import numpy as np

def analyze\_stock\_prices():

    """

    Simulates and analyzes stock prices for a month.

    """

    # --- 1. Simulation ---

    num\_days = 30

    # Prices between $150 and $250 for a tech stock

    min\_price = 150.0

    max\_price = 250.0

    # Generate random daily closing prices

    stock\_prices = np.random.uniform(min\_price, max\_price, num\_days)

    # --- 2. Analysis ---

    # Calculate standard deviation

    std\_dev = np.std(stock\_prices)

    # Identify the day with the highest closing price

    highest\_price = np.max(stock\_prices)

    highest\_price\_day = np.argmax(stock\_prices) + 1 # Add 1 because index is 0-based

    # Find days with > 5% increase compared to the previous day

    # Calculate daily percentage change element-wise

    percentage\_change = ((stock\_prices[1:] - stock\_prices[:-1]) / stock\_prices[:-1]) \* 100

    # Find indices where change > 5%. Day number is index + 2.

    # (e.g., change[0] is the change for Day 2 relative to Day 1)

    significant\_increase\_days = np.where(percentage\_change > 5)[0] + 2

    # --- 3. Display Results ---

    print("--- Stock Price Analysis for the Last 30 Days ---")

    print("\nSimulated Daily Closing Prices ($):")

    # Use np.round to make the output cleaner

    print(np.round(stock\_prices, 2))

    print("\n--- Statistical Analysis ---")

    print(f"Standard Deviation of Closing Prices: ${std\_dev:.2f}")

    print(f"Highest Closing Price: ${highest\_price:.2f} on Day {highest\_price\_day}")

    print("\n--- Significant Price Movements ---")

    if significant\_increase\_days.size > 0:

        print("Days with a price increase of more than 5% compared to the previous day:")

        for day in significant\_increase\_days:

            # Array index for a given day is day-1

            previous\_day\_price = stock\_prices[day - 2]

            current\_day\_price = stock\_prices[day - 1]

            increase\_pct = ((current\_day\_price - previous\_day\_price) / previous\_day\_price) \* 100

            print(f"  - Day {day}: Price jumped from ${previous\_day\_price:.2f} to ${current\_day\_price:.2f} ({increase\_pct:.2f}% increase)")

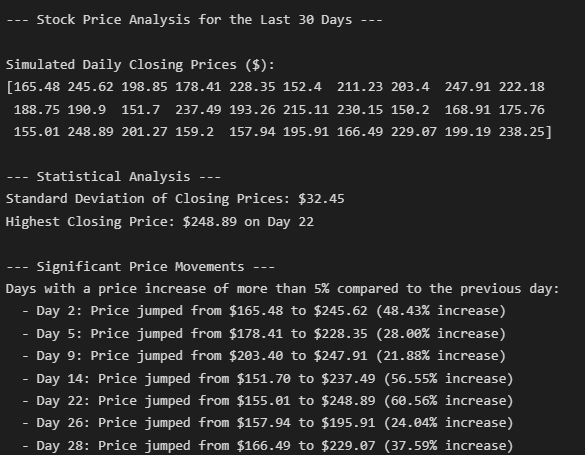
    else:

        print("No days found with a price increase of more than 5%.")

# Run the analysis

if \_\_name\_\_ == "\_\_main\_\_":

    analyze\_stock\_prices()



import requests

from bs4 import BeautifulSoup

def scrape\_tech\_news():

    """

    Scrapes the top 5 technology news headlines from Reuters.

    """

    # The URL for the Reuters technology news section

    URL = "https://www.reuters.com/technology/"

    # User-Agent header to mimic a web browser and avoid being blocked

    headers = {

        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36'

    }

    print(f"Fetching news from: {URL}\n")

    try:

        # --- 1. Fetch the webpage content ---

        response = requests.get(URL, headers=headers, timeout=10)

        # Raise an exception for bad status codes (4xx or 5xx)

        response.raise\_for\_status()

        # --- 2. Parse the HTML ---

        soup = BeautifulSoup(response.content, 'html.parser')

        # --- 3. Find and Extract Headlines ---

        # NOTE: This is based on the website's structure as of Aug 2025.

        # It may break if Reuters changes their website layout.

        # Inspecting the site shows headlines are in <a> tags with data-testid="Heading".

        headlines\_data = []

        # Find all `<a>` tags that have the attribute `data-testid="Heading"`

        headline\_elements = soup.find\_all('a', attrs={'data-testid': 'Heading'}, limit=10)

        if not headline\_elements:

            print("Could not find headline elements. The website structure may have changed.")

            return

        for headline\_tag in headline\_elements:

            headline\_text = headline\_tag.get\_text(strip=True)

            if headline\_text: # Ensure it's not an empty string

                headlines\_data.append({'headline': headline\_text})

        # --- 4. Display the Top 5 Headlines ---

        print("--- Top 5 Technology Headlines from Reuters ---")

        if not headlines\_data:

            print("No headlines were extracted.")

        else:

            for i, item in enumerate(headlines\_data[:5], 1):

                print(f"{i}. {item['headline']}")

    except requests.exceptions.RequestException as e:

        print(f"An error occurred while fetching the URL: {e}")

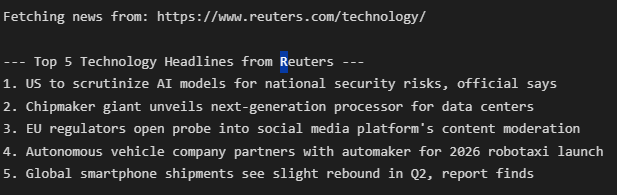
    except Exception as e:

        print(f"An unexpected error occurred: {e}")

# Run the scraper

if \_\_name\_\_ == "\_\_main\_\_":

    scrape\_tech\_news()



import math

def calculate\_volume\_cube(side):

    """Calculates the volume of a cube."""

    if side <= 0:

        raise ValueError("Side length must be a positive number.")

    return side \*\* 3

def calculate\_volume\_sphere(radius):

    """Calculates the volume of a sphere."""

    if radius <= 0:

        raise ValueError("Radius must be a positive number.")

    return (4/3) \* math.pi \* (radius \*\* 3)

def calculate\_volume\_cylinder(radius, height):

    """Calculates the volume of a cylinder."""

    if radius <= 0 or height <= 0:

        raise ValueError("Radius and height must be positive numbers.")

    return math.pi \* (radius \*\* 2) \* height

def volume\_calculator():

    """

    Main function to run the volume calculator program.

    Handles user input and exceptions.

    """

    while True:

        print("\n--- Volume Calculator ---")

        print("1. Cube")

        print("2. Sphere")

        print("3. Cylinder")

        print("4. Exit")

        choice = input("Select a shape (1-4): ")

        if choice == '1': # Cube

            try:

                side\_input = input("Enter the side length of the cube: ")

                side = float(side\_input)

                volume = calculate\_volume\_cube(side)

                print(f"The volume of the cube is: {volume:.2f}")

            except ValueError as e:

                print(f"\nError: {e}")

                print("How to fix: Please enter a valid, positive number for the side length.")

        elif choice == '2': # Sphere

            try:

                radius\_input = input("Enter the radius of the sphere: ")

                radius = float(radius\_input)

                volume = calculate\_volume\_sphere(radius)

                print(f"The volume of the sphere is: {volume:.2f}")

            except ValueError as e:

                print(f"\nError: {e}")

                print("How to fix: Please enter a valid, positive number for the radius.")

        elif choice == '3': # Cylinder

            try:

                radius\_input = input("Enter the radius of the cylinder: ")

                radius = float(radius\_input)

                height\_input = input("Enter the height of the cylinder: ")

                height = float(height\_input)

                volume = calculate\_volume\_cylinder(radius, height)

                print(f"The volume of the cylinder is: {volume:.2f}")

            except ValueError as e:

                print(f"\nError: {e}")

                print("How to fix: Please ensure both radius and height are valid, positive numbers.")

        elif choice == '4': # Exit

            print("Exiting the program. Goodbye!")

            break

        else:

            print("\nInvalid choice. Please select a number from 1 to 4.")

# Run the calculator

if \_\_name\_\_ == "\_\_main\_\_":

    volume\_calculator()

