1) Suppose you are a teacher, and you want to analyze the exam scores of your students in a particular subject. You have recorded the scores of your students for a recent exam, and you want to represent this data using a Pandas Series.

Input:

students = ['Alice', 'Bob', 'Charlie', 'David', 'Eve', 'Frank', 'Grace', 'Hannah', 'Ivy', 'Jack'] exam\_scores = [92, 88, 76, 94, 82, 90, 85, 89, 78, 91]

# Import the pandas library

import pandas as pd

# List of students' names

students = ['Alice', 'Bob', 'Charlie', 'David', 'Eve', 'Frank', 'Grace', 'Hannah', 'Ivy', 'Jack']

# Corresponding exam scores

exam\_scores = [92, 88, 76, 94, 82, 90, 85, 89, 78, 91]

# Create a Pandas Series to hold the exam scores with student names as the index

exam\_series = pd.Series(data=exam\_scores, index=students)

# Display the Pandas Series

print(exam\_series)

Output:

Alice 92

Bob 88

Charlie 76

David 94

Eve 82

Frank 90

Grace 85

Hannah 89

Ivy 78

Jack 91

dtype: int64

2) Suppose you want to track and analyze your household expenses for a month. You have recorded the expenses for various categories, such as groceries, utilities, rent, transportation, and entertainment. You can represent this expense data using a Pandas Series.

Input:

# Expense categories

categories = ['Groceries', 'Utilities', 'Rent', 'Transportation', 'Entertainment']

# Monthly expense data (example data in USD)

expenses = [500, 200, 1200, 300, 150]

# Import the pandas library

import pandas as pd

# List of expense categories

categories = ['Groceries', 'Utilities', 'Rent', 'Transportation', 'Entertainment']

# Corresponding monthly expense data (example data in USD)

expenses = [500, 200, 1200, 300, 150]

# Create a Pandas Series to hold the expense data with categories as the index

expense\_series = pd.Series(data=expenses, index=categories)

# Display the Pandas Series

print(expense\_series)

Output:

Groceries 500

Utilities 200

Rent 1200

Transportation 300

Entertainment 150

dtype: int64

3) ChatGPT Exercise:

Using ChatGPT generate the python code to solve the same problem

Scenario: Analyzing Stock Prices

Suppose you are an investor interested in analyzing the stock prices of a particular company over a year. You have collected daily closing prices for that company's stock, and you want to perform some analysis like Calculate the average daily return, Find the date with the highest closing price and also generate a line chart using Pandas Series.

Further, you need to get some inference out of the chart.

Create a ChatGPT prompt to generate the code for this scenario. Based on the code generated, ask ChatGPT to give the conclusion/inference.

Note. You can provide the data to ChatGPT in the form of a list or dictionary or ask it to use sample data.

# Import necessary libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# Sample data: dates and closing prices

dates = pd.date\_range(start="2023-01-01", periods=365, freq='D') # Create a range of dates for one year

closing\_prices = np.random.normal(loc=100, scale=10, size=365).round(2) # Generate random closing prices around $100

# Creating a Pandas Series for the closing prices

stock\_prices = pd.Series(data=closing\_prices, index=dates) # Create a Series with dates as index and closing prices as data

# Calculating the average daily return

average\_daily\_return = stock\_prices.pct\_change().mean() # Calculate the percentage change and find the mean

# Finding the date with the highest closing price

highest\_closing\_price\_date = stock\_prices.idxmax() # Find the index (date) of the maximum closing price

# Generating a line chart of the stock prices

plt.figure(figsize=(10, 6)) # Set the figure size

stock\_prices.plot() # Plot the stock prices

plt.title('Daily Closing Prices') # Add title to the chart

plt.xlabel('Date') # Label the x-axis

plt.ylabel('Closing Price (USD)') # Label the y-axis

plt.grid(True) # Add grid to the chart

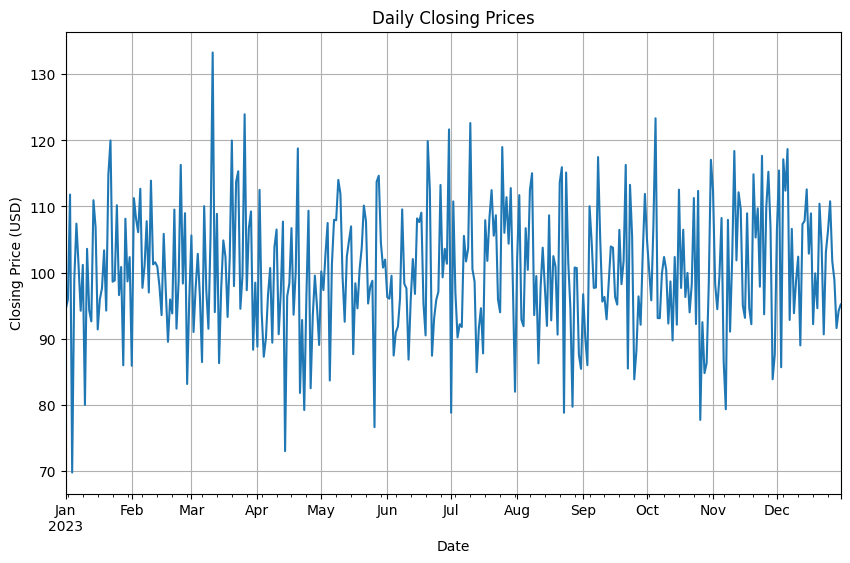
plt.show() # Display the chart

# Output the results

print(f"Average Daily Return: {average\_daily\_return:.4f}") # Print the average daily return

print(f"Date with Highest Closing Price: {highest\_closing\_price\_date}") # Print the date with the highest closing price

Output: The graph below describes the output



Average Daily Return: 0.0098

Date with Highest Closing Price: 2023-03-11 00:00:00

**Conclusions/Inferences from the Chart:**

After running the above code, here are some possible conclusions or inferences:

1. **Trend Analysis:**
   * The line chart will show the trend of the stock prices over the year. We can observe periods of upward or downward trends, indicating bullish or bearish phases.
2. **Volatility:**
   * The fluctuations in the line chart can give an idea about the stock's volatility. Large swings suggest high volatility, while a smoother line indicates stability.
3. **Average Daily Return:**
   * The calculated average daily return provides an estimate of the stock's performance on a daily basis. A positive value indicates overall growth, while a negative value suggests a decline.
4. **Highest Closing Price:**
   * The date with the highest closing price indicates the peak performance of the stock during the year. This could be associated with specific events or market conditions.
5. **Support and Resistance Levels:**
   * By examining the peaks and troughs in the chart, we can infer potential support and resistance levels where the stock price tends to reverse its direction.

These inferences can help in making informed investment decisions and understanding the stock's behavior over time.