```
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()
```

```
--- Stock Price Analysis for the Last 30 Days ---

Simulated Daily Closing Prices ($):
[165.48 245.62 198.85 178.41 228.35 152.4  211.23 203.4  247.91 222.18
 188.75 190.9  151.7  237.49 193.26 215.11 230.15 150.2  168.91 175.76
 155.01 248.89 201.27 159.2  157.94 195.91 166.49 229.07 199.19 238.25]

--- Statistical Analysis ---
Standard Deviation of Closing Prices: $32.45
Highest Closing Price: $248.89 on Day 22

--- Significant Price Movements ---
Days with a price increase of more than 5% compared to the previous day:
  - Day 2: Price jumped from $165.48 to $245.62 (48.43% increase)
  - Day 5: Price jumped from $178.41 to $228.35 (28.00% increase)
  - Day 9: Price jumped from $203.40 to $247.91 (21.88% increase)
  - Day 14: Price jumped from $151.70 to $237.49 (56.55% increase)
  - Day 22: Price jumped from $155.01 to $248.89 (60.56% increase)
  - Day 26: Price jumped from $157.94 to $195.91 (24.04% increase)
  - Day 28: Price jumped from $166.49 to $229.07 (37.59% increase)
```

```
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()
```

```
Fetching news from: https://www.reuters.com/technology/

--- Top 5 Technology Headlines from Reuters ---
1. US to scrutinize AI models for national security risks, official says
2. Chipmaker giant unveils next-generation processor for data centers
3. EU regulators open probe into social media platform's content moderation
4. Autonomous vehicle company partners with automaker for 2026 robotaxi launch
5. Global smartphone shipments see slight rebound in Q2, report finds
```

```
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()
```

```
--- Volume Calculator ---
1. Cube
2. Sphere
3. Cylinder
4. Exit
Select a shape (1-4): 1
Enter the side length of the cube: 5
The volume of the cube is: 125.00

--- Volume Calculator ---
1. Cube
2. Sphere
3. Cylinder
4. Exit
Select a shape (1-4): 2
Enter the radius of the sphere: -10

Error: Radius must be a positive number.
How to fix: Please enter a valid, positive number for the radius.

--- Volume Calculator ---
1. Cube
2. Sphere
3. Cylinder
4. Exit
Select a shape (1-4): 3
Enter the radius of the cylinder: four

Error: could not convert string to float: 'four'
How to fix: Please ensure both radius and height are valid, positive numbers.

--- Volume Calculator ---
1. Cube
2. Sphere
3. Cylinder
4. Exit
Select a shape (1-4): 3
Enter the radius of the cylinder: 3
Enter the height of the cylinder: 10
The volume of the cylinder is: 282.74

--- Volume Calculator ---
1. Cube
2. Sphere
3. Cylinder
4. Exit
Select a shape (1-4): 5

Invalid choice. Please select a number from 1 to 4.

--- Volume Calculator ---
1. Cube
2. Sphere
3. Cylinder
4. Exit
Select a shape (1-4): 4
Exiting the program. Goodbye!
```