
 plt.xlabel("x")
 plt.ylabel("y")
 plt.grid(True)
 plt.show()
```



4. Pandas: Manipulación de datos

In [20]:

```
import pandas as pd
# Crear un DataFrame y realizar operaciones básicas
data = {'Nombre': ['Ana', 'Luis', 'María'], 'Edad': [25, 30, 22]}
df = pd.DataFrame(data)
print("DataFrame:")
print(df)
print("\nEdades promedio:", df['Edad'].mean())
```

```
DataFrame:
  Nombre  Edad
0    Ana    25
1   Luis    30
2  María    22

Edades promedio: 25.666666666666668
```

5. Scipy: Optimización

In [19]:

```
import scipy.optimize as opt
# Encontrar el mínimo de una función cuadrática
def f(x):
    return x**2 + 3*x + 2
resultado = opt.minimize(f, x0=0) # x0 es el punto inicial
print("Punto mínimo:", resultado.x)
```

```
Punto mínimo: [-1.50000001]
```

6. Time: Medición del tiempo de ejecución

In [22]:

```
import time
# Medir cuánto tiempo tarda un bucle
start_time = time.time()
for i in range(1000000):
    pass
end_time = time.time()
print("Tiempo de ejecución:", end_time - start_time, "segundos")
```

```
Tiempo de ejecución: 0.15262413024902344 segundos
```

7. Yfinance: Datos financieros

In [23]:

```
import yfinance as yf
# Descargar datos históricos de una acción
apple = yf.Ticker("AAPL")
hist = apple.history(period="5d") # Últimos 5 días
print("Datos históricos de AAPL:")
print(hist)
```

```
Datos históricos de AAPL:
                                 Open        High         Low       Close  \
Date
2025-01-16 00:00:00-05:00  237.350006  238.009995  228.029999  228.259995
2025-01-17 00:00:00-05:00  232.119995  232.289993  228.479996  229.979996
2025-01-21 00:00:00-05:00  224.000000  224.419998  219.380005  222.639999
2025-01-22 00:00:00-05:00  219.789993  224.119995  219.789993  223.830002
2025-01-23 00:00:00-05:00  224.740005  227.029999  222.300003  223.660004

                             Volume  Dividends  Stock Splits
Date
2025-01-16 00:00:00-05:00  71759100        0.0           0.0
2025-01-17 00:00:00-05:00  68488300        0.0           0.0
2025-01-21 00:00:00-05:00  98070400        0.0           0.0
2025-01-22 00:00:00-05:00  64126500        0.0           0.0
2025-01-23 00:00:00-05:00  60178200        0.0           0.0
```

8. Sklearn: Preprocesamiento de datos

In [24]:

```
from sklearn.preprocessing import StandardScaler
import numpy as np
# Escalar datos
data = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
scaler = StandardScaler()
scaled_data = scaler.fit_transform(data)
print("Datos originales:")
print(data)
print("\nDatos escalados:")
print(scaled_data)
```

```
Datos originales:
[[1 2 3]
 [4 5 6]
 [7 8 9]]

Datos escalados:
[[-1.22474487 -1.22474487 -1.22474487]
 [ 0.          0.          0.        ]
 [ 1.22474487  1.22474487  1.22474487]]
```

$$\begin{bmatrix} 1.2274/4486 & 1.2274/4486 & 1.2274/4486 \end{bmatrix}$$