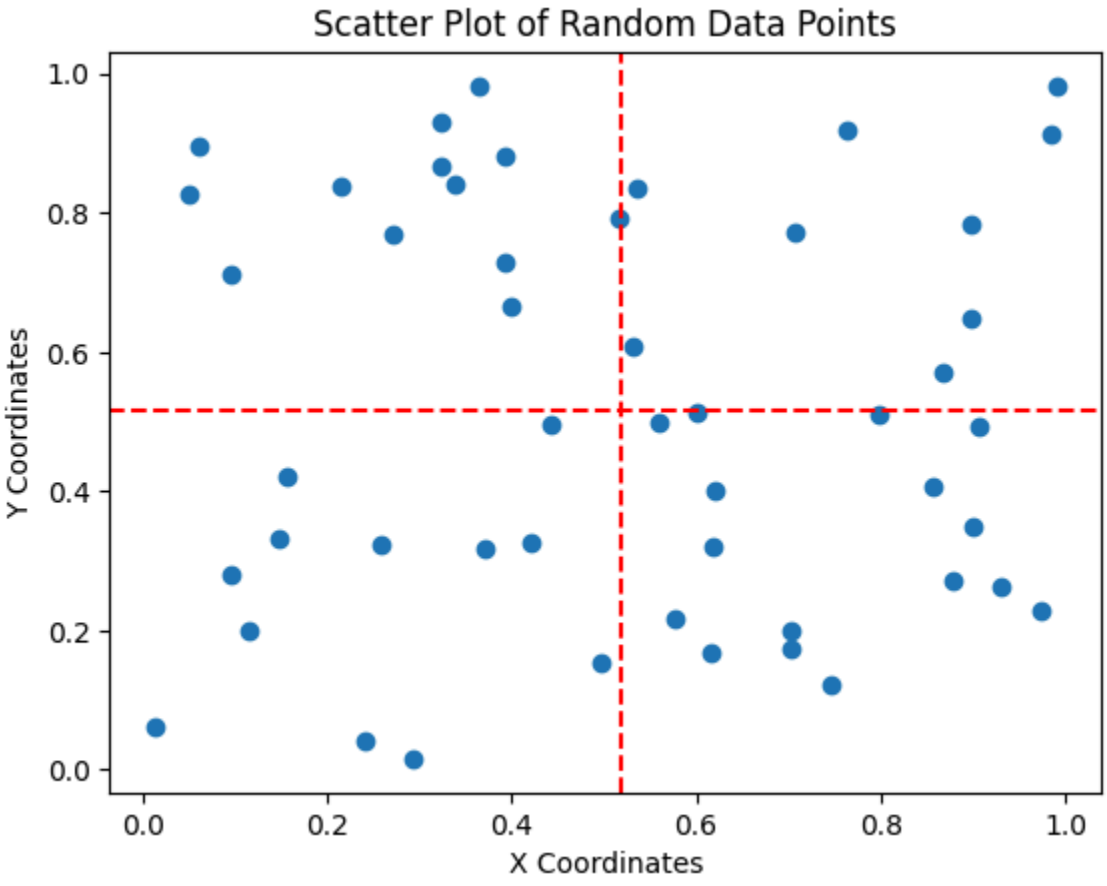
Question 1

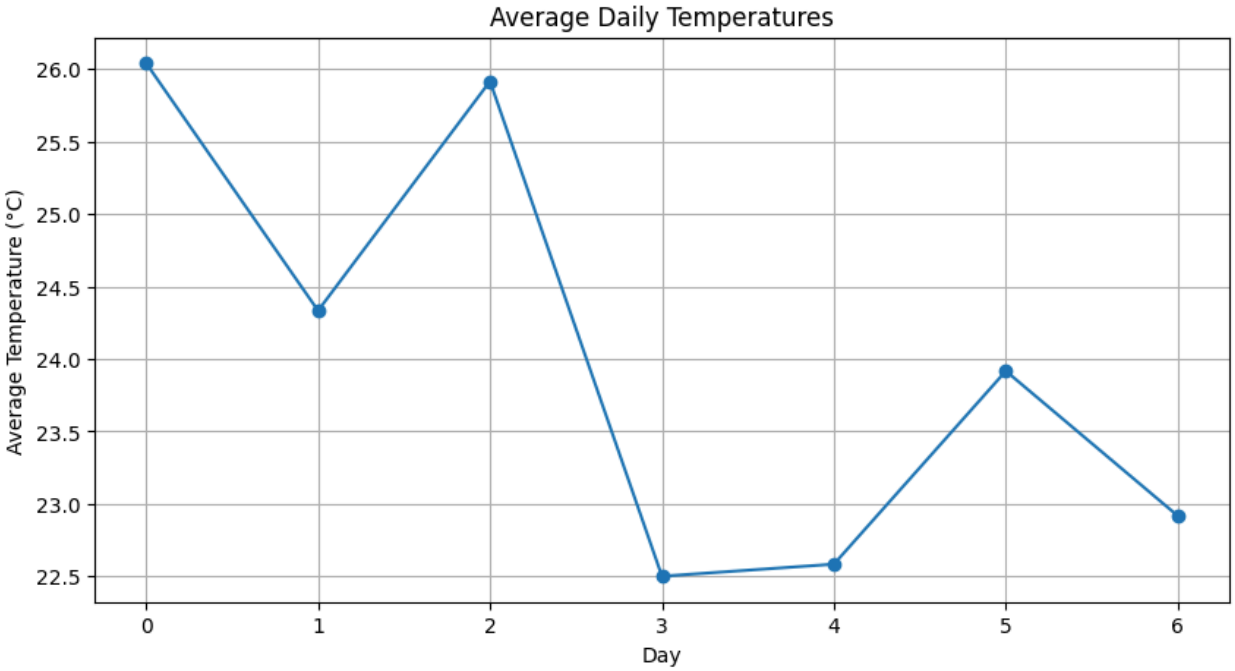
In [ ]:
```
# Consider two lists representing the x and y coordinates of 50 random data po
# 1. Create a scatter plot of the data points.
# 2. Calculate and display the average value of the x-coordinates and the y-co

import matplotlib.pyplot as plt
import numpy as np

x = np.random.rand(50)
y = np.random.rand(50)

plt.scatter(x, y)
plt.xlabel('X Coordinates')
plt.ylabel('Y Coordinates')
plt.title('Scatter Plot of Random Data Points')

avg_x = np.mean(x)
avg_y = np.mean(y)

plt.axhline(y=avg_y, color='r', linestyle='--')
plt.axvline(x=avg_x, color='r', linestyle='--')

plt.show()
```

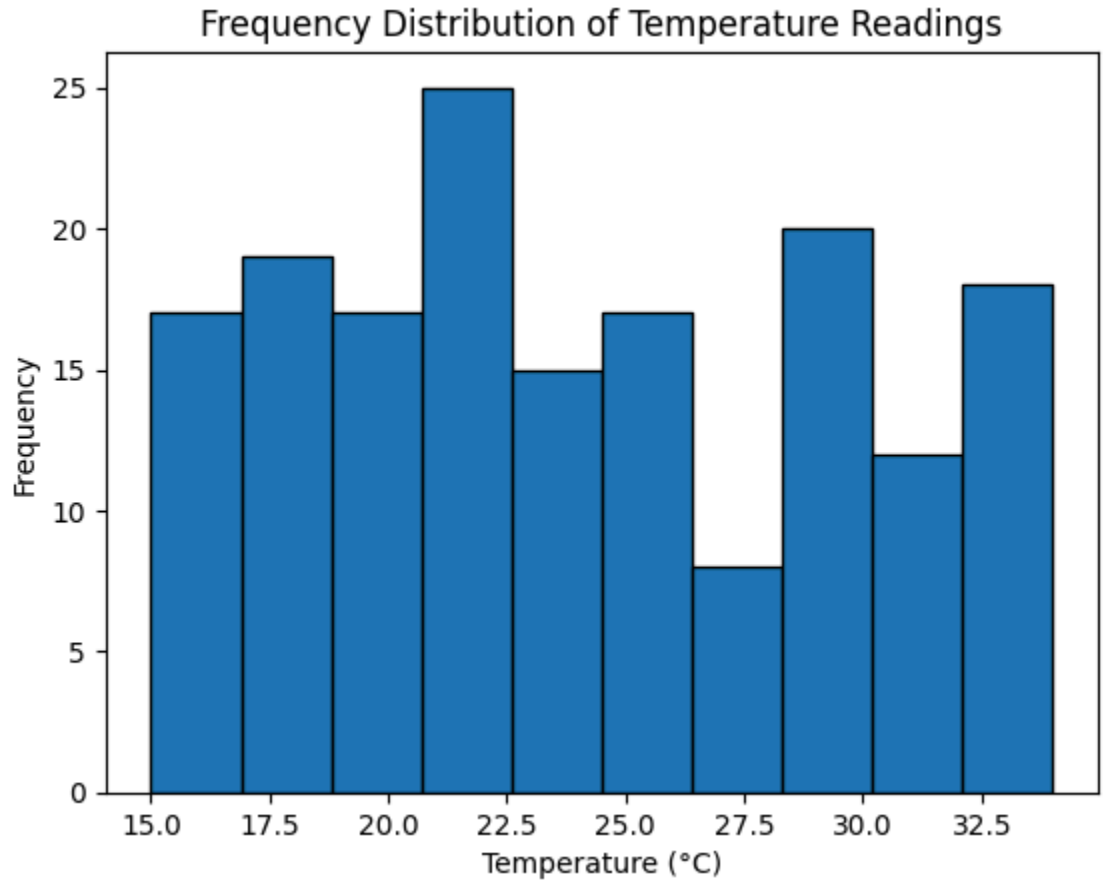


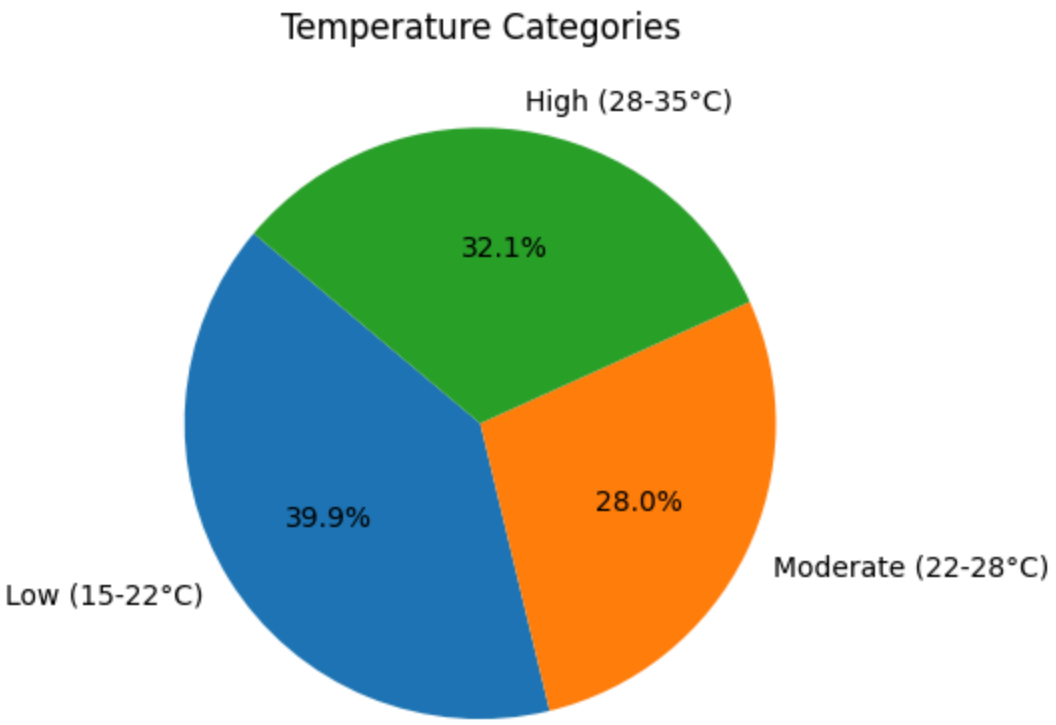
Question 2

In [ ]:

```python
# Analyze temperature data from a weather station for seven days by generating

import numpy as np
import matplotlib.pyplot as plt

# Generate random temperature readings
temps = np.random.randint(15, 35, size=(7, 24))

# Calculate average daily temperatures
avg_daily_temps = np.mean(temps, axis=1)

# Identify maximum and minimum readings over the week
max_temp = np.max(temps)
min_temp = np.min(temps)

# Visualize daily averages in a line chart
plt.figure(figsize=(10, 5))
plt.plot(avg_daily_temps, marker='o')
plt.xlabel('Day')
plt.ylabel('Average Temperature (°C)')
plt.title('Average Daily Temperatures')
plt.grid(True)
plt.show()

# Show frequency distribution of all readings in a histogram
plt.hist(temps.flatten(), bins=10, edgecolor='black')
plt.xlabel('Temperature (°C)')
plt.ylabel('Frequency')
plt.title('Frequency Distribution of Temperature Readings')
plt.show()

# Create a pie chart categorizing the readings
low = np.sum((temps >= 15) & (temps < 22))
moderate = np.sum((temps >= 22) & (temps < 28))
high = np.sum((temps >= 28) & (temps <= 35))

labels = ['Low (15-22°C)', 'Moderate (22-28°C)', 'High (28-35°C)']
sizes = [low, moderate, high]

plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
plt.title('Temperature Categories')
plt.show()
```

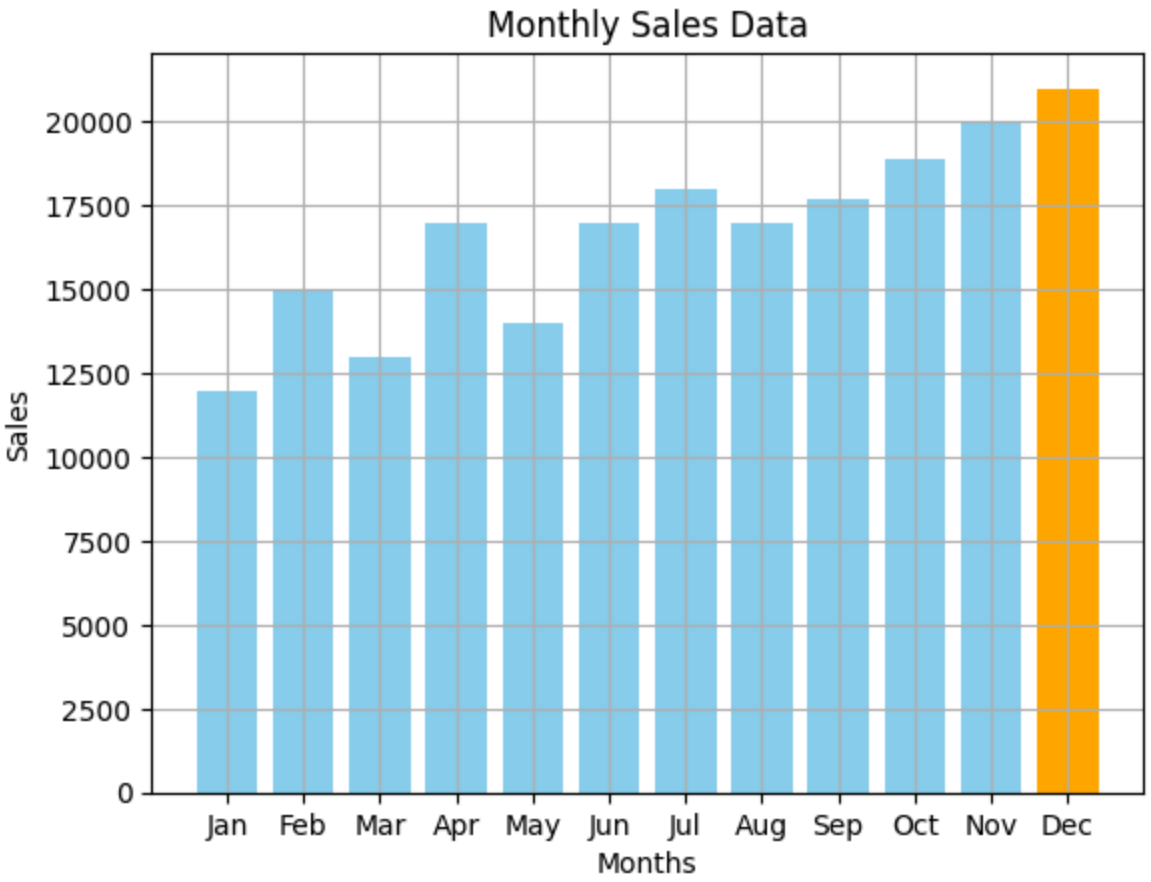

Average Daily Temperatures
Average Temperature (°C)
26.0
25.5
25.0
24.5
24.0
23.5
23.0
22.5
0
1
2
3
4
5
6
Day


Frequency Distribution of Temperature Readings
Frequency
25
20
15
10
5
0
15.0
17.5
20.0
22.5
25.0
27.5
30.0
32.5
Temperature (°C)



Question 3

In [23]:
```
# Consider sales data for each month in a year, stored in two lists (months an
# 1. Creates a bar chart to represent monthly sales.
# 2. Highlights the month with the highest sales.
# 3. Adds labels, a title, and appropriate colors to make the chart visually a

months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct'
sales = [12000, 15000, 13000, 17000, 14000, 17000, 18000, 17000, 17700, 18900,

plt.bar(months, sales, color='skyblue')
plt.xlabel('Months')
plt.ylabel('Sales')
plt.title('Monthly Sales Data')

# Highlight the month with the highest sales
max_sales = max(sales)
max_month = months[sales.index(max_sales)]
plt.bar(max_month, max_sales, color='orange')

plt.grid(True)
plt.show()
```



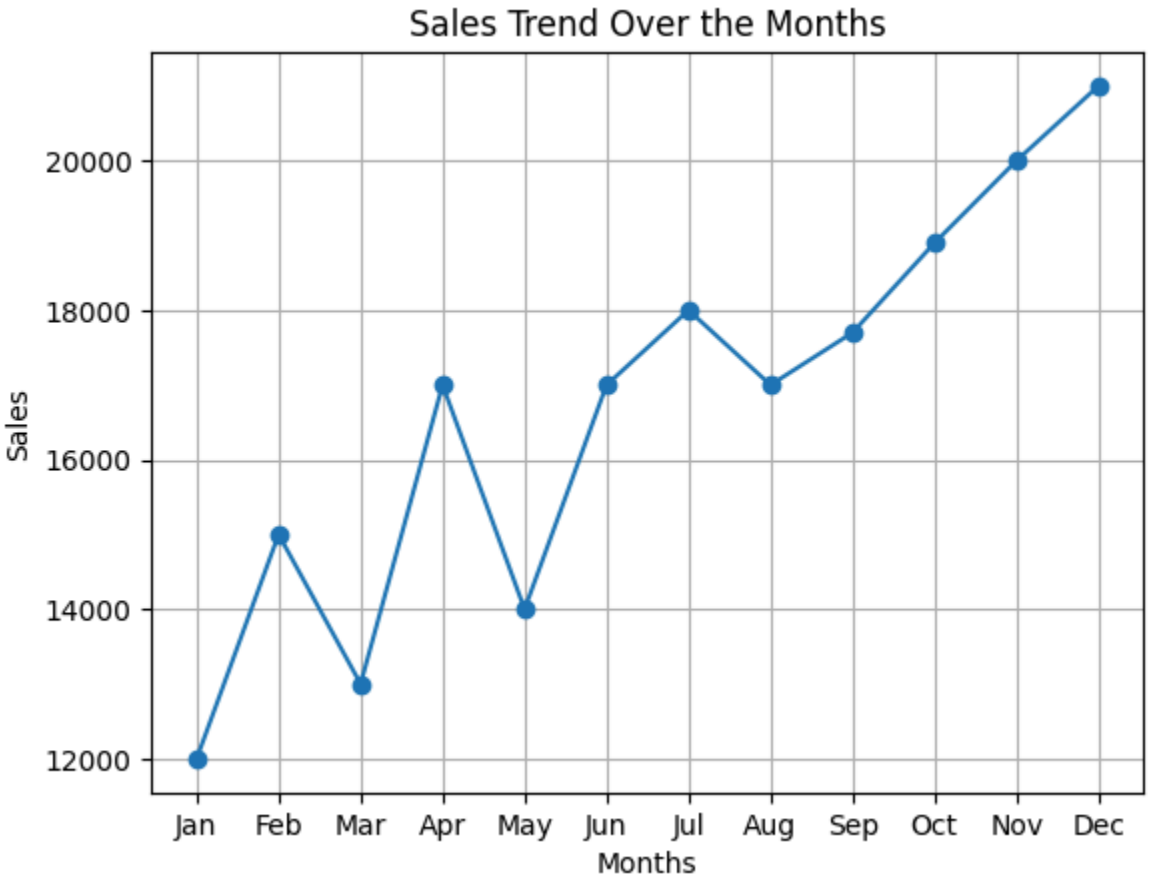
Question 4

In [ ]:
```
# Implement a program that searches for a specific element in a NumPy array ar

import numpy as np

# Search for a specific element in a NumPy array
arr = np.random.randint(1, 100, size=20)
element = 50
index = np.where(arr == element)
print(f'Element {element} found at index: {index}')

# Generate a random NumPy array and sort it
random_arr = np.random.randint(1, 100, size=20)
sorted_arr_asc = np.sort(random_arr)
sorted_arr_desc = np.sort(random_arr)[::-1]

print('Original Array:', random_arr)
print('Sorted Array (Ascending):', sorted_arr_asc)
print('Sorted Array (Descending):', sorted_arr_desc)
```

Question 5

In [ ]:
```
# You are a data analyst working for a retail company. The company wants to vi
# Write a Python program that:
```

```
# 1. Create a line chart to show the trend of sales over the months,
# 2. Add markers, a grid, labels for the x and y axes, and a title to the char

months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct'
sales = [12000, 15000, 13000, 17000, 14000, 17000, 18000, 17000, 17700, 18900,

plt.plot(months, sales, marker='o')
plt.xlabel('Months')
plt.ylabel('Sales')
plt.title('Sales Trend Over the Months')
plt.grid(True)
plt.show()
```

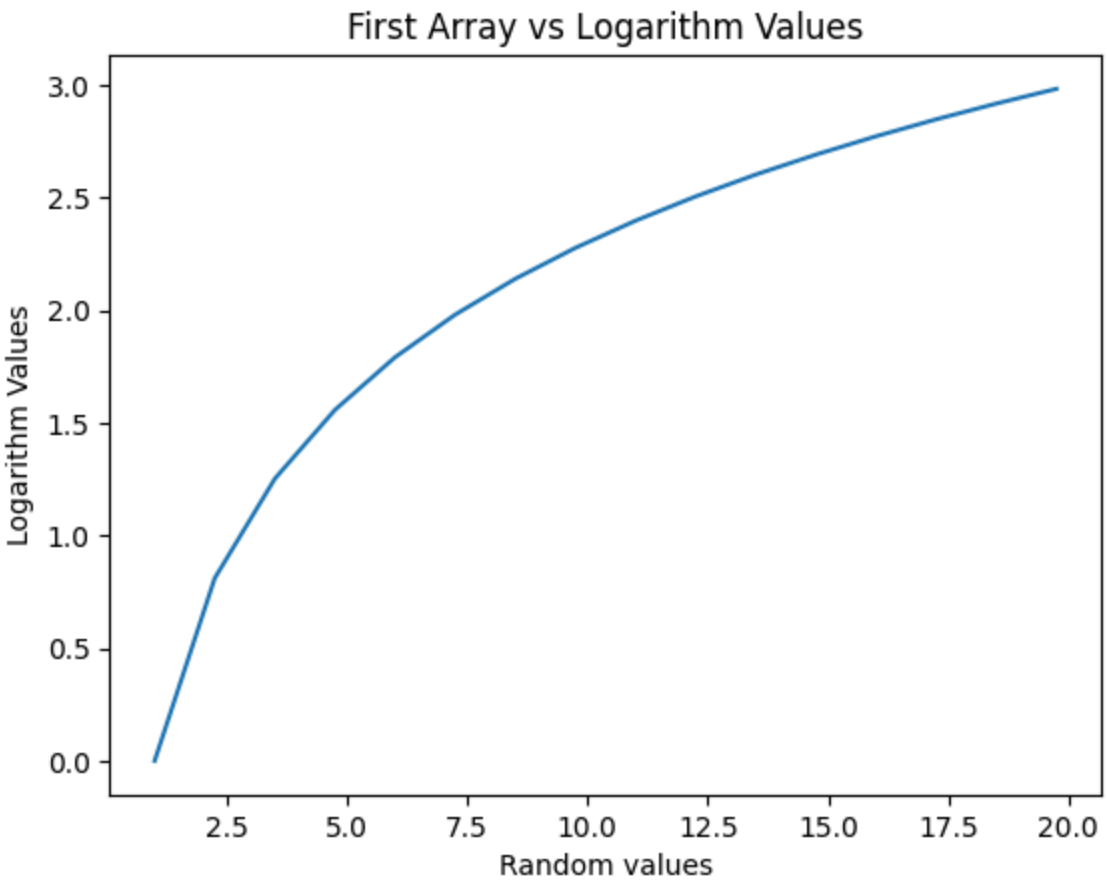


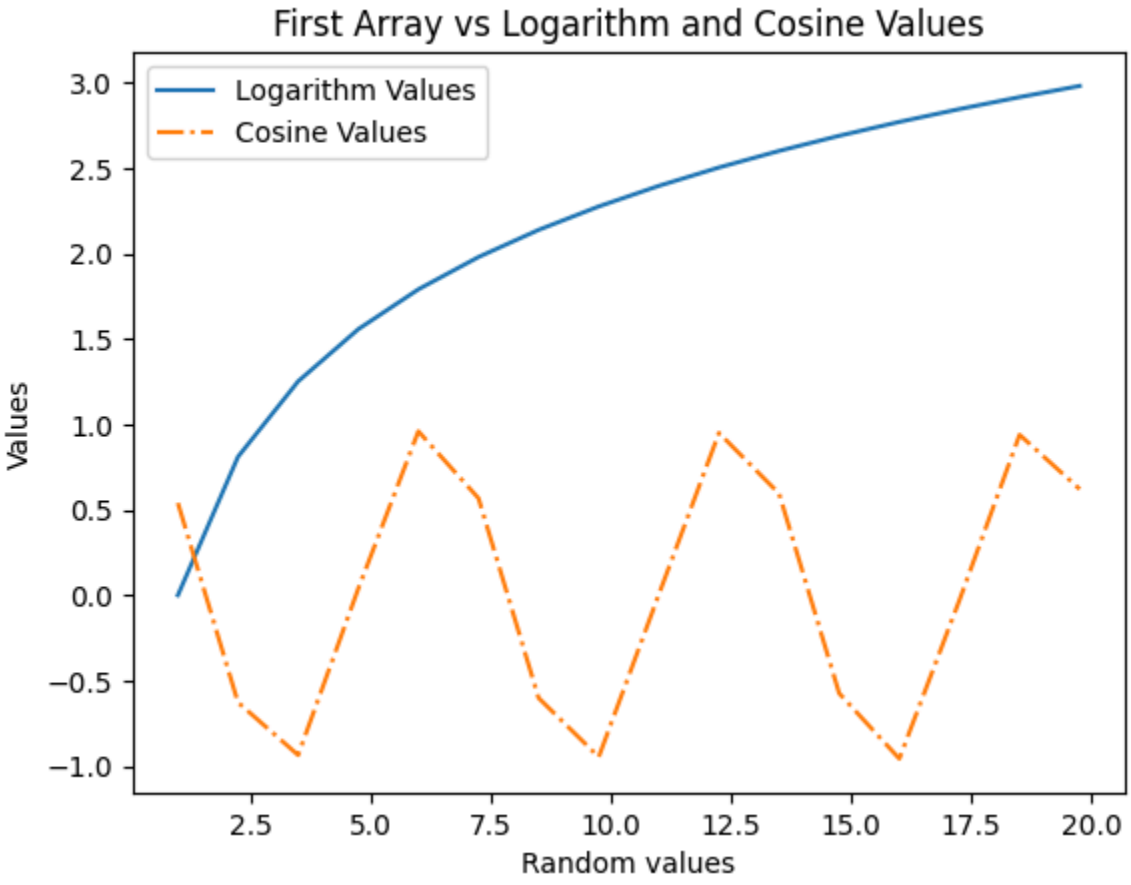
Question 6

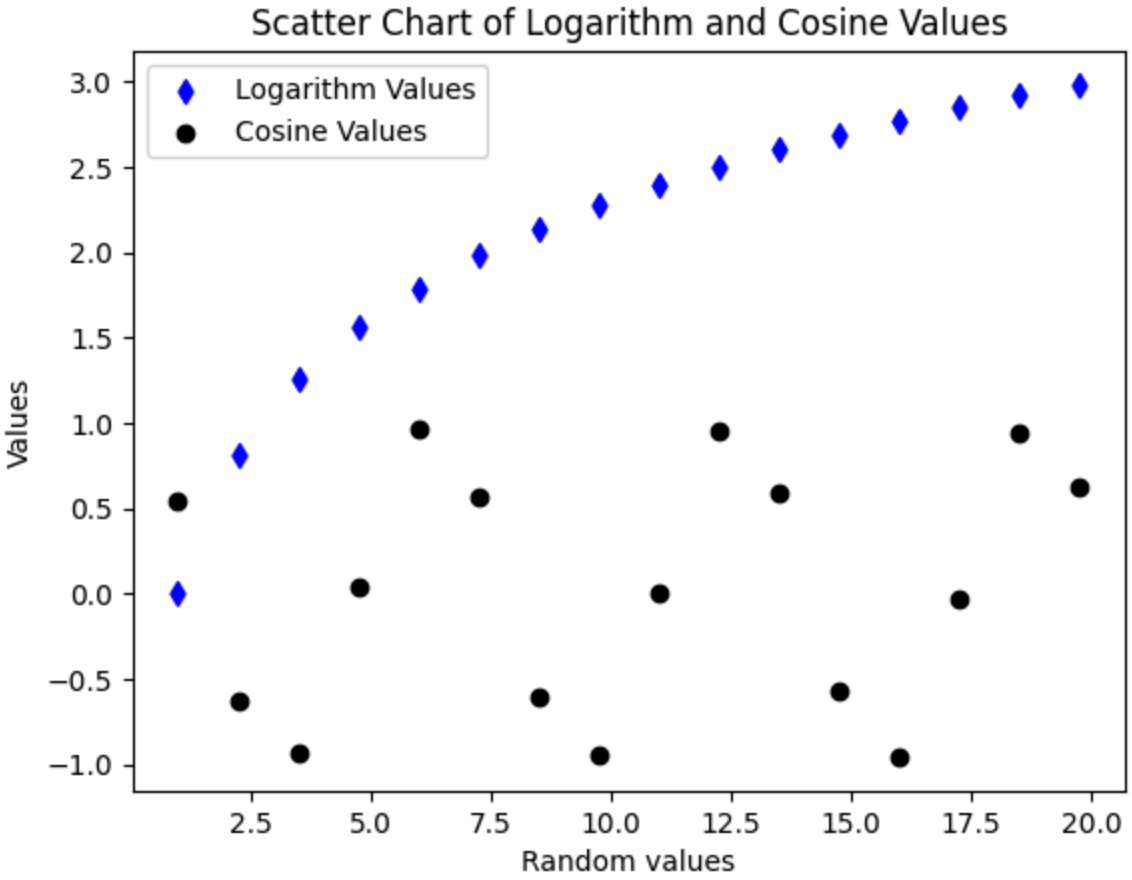
In [ ]:

```
# Create an array in the range 1 to 20 with values 1.25 apart. Another array c
# a) Create a plot of first vs second array: specify the x-axis(containing fir
# b) Create a third array that stores the cos value of first array and then pl
# c) Create scatterchart as this: second array data points as blue small diamc

import numpy as np
import matplotlib.pyplot as plt

# Create an array in the range 1 to 20 with values 1.25 apart
first_array = np.arange(1, 20, 1.25)
second_array = np.log(first_array)
```

```
# Plot first vs second array
plt.plot(first_array, second_array)
plt.xlabel('Random values')
plt.ylabel('Logarithm Values')
plt.title('First Array vs Logarithm Values')
plt.show()

# Create a third array that stores the cos value of first array
third_array = np.cos(first_array)

# Plot both the second and third arrays vs first array
plt.plot(first_array, second_array, label='Logarithm Values')
plt.plot(first_array, third_array, linestyle='-.', label='Cosine Values')
plt.xlabel('Random values')
plt.ylabel('Values')
plt.title('First Array vs Logarithm and Cosine Values')
plt.legend()
plt.show()

# Create scatterchart
plt.scatter(first_array, second_array, color='blue', marker='d', label='Logari
plt.scatter(first_array, third_array, color='black', marker='o', label='Cosine
plt.xlabel('Random values')
plt.ylabel('Values')
plt.title('Scatter Chart of Logarithm and Cosine Values')
plt.legend()
plt.show()
```


First Array vs Logarithm Values
Logarithm Values
Random values
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


First Array vs Logarithm and Cosine Values
Logarithm Values
Cosine Values
Values
Random values
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0



Question 7

In [ ]:
```python
# Create a simple Python package named mypackage with a module math_operations
# Write a script that imports this function and uses it to add two numbers.

# Directory structure:
# mypackage/
# ├── __init__.py
# └── math_operations.py

# math_operations.py
def add(a, b):
    return a + b

# Script to use the function
from mypackage.math_operations import add

result = add(5, 3)
print(f'The sum is: {result}')
```

Question 8

In [ ]:
```python
# Create a subplot with a bar chart on the left and a pie chart on the right.
# Products: ["A", "B", "C", "D", "E"]
# Sales: [15, 30, 25, 10, 20]
```

```
import matplotlib.pyplot as plt

products = ["A", "B", "C", "D", "E"]
sales = [15, 30, 25, 10, 20]

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))

# Bar chart
ax1.bar(products, sales, color='skyblue')
ax1.set_xlabel('Products')
ax1.set_ylabel('Sales')
ax1.set_title('Sales of Products')

# Pie chart
ax2.pie(sales, labels=products, autopct='%1.1f%%', startangle=140)
ax2.set_title('Market Share of Products')

plt.show()
```

Question 9

In [ ]:

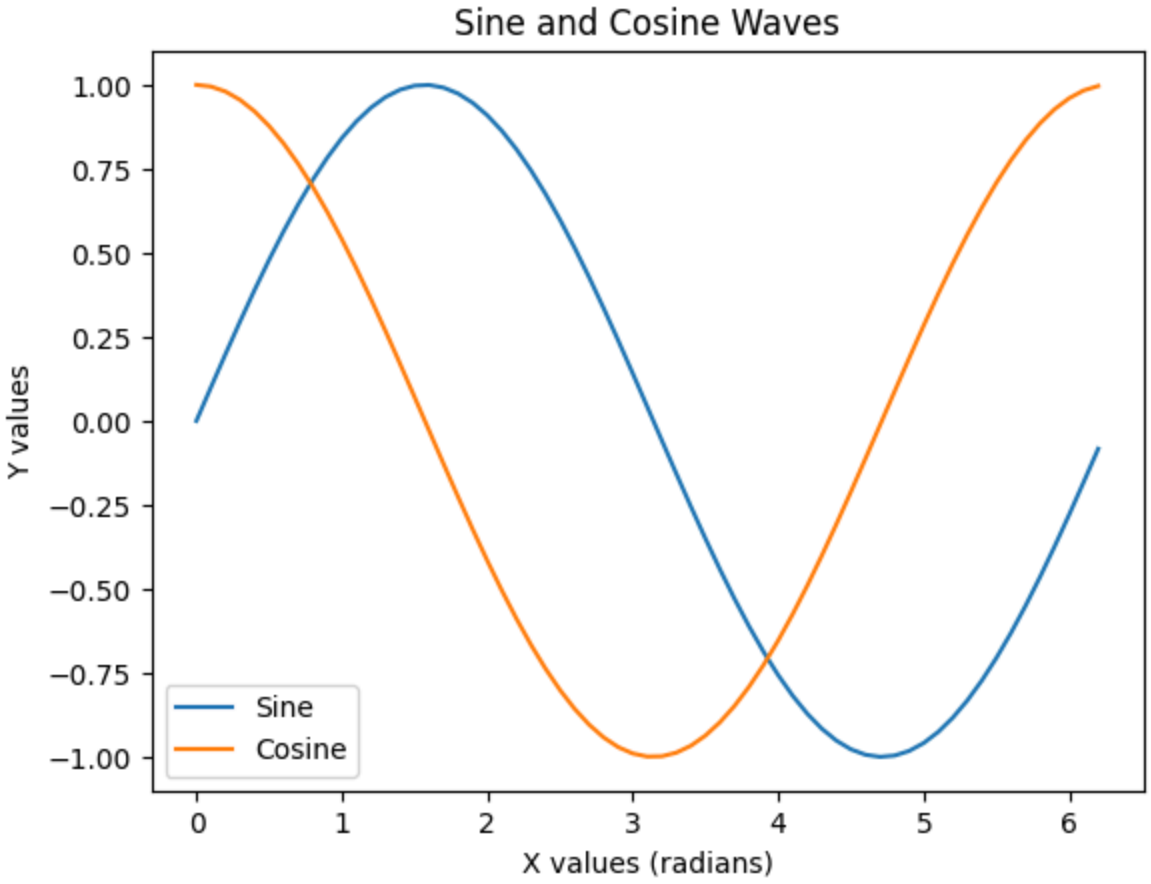
```
# Create a 2x2 grid of subplots in Matplotlib, with each subplot displaying a
# Data Set:
# Line chart and Scatter plot:
# x = [5, 7, 8, 7, 2, 17, 2, 9, 4, 11], y = [99, 86, 87, 88, 100, 86, 103, 87,
# Histogram Data: data = [22, 87, 5, 43, 56, 73, 55, 54, 11, 20, 51, 5, 79, 31
# Bar Chart Data: categories = ['A', 'B', 'C', 'D'], values = [5, 7, 3, 8]

import matplotlib.pyplot as plt

x = [5, 7, 8, 7, 2, 17, 2, 9, 4, 11]
y = [99, 86, 87, 88, 100, 86, 103, 87, 94, 78]
data = [22, 87, 5, 43, 56, 73, 55, 54, 11, 20, 51, 5, 79, 31, 27]
categories = ['A', 'B', 'C', 'D']
values = [5, 7, 3, 8]

fig, axs = plt.subplots(2, 2, figsize=(10, 10))

# Line chart
axs[0, 0].plot(x, y)
axs[0, 0].set_title('Line Chart')

# Scatter plot
axs[0, 1].scatter(x, y)
axs[0, 1].set_title('Scatter Plot')

# Histogram
axs[1, 0].hist(data, bins=5, edgecolor='black')
axs[1, 0].set_title('Histogram')

# Bar chart
axs[1, 1].bar(categories, values, color='skyblue')
```

```
axs[1, 1].set_title('Bar Chart')

plt.tight_layout()
plt.show()
```

Question 10

In [21]:

```
# Write a Python program that performs the following tasks:
# 1. Generate Data:
# ○ Use NumPy to create a range of values (x) from 0 to 2π (approximately 6.28
# ○ Compute the sine and cosine values for each x value and store them in sepa
# 2. Plotting:
# ○ Use Matplotlib to create a line plot that displays both the sine and cosin
# ○ Label the x-axis as "X values (radians)", the y-axis as "Y values", and gi

import numpy as np
import matplotlib.pyplot as plt

x = np.arange(0, 2 * np.pi, 0.1)
sine_values = np.sin(x)
cosine_values = np.cos(x)

plt.plot(x, sine_values, label='Sine')
plt.plot(x, cosine_values, label='Cosine')
plt.xlabel('X values (radians)')
plt.ylabel('Y values')
plt.title('Sine and Cosine Waves')
plt.legend()
plt.show()
```

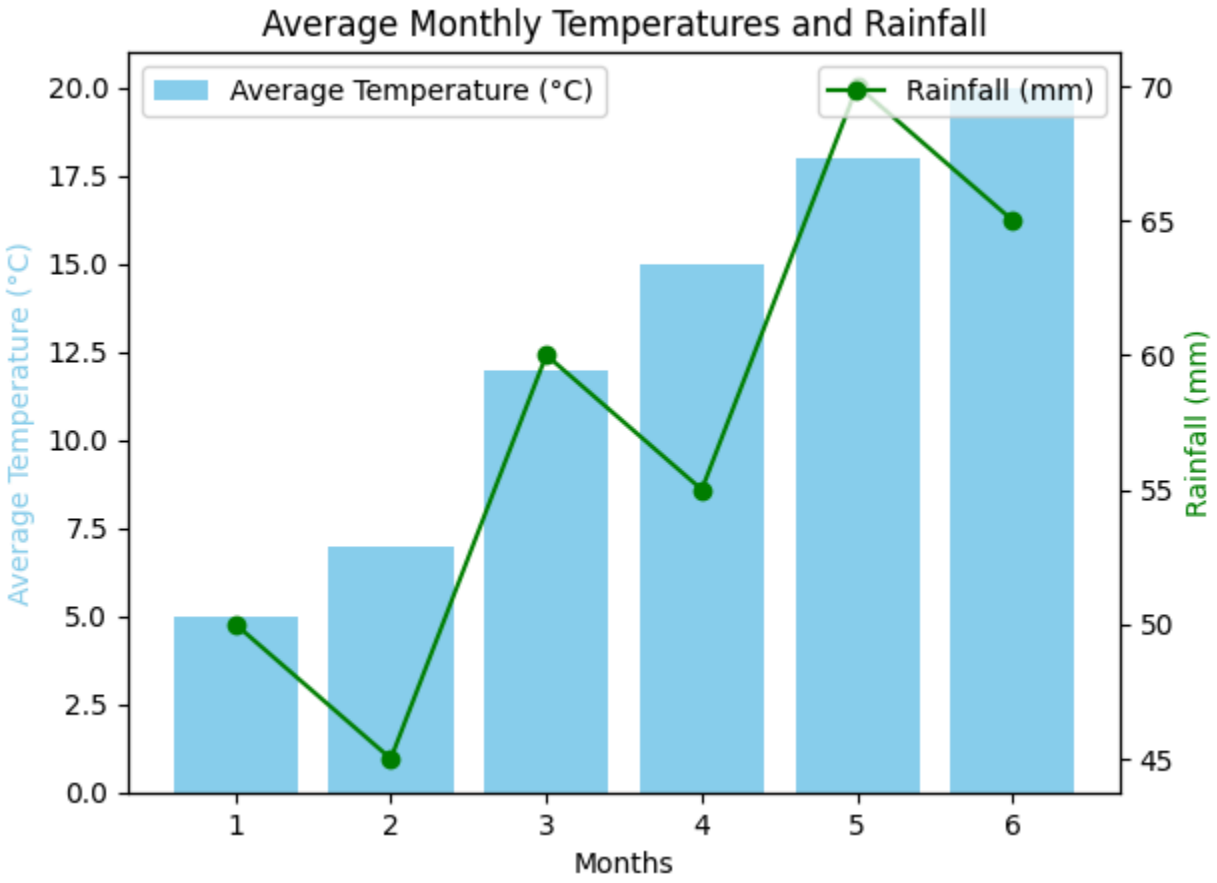


Question 11

In [ ]:

```
# Create two subplots that share the same x-axis. The first subplot should be
# Months: [1, 2, 3, 4, 5, 6]
# Average Temperatures (°C): [5, 7, 12, 15, 18, 20]
# Rainfall (mm): [50, 45, 60, 55, 70, 65]

import matplotlib.pyplot as plt

months = [1, 2, 3, 4, 5, 6]
avg_temps = [5, 7, 12, 15, 18, 20]
rainfall = [50, 45, 60, 55, 70, 65]

fig, ax1 = plt.subplots()

ax2 = ax1.twinx()
ax1.bar(months, avg_temps, color='skyblue', label='Average Temperature (°C)')
ax2.plot(months, rainfall, color='green', marker='o', label='Rainfall (mm)')

ax1.set_xlabel('Months')
ax1.set_ylabel('Average Temperature (°C)', color='skyblue')
ax2.set_ylabel('Rainfall (mm)', color='green')

ax1.set_title('Average Monthly Temperatures and Rainfall')
ax1.legend(loc='upper left')
```

```
ax2.legend(loc='upper right')

plt.show()
```



Question 12

In [ ]:
```
# a) WAP to create a 2D NumPy array of shape (4, 5) containing integers from 1
# b) WAP to create a NumPy array with random integers between 1 and 100. Sort

import numpy as np

# Part a
array_2d = np.arange(1, 21).reshape(4, 5)
reshaped_array = array_2d.reshape(5, 4)
print("Original array:\n", array_2d)
print("Reshaped array:\n", reshaped_array)

# Part b
random_array = np.random.randint(1, 101, size=20)
sorted_array = np.sort(random_array)
top_5_elements = sorted_array[-5:]
print("Sorted array:\n", sorted_array)
print("Top 5 largest elements:\n", top_5_elements)
```

Question 13

In [ ]:
```python
# Write a Python program to create bar plots with error bars on the same figur
# Sample Data:
# Mean velocity: 0.2474, 0.1235, 0.1737, 0.1824
# Standard deviation of velocity: 0.3314, 0.2278, 0.2836, 0.2645

import matplotlib.pyplot as plt
import numpy as np

mean_velocity = [0.2474, 0.1235, 0.1737, 0.1824]
std_deviation = [0.3314, 0.2278, 0.2836, 0.2645]
x_pos = np.arange(len(mean_velocity))

plt.bar(x_pos, mean_velocity, yerr=std_deviation, capsize=5, color='skyblue')
plt.xlabel('Sample')
plt.ylabel('Mean Velocity')
plt.title('Bar Plot with Error Bars')
plt.show()
```

Question 14

In [ ]:
```python
# 300 children were asked to choose their favorite ice cream flavor.
# WAP to show this data on a Pie chart with the percentage of children choosin
# Flavor Frequency:
# Strawberry 44
# Vanilla 76
# Chocolate 30
# Butterscotch 78
# Raspberry 39
# Mint 11
# Blueberry 22

import matplotlib.pyplot as plt

flavors = ['Strawberry', 'Vanilla', 'Chocolate', 'Butterscotch', 'Raspberry',
frequencies = [44, 76, 30, 78, 39, 11, 22]

plt.pie(frequencies, labels=flavors, autopct='%1.1f%%', startangle=140)
plt.title('Favorite Ice Cream Flavors')
plt.show()
```

Question 15

In [ ]:
```python
# Create an array of prime numbers between 2 and 1000. Create another array of
# Truncate the larger array to make it the same size as the smaller array. The

import numpy as np

def is_prime(num):
    if num < 2:
        return False
    for i in range(2, int(np.sqrt(num)) + 1):
        if num % i == 0:
```

```
            return False
    return True

primes_2_1000 = np.array([x for x in range(2, 1001) if is_prime(x)])
primes_2000_4000 = np.array([x for x in range(2000, 4001) if is_prime(x)])

# Truncate the larger array
min_length = min(len(primes_2_1000), len(primes_2000_4000))
primes_2000_4000 = primes_2000_4000[:min_length]

# Find the correlation
correlation = np.corrcoef(primes_2_1000, primes_2000_4000)[0, 1]
print("Correlation between the two arrays:", correlation)
```

Question 16

In [20]:

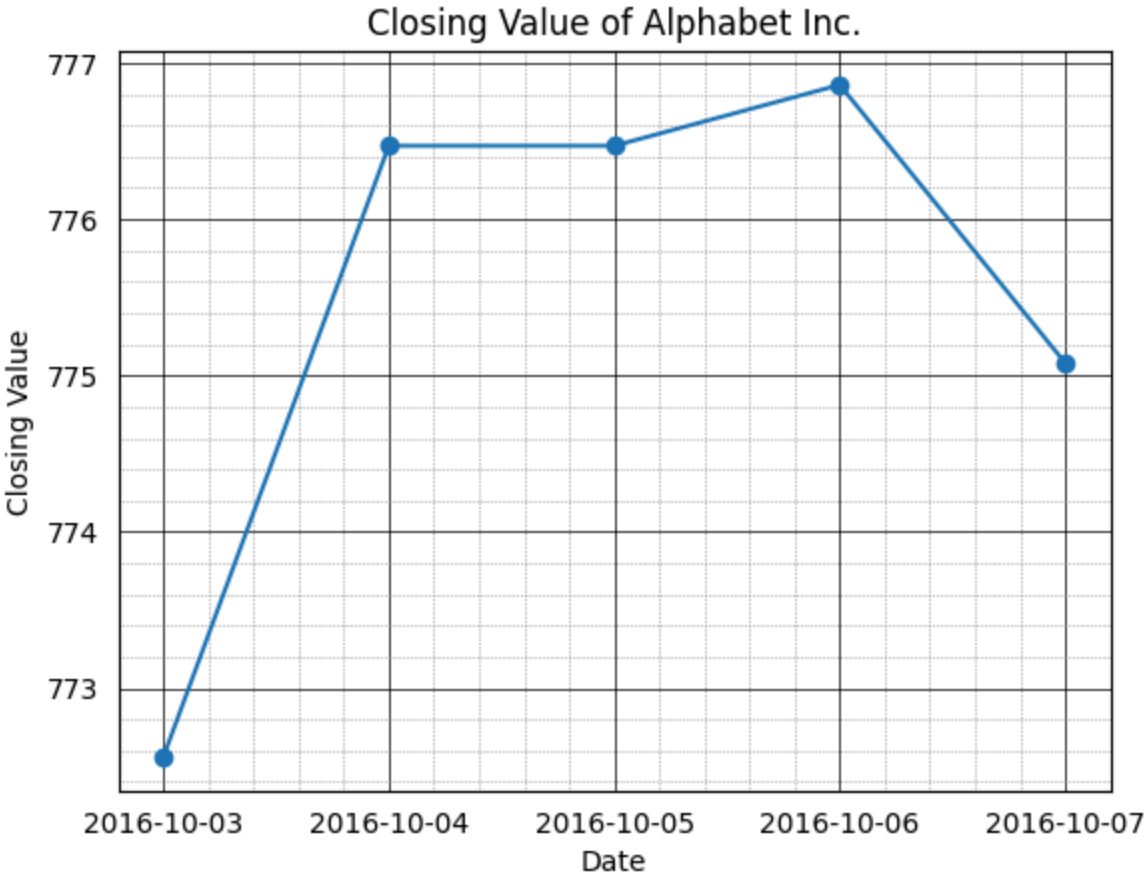
```
# Write a Python program to create a NumPy array that contains the names of fi
# 1. Create another NumPy array for their corresponding scores in Mathematics.
# 2. Calculate the average score of the students in Mathematics. Print the ave
# 3. Find out students who scored above a specified 75 in Mathematics using bo
# 4. Sort the scores in descending order. Display the sorted scores along with
# 5. Find and print the highest and lowest scores in the scores array along wi

import numpy as np

# Create arrays
students = np.array(['Alice', 'Bob', 'Charlie', 'David', 'Eve'])
scores = np.array([85, 92, 78, 65, 88])

# Print arrays
print("Students:", students)
print("Scores:", scores)

# Calculate and print average score
average_score = np.mean(scores)
print("Average Score:", average_score)

# Find students who scored above 75
above_75 = students[scores > 75]
print("Students scoring above 75:", above_75)

# Sort scores in descending order and display with corresponding students
sorted_indices = np.argsort(scores)[::-1]
sorted_students = students[sorted_indices]
sorted_scores = scores[sorted_indices]
print("Sorted Scores and Students:")
for student, score in zip(sorted_students, sorted_scores):
    print(f"{student}: {score}")

# Find and print highest and lowest scores with names
highest_score = np.max(scores)
lowest_score = np.min(scores)
highest_scorer = students[scores == highest_score][0]
```

```
lowest_scorer = students[scores == lowest_score][0]
print(f"Highest Score: {highest_score} by {highest_scorer}")
print(f"Lowest Score: {lowest_score} by {lowest_scorer}")
```

```
Students: ['Alice' 'Bob' 'Charlie' 'David' 'Eve']
Scores: [85 92 78 65 88]
Average Score: 81.6
Students scoring above 75: ['Alice' 'Bob' 'Charlie' 'Eve']
Sorted Scores and Students:
Bob: 92
Eve: 88
Alice: 85
Charlie: 78
David: 65
Highest Score: 92 by Bob
Lowest Score: 65 by David
```

Question 17

In [19]:
```
# Write a Python program to display the grid and draw line charts of the closi
# Customize the grid lines with rendering with a larger grid (major grid) and

import matplotlib.pyplot as plt

dates = ['2016-10-03', '2016-10-04', '2016-10-05', '2016-10-06', '2016-10-07']
closing_values = [772.56, 776.47, 776.47, 776.86, 775.08]

plt.plot(dates, closing_values, marker='o')
plt.xlabel('Date')
plt.ylabel('Closing Value')
plt.title('Closing Value of Alphabet Inc.')

# Customize grid
plt.grid(which='major', linestyle='-', linewidth='0.5', color='black')
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='gray')
plt.minorticks_on()
plt.tick_params(which='both', bottom=False, left=False)

plt.show()
```



Question 18

In [ ]:
```
# Write a Python program to add and subtract two numpy arrays.
# For the given sample numpy array retrieve and display only those elements wh
# Sample array: [2, 4, 6, 8, 10], [1, 3, 5, 7, 9]

import numpy as np

array1 = np.array([2, 4, 6, 8, 10])
array2 = np.array([1, 3, 5, 7, 9])

# Add and subtract arrays
sum_array = np.add(array1, array2)
diff_array = np.subtract(array1, array2)

print("Sum of arrays:", sum_array)
print("Difference of arrays:", diff_array)

# Retrieve and display elements between 4 and 9
filtered_elements = array1[(array1 > 4) & (array1 < 9)]
print("Elements between 4 and 9:", filtered_elements)
```

Question 19

In [ ]:
```
# Write a program to create two NumPy arrays of shape (3, 3) with random integ
```

```python
# Perform element-wise addition, subtraction, multiplication, and division bet

import numpy as np

array1 = np.random.randint(1, 11, size=(3, 3))
array2 = np.random.randint(1, 11, size=(3, 3))

# Element-wise operations
addition = np.add(array1, array2)
subtraction = np.subtract(array1, array2)
multiplication = np.multiply(array1, array2)
division = np.divide(array1, array2)

print("Array 1:\n", array1)
print("Array 2:\n", array2)
print("Addition:\n", addition)
print("Subtraction:\n", subtraction)
print("Multiplication:\n", multiplication)
print("Division:\n", division)
```

Question 20

In [ ]:

```python
# Write one Python program to find the following from the given dataframe DF:
# Rollno Name Age Marks
# S1001 Arun 18 68
# S1002 Mohit 14 47
# S1003 Karan 13 78
# S1004 Lalit 16 87
# S1005 Ravi 14 60
# a) Maximum marks and minimum marks
# b) Sum of all the marks
# c) Mean and mode of age of the students
# d) Count the number of rows present in the dataframe

import pandas as pd
from scipy import stats

data = {
    'Rollno': ['S1001', 'S1002', 'S1003', 'S1004', 'S1005'],
    'Name': ['Arun', 'Mohit', 'Karan', 'Lalit', 'Ravi'],
    'Age': [18, 14, 13, 16, 14],
    'Marks': [68, 47, 78, 87, 60]
}

df = pd.DataFrame(data)

# a) Maximum and minimum marks
max_marks = df['Marks'].max()
min_marks = df['Marks'].min()
print(f"Maximum Marks: {max_marks}")
print(f"Minimum Marks: {min_marks}")

# b) Sum of all the marks
```

```
total_marks = df['Marks'].sum()
print(f"Total Marks: {total_marks}")

# c) Mean and mode of age
mean_age = df['Age'].mean()
mode_age = stats.mode(df['Age'])[0][0]
print(f"Mean Age: {mean_age}")
print(f"Mode Age: {mode_age}")

# d) Count the number of rows
row_count = len(df)
print(f"Number of Rows: {row_count}")
```

Question 21

In [ ]:
```
# Demonstrate your understanding of NumPy operations including array creation,

import numpy as np

# Array creation
array = np.random.randint(1, 100, size=(5, 5))
print("Original Array:\n", array)

# Statistical analysis
mean = np.mean(array)
std_dev = np.std(array)
print("Mean:", mean)
print("Standard Deviation:", std_dev)

# Reshaping
reshaped_array = array.reshape(25)
print("Reshaped Array:\n", reshaped_array)

# Filtering
filtered_array = array[array > 50]
print("Filtered Array (values > 50):\n", filtered_array)

# Mathematical operations
squared_array = np.square(array)
print("Squared Array:\n", squared_array)
```

Question 22

In [ ]:
```
# Create a Python program using Matplotlib to generate a Bar chart displaying
# Years: [2019, 2020, 2021, 2022, 2023]
# Heatwave Days: [15, 20, 25, 30, 35]

import matplotlib.pyplot as plt

years = [2019, 2020, 2021, 2022, 2023]
heatwave_days = [15, 20, 25, 30, 35]
```

```
plt.bar(years, heatwave_days, color='orange')
plt.xlabel('Years')
plt.ylabel('Heatwave Days')
plt.title('Heatwave Days During Summer (March to May)')
plt.show()
```

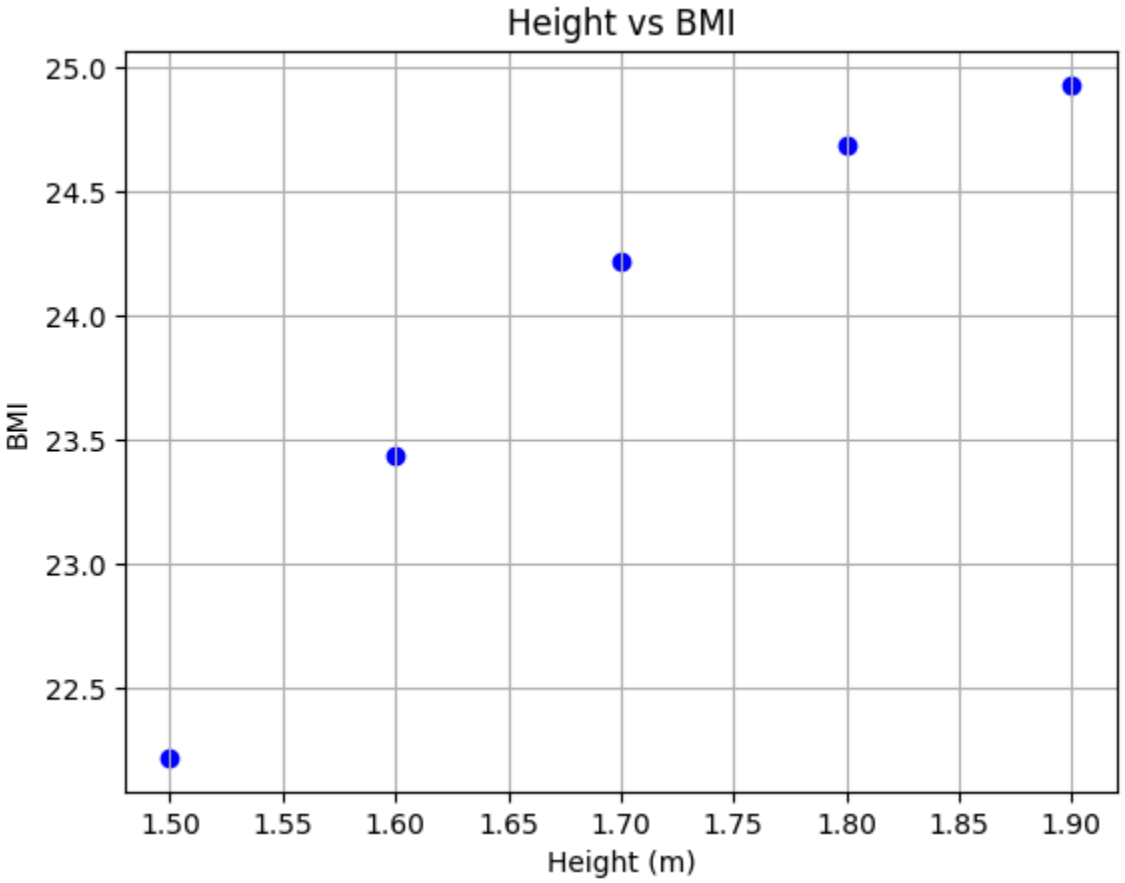
Question 23

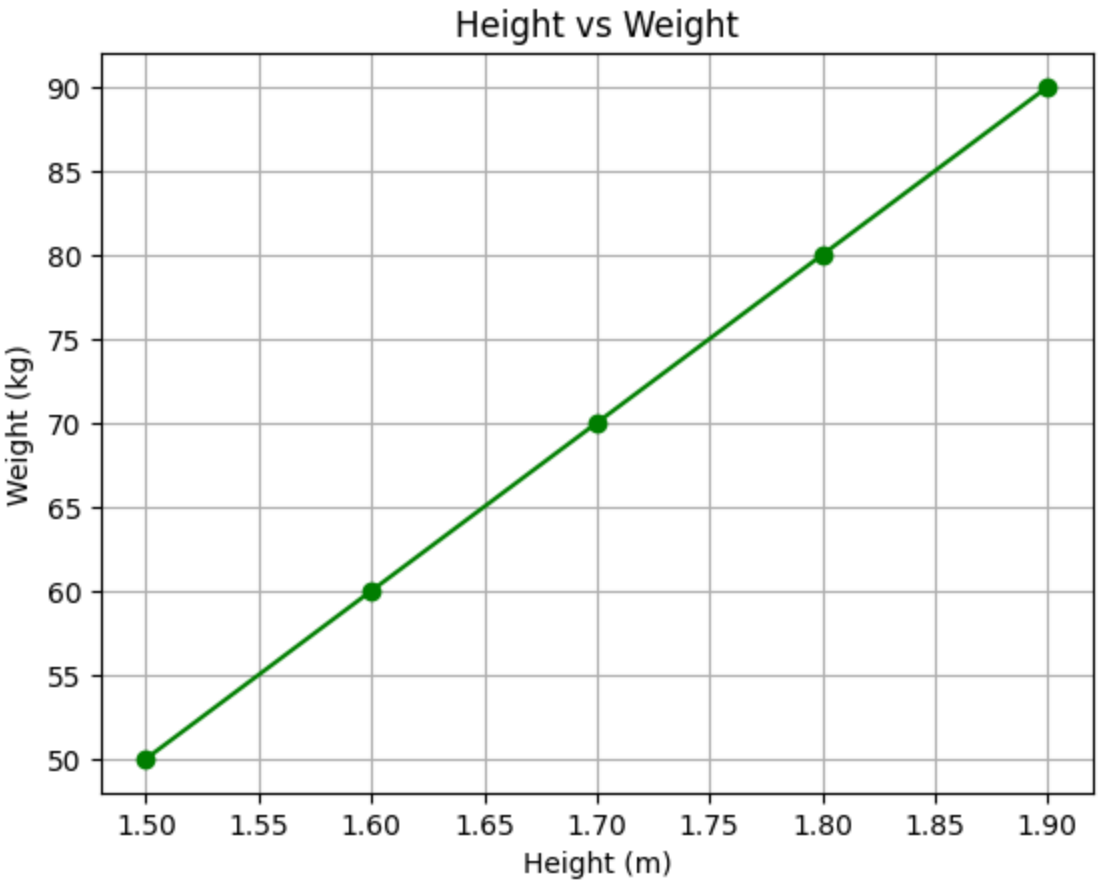
In [ ]:
```
# Read the company_sales_data file using Pandas or NumPy or using in-built mat
# Calculate total sale data for last year for each product and show it using a

import pandas as pd
import matplotlib.pyplot as plt

# Assuming the company_sales_data file is a CSV file
data = pd.read_csv('company_sales_data.csv')

# Calculate total sales for the last year
total_sales = data.groupby('Product')['Sales'].sum()

# Plot pie chart
plt.pie(total_sales, labels=total_sales.index, autopct='%1.1f%%', startangle=1
plt.title('Total Sales Data for Last Year')
plt.show()
```

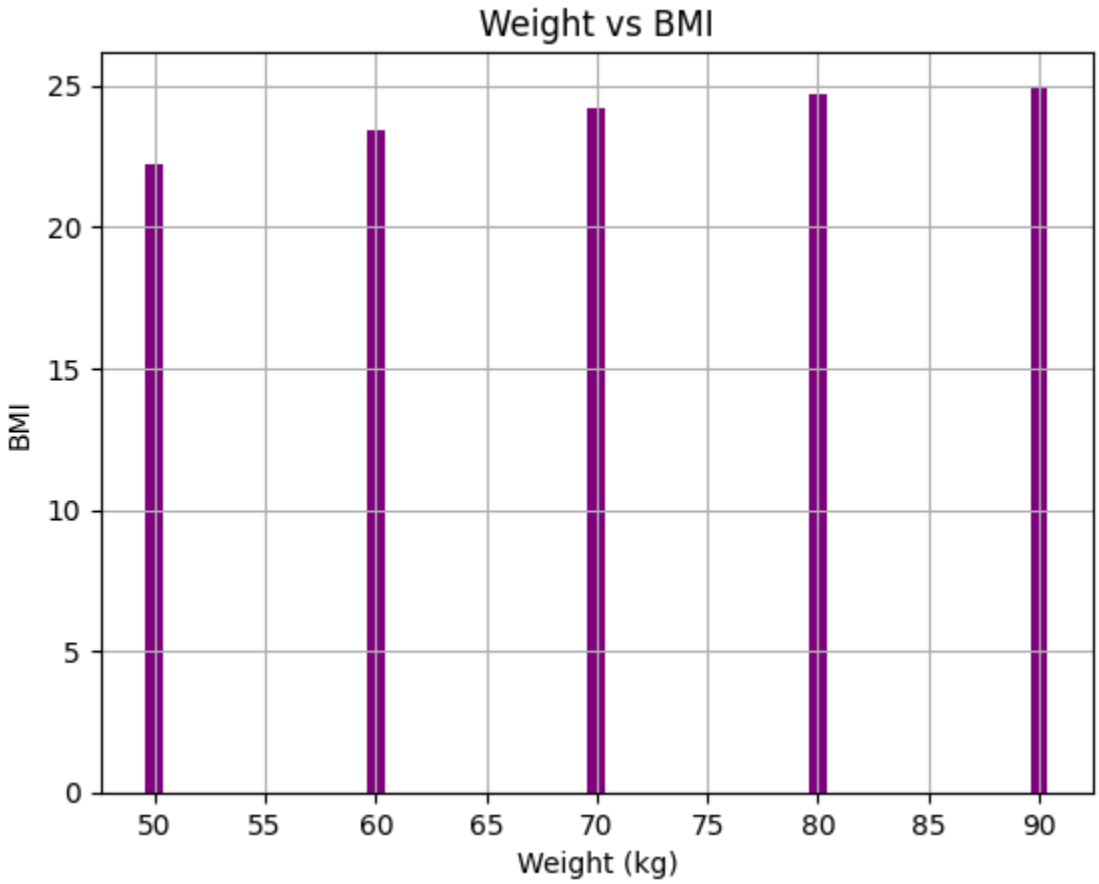
Question 24

In [18]:
```
# Create two single dimensional NumPy arrays, one is height, and another is we
# Create a scatter plot height and BMI, line plot of Height and weight, Bar pl
# Use proper formatting of x-label, y-label, title, color, and grid.

import numpy as np
import matplotlib.pyplot as plt

height = np.array([1.5, 1.6, 1.7, 1.8, 1.9])
weight = np.array([50, 60, 70, 80, 90])
bmi = weight / height**2

# Scatter plot of height and BMI
plt.scatter(height, bmi, color='blue')
plt.xlabel('Height (m)')
plt.ylabel('BMI')
plt.title('Height vs BMI')
plt.grid(True)
plt.show()

# Line plot of height and weight
plt.plot(height, weight, marker='o', color='green')
plt.xlabel('Height (m)')
plt.ylabel('Weight (kg)')
plt.title('Height vs Weight')
plt.grid(True)
```

```
plt.show()

# Bar plot of weight and BMI
plt.bar(weight, bmi, color='purple')
plt.xlabel('Weight (kg)')
plt.ylabel('BMI')
plt.title('Weight vs BMI')
plt.grid(True)
plt.show()
```




Height vs Weight
Weight (kg)
90
85
80
75
70
65
60
55
50
1.50
1.55
1.60
1.65
1.70
1.75
1.80
1.85
1.90
Height (m)


Weight vs BMI
BMI
25
20
15
10
5
0
50
55
60
65
70
75
80
85
90
Weight (kg)

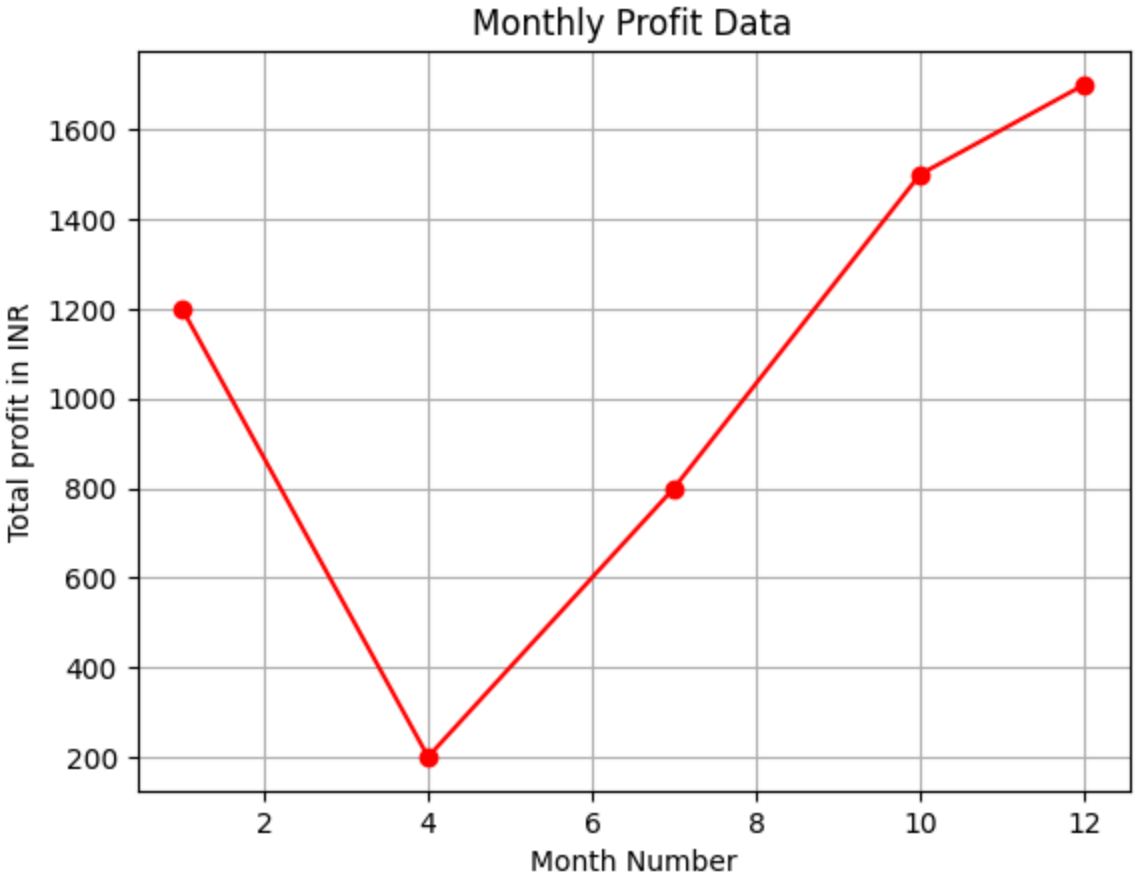
Question 25

In [17]:
```
# Create a filter array that will return only even elements from the original

import numpy as np

original_array = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
even_elements = original_array[original_array % 2 == 0]
print("Even elements:", even_elements)
```

```
Even elements: [ 2  4  6  8 10]
```

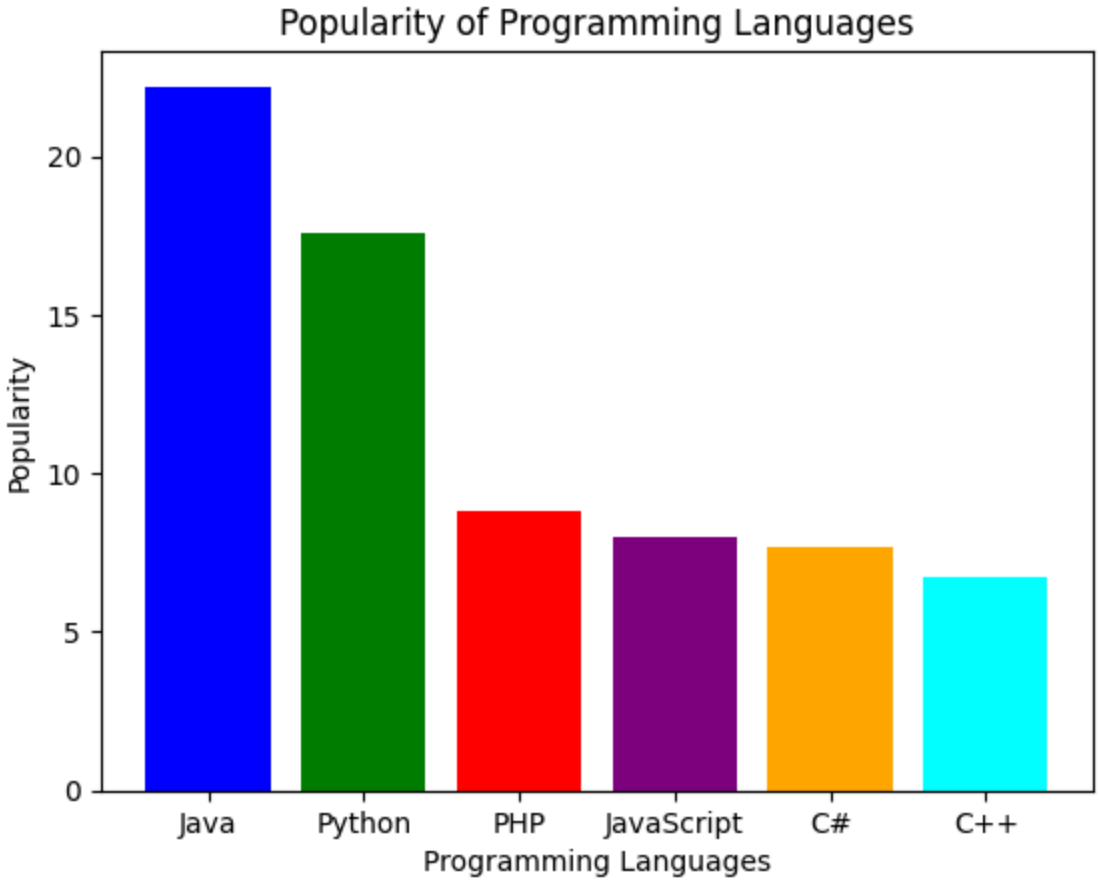
Question 26

In [16]:
```
# Total profit data provided for each month
# Month Number: [1, 4, 7, 10, 12]
# Total Profit: [1200, 200, 800, 1500, 1700]
# Generate line plot with following properties:
# X label name = Month Number
# Y label name = Total profit in INR
# Add a circle marker
# Line marker color as red

import matplotlib.pyplot as plt

month_number = [1, 4, 7, 10, 12]
total_profit = [1200, 200, 800, 1500, 1700]

plt.plot(month_number, total_profit, marker='o', color='red')
plt.xlabel('Month Number')
plt.ylabel('Total profit in INR')
plt.title('Monthly Profit Data')
plt.grid(True)
plt.show()
```



Question 27

In [15]:
```
# Write a Python program to display a bar chart of the popularity of programmi
# Sample data:
# Programming languages: Java, Python, PHP, JavaScript, C#, C++
# Popularity: 22.2, 17.6, 8.8, 8, 7.7, 6.7

import matplotlib.pyplot as plt

languages = ['Java', 'Python', 'PHP', 'JavaScript', 'C#', 'C++']
popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]
colors = ['blue', 'green', 'red', 'purple', 'orange', 'cyan']

plt.bar(languages, popularity, color=colors)
plt.xlabel('Programming Languages')
plt.ylabel('Popularity')
plt.title('Popularity of Programming Languages')
plt.show()
```



Question 28

In [14]:
```
# Create a 6x6 matrix with random integers between 10 and 50.
# Replace all even numbers in the matrix with -1.

import numpy as np

matrix = np.random.randint(10, 51, size=(6, 6))
matrix[matrix % 2 == 0] = -1
print("Modified Matrix:\n", matrix)
```

```
Modified Matrix:
 [[21 15 -1 -1 -1 -1]
 [-1 -1 -1 47 23 -1]
 [13 25 15 49 47 13]
 [31 35 -1 -1 -1 27]
 [15 29 -1 -1 -1 -1]
 [13 -1 41 43 -1 43]]
```

Question 29

In [12]:
```
# Write a Python program using NumPy to create a 3x3 matrix filled with random
# Calculate the matrix's determinant, transpose, and inverse (if it exists). D

import numpy as np
```

```
matrix = np.random.randint(1, 101, size=(3, 3))
print("Original Matrix:\n", matrix)

try:
    determinant = np.linalg.det(matrix)
    transpose = np.transpose(matrix)
    inverse = np.linalg.inv(matrix)
    print("Determinant:", determinant)
    print("Transpose:\n", transpose)
    print("Inverse:\n", inverse)
except np.linalg.LinAlgError:
    print("Matrix is singular and cannot be inverted.")
```

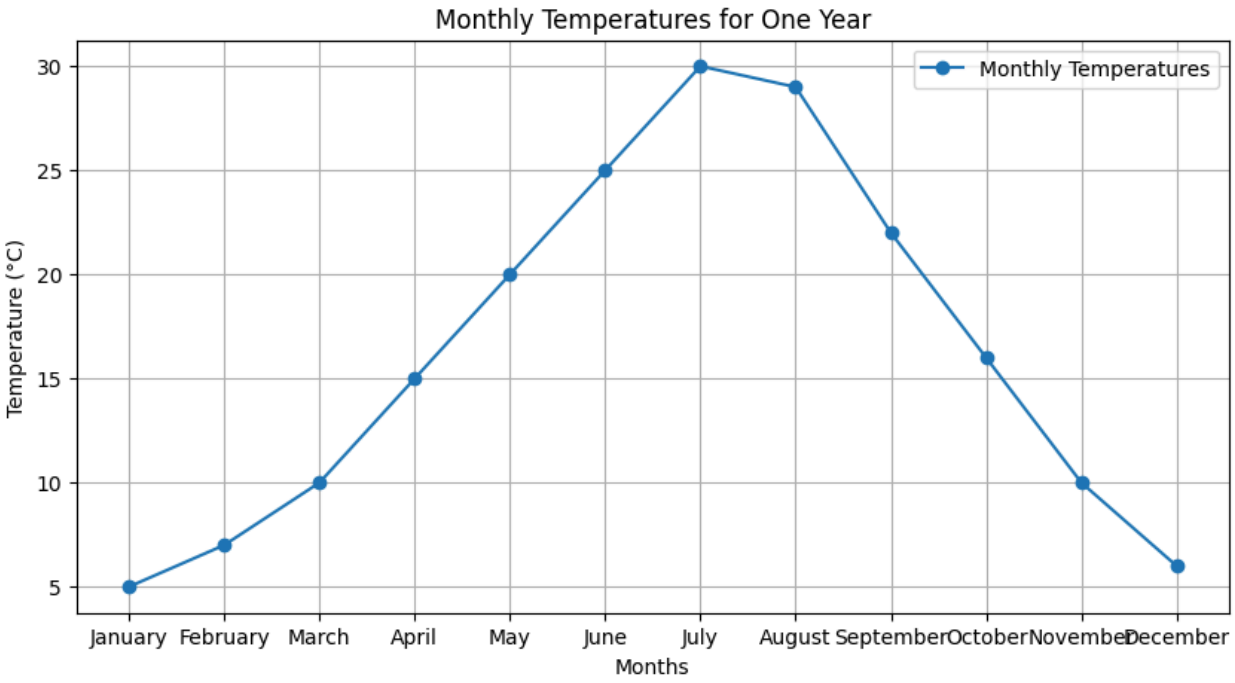
```
Original Matrix:
 [[33 23 83]
 [ 7 98 37]
 [77 52 74]]
Determinant: -366668.9999999997
Transpose:
 [[33  7 77]
 [23 98 52]
 [83 37 74]]
Inverse:
 [[-0.01453082 -0.00712905  0.0198626 ]
 [-0.00635723  0.01076993  0.00174544]
 [ 0.01958715 -0.00015    -0.00838086]]
```

Question 30

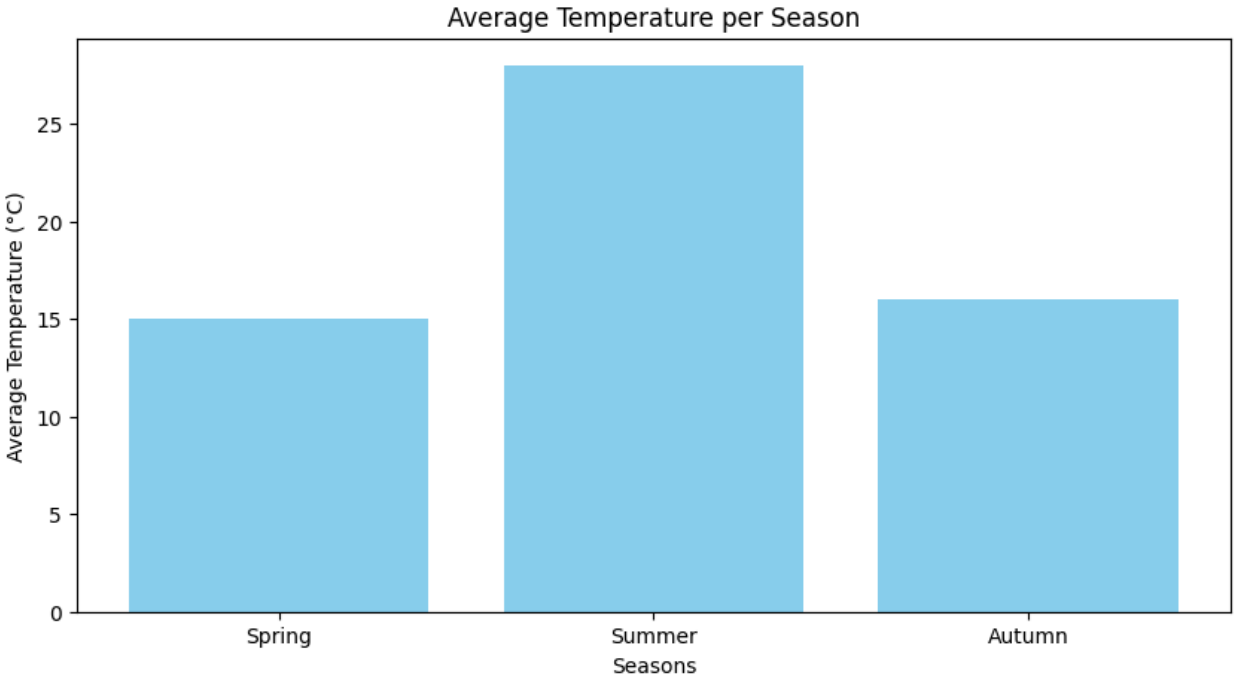
In [11]:

```
# Create a Python program using Matplotlib to generate a line chart displaying
# Additionally, create a subplot with a bar chart showing the average temperat
# months = ["January", "February", "March", "April", "May", "June", "July", "A
# temperatures = [5, 7, 10, 15, 20, 25, 30, 29, 22, 16, 10, 6]
# Hint: Winter (Dec, Jan, Feb), Spring (Mar, Apr, May) Summer (Jun, Jul, Aug)

import matplotlib.pyplot as plt

months = ["January", "February", "March", "April", "May", "June", "July", "Aug
temperatures = [5, 7, 10, 15, 20, 25, 30, 29, 22, 16, 10, 6]

# Line chart for monthly temperatures
plt.figure(figsize=(10, 5))
plt.plot(months, temperatures, marker='o', label='Monthly Temperatures')
plt.xlabel('Months')
plt.ylabel('Temperature (°C)')
plt.title('Monthly Temperatures for One Year')
plt.legend()
plt.grid(True)
plt.show()

# Bar chart for average temperature per season
seasons = ['Winter', 'Spring', 'Summer', 'Autumn']
avg_temperatures = [np.mean(temperatures[11:2]), np.mean(temperatures[2:5]), r
```

```
plt.figure(figsize=(10, 5))
plt.bar(seasons, avg_temperatures, color='skyblue')
plt.xlabel('Seasons')
plt.ylabel('Average Temperature (°C)')
plt.title('Average Temperature per Season')
plt.show()
```



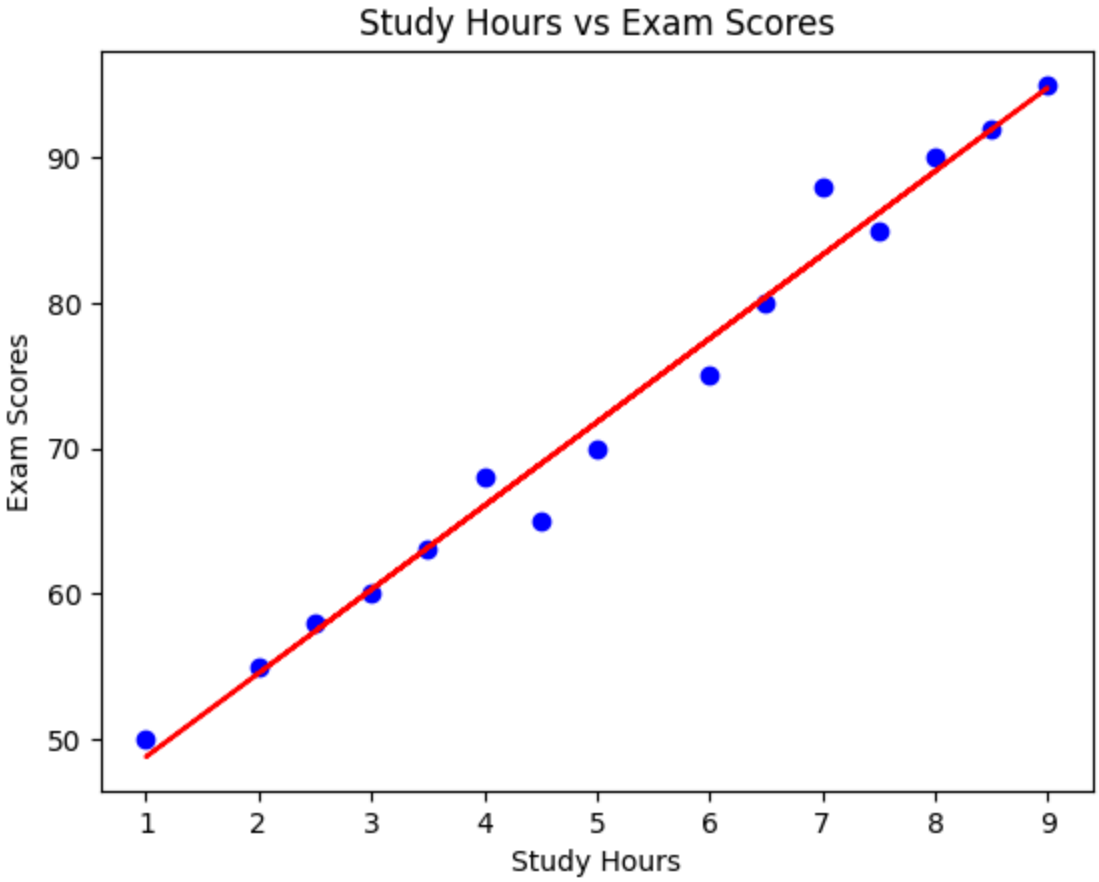

```
/usr/local/lib/python3.10/dist-packages/numpy/core/fromnumeric.py:3504: RuntimeWarning: Mean of empty slice.
  return _methods._mean(a, axis=axis, dtype=dtype,
/usr/local/lib/python3.10/dist-packages/numpy/core/_methods.py:129: RuntimeWarning: invalid value encountered in scalar divide
  ret = ret.dtype.type(ret / rcount)
```

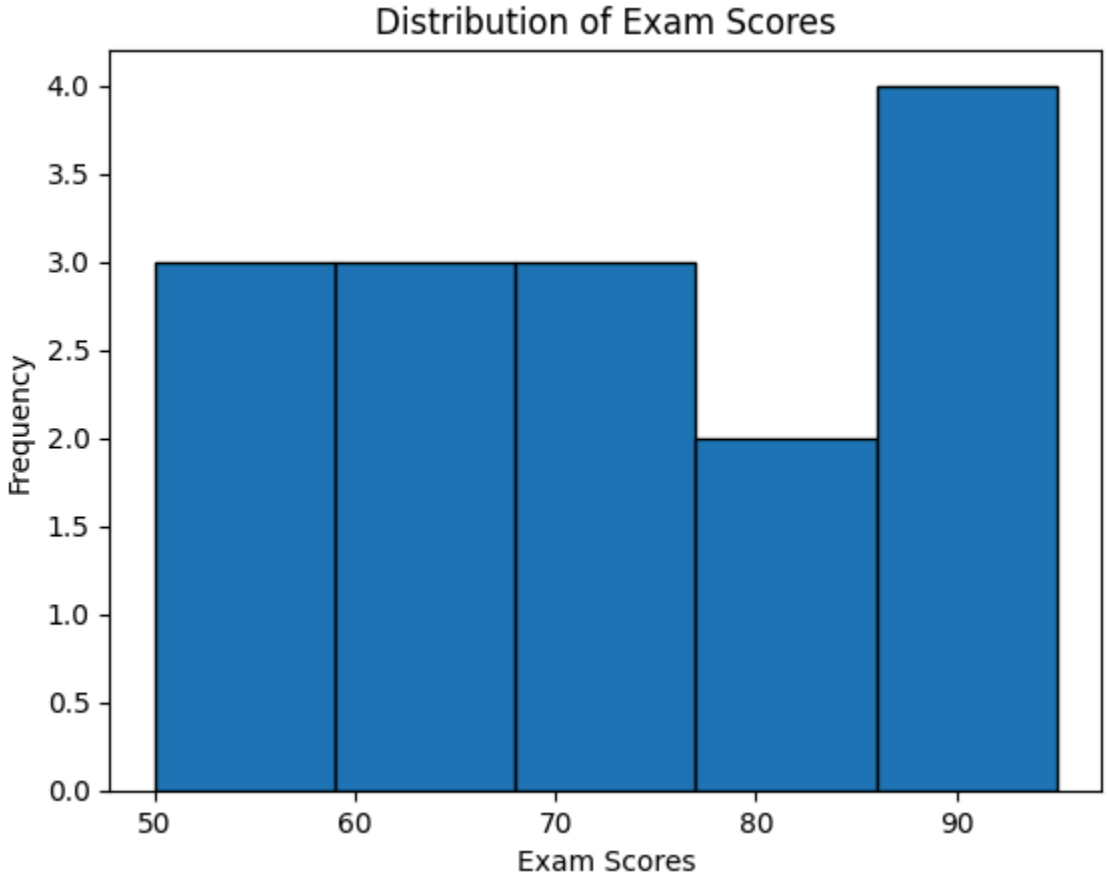


Question 31

In [10]:

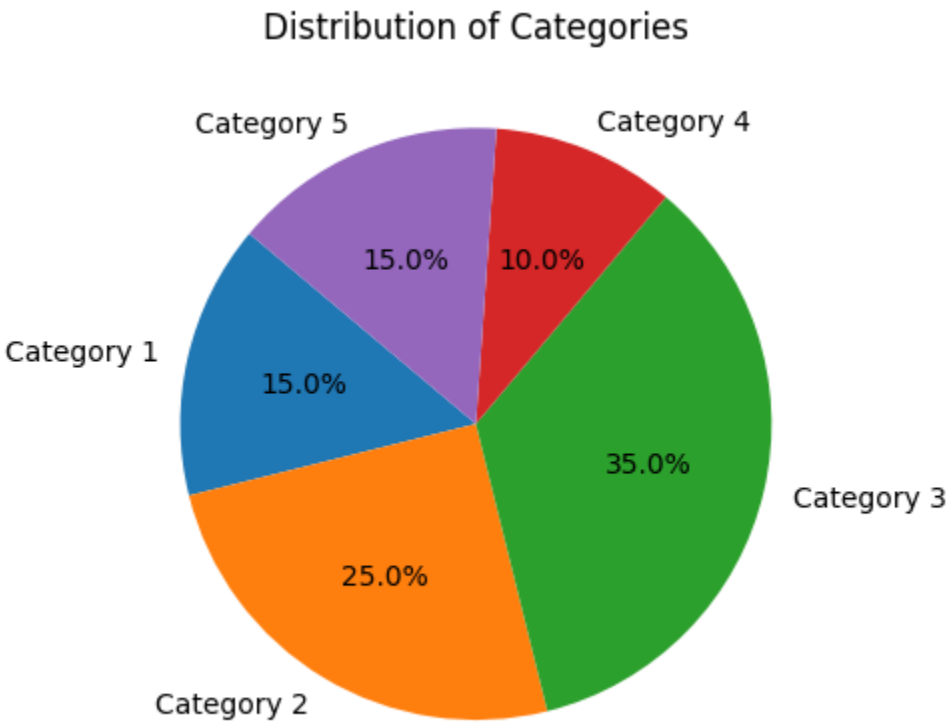
```python
# Create a Python program using Matplotlib to generate a scatter plot showing
# study_hours = [2, 3, 4.5, 1, 6, 7.5, 8, 5, 9, 2.5, 3.5, 7, 6.5, 4, 8.5]
# exam_scores = [55, 60, 65, 50, 75, 85, 90, 70, 95, 58, 63, 88, 80, 68, 92]

import matplotlib.pyplot as plt
import numpy as np

study_hours = [2, 3, 4.5, 1, 6, 7.5, 8, 5, 9, 2.5, 3.5, 7, 6.5, 4, 8.5]
exam_scores = [55, 60, 65, 50, 75, 85, 90, 70, 95, 58, 63, 88, 80, 68, 92]

# Scatter plot with trendline
plt.scatter(study_hours, exam_scores, color='blue')
plt.xlabel('Study Hours')
plt.ylabel('Exam Scores')
plt.title('Study Hours vs Exam Scores')

# Trendline
z = np.polyfit(study_hours, exam_scores, 1)
p = np.poly1d(z)
plt.plot(study_hours, p(study_hours), color='red')

plt.show()

# Histogram of exam scores
plt.hist(exam_scores, bins=5, edgecolor='black')
plt.xlabel('Exam Scores')
plt.ylabel('Frequency')
plt.title('Distribution of Exam Scores')
plt.show()
```


Study Hours vs Exam Scores
Exam Scores
Study Hours
50
60
70
80
90
1
2
3
4
5
6
7
8
9


Distribution of Exam Scores
Frequency
Exam Scores
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
50
60
70
80
90

Question 32

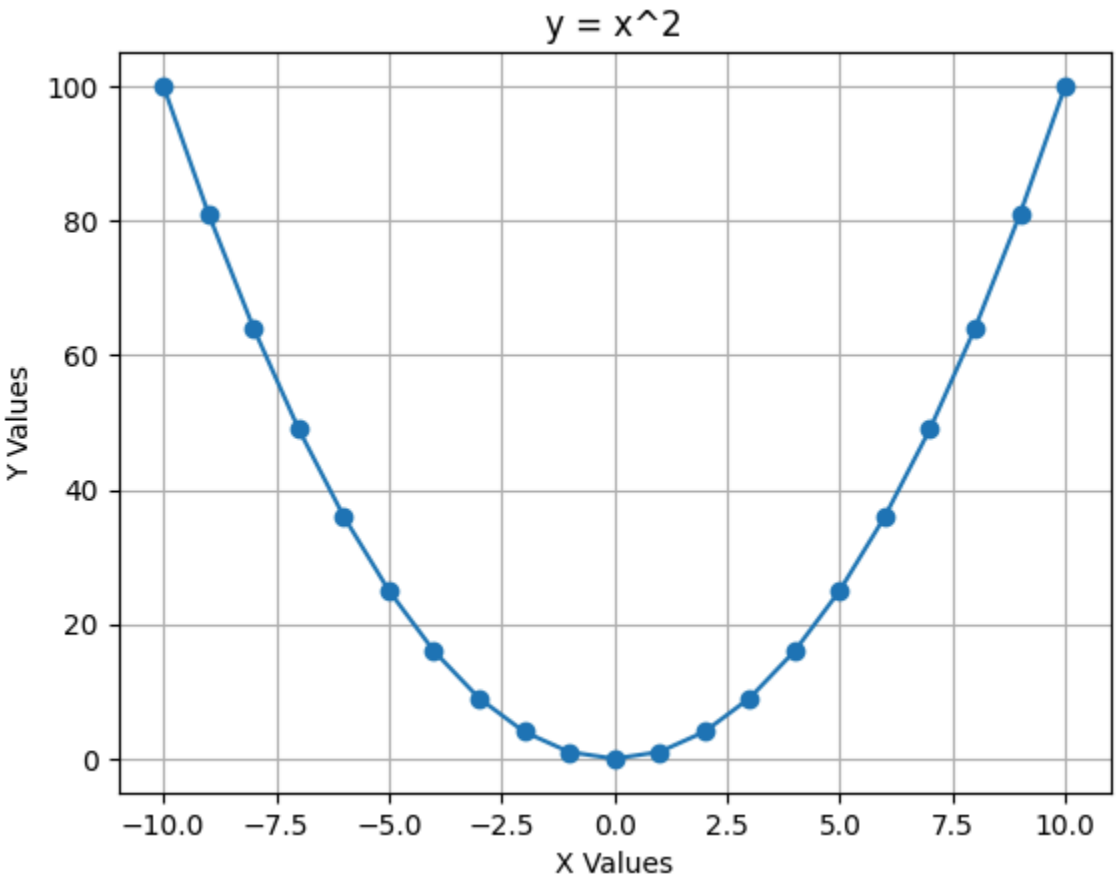
In [9]:
```
# Write a Python program using Matplotlib to create a pie chart that shows the

import matplotlib.pyplot as plt

categories = ['Category 1', 'Category 2', 'Category 3', 'Category 4', 'Categor
percentages = [15, 25, 35, 10, 15]

plt.pie(percentages, labels=categories, autopct='%1.1f%%', startangle=140)
plt.title('Distribution of Categories')
plt.show()
```



Question 33

In [8]:
```
# Write a Python program using Matplotlib to create a line plot of the functic

import matplotlib.pyplot as plt
import numpy as np

x = np.arange(-10, 11, 1)
y = x**2

plt.plot(x, y,marker='o')
plt.xlabel('X Values')
plt.ylabel('Y Values')
plt.title('y = x^2')
plt.grid(True)
```

```
plt.show()
```



Question 34

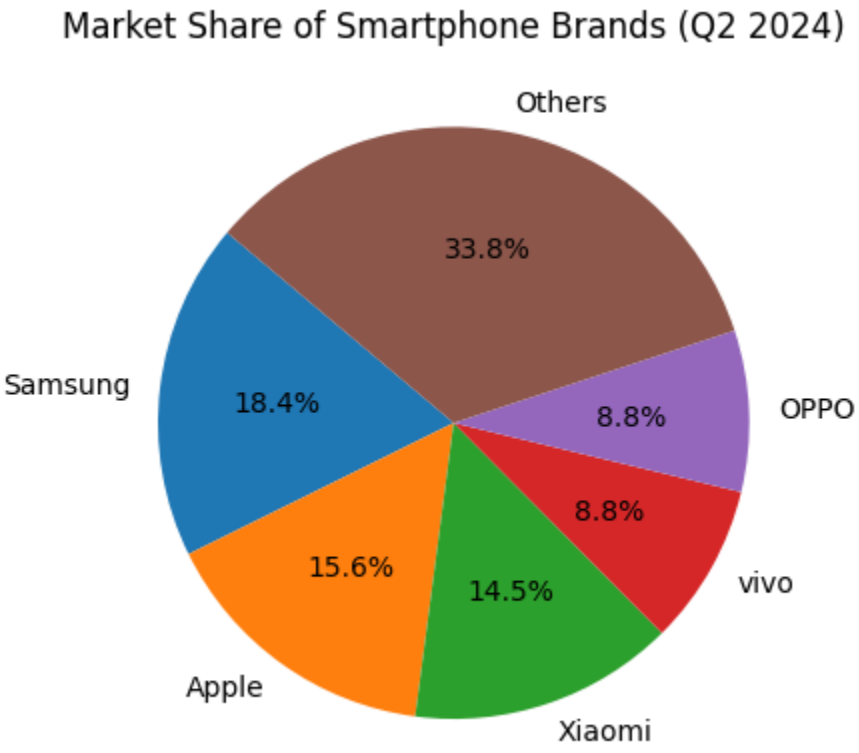
In [4]:
```
# Create a 2D NumPy array where each row represents a student and columns repr

import numpy as np

students_scores = np.random.randint(50, 101, size=(5, 3))
print("Students' Scores:\n", students_scores)
```

```
Students' Scores:
 [[55 62 73]
 [73 76 79]
 [54 75 50]
 [76 94 86]
 [80 71 97]]
```

Question 35

In [3]:
```
# The data given below is the market share of different smartphone brands as c
# Brand Percentage /Share
# Samsung: 18.4%
# Apple: 15.6%
# Xiaomi: 14.5%
# vivo: 8.8%
```

```
# OPPO: 8.8%
# Others: 33.8%
# Visualize the above data with Pie Chart using Python Programming Language.

import matplotlib.pyplot as plt

brands = ['Samsung', 'Apple', 'Xiaomi', 'vivo', 'OPPO', 'Others']
market_share = [18.4, 15.6, 14.5, 8.8, 8.8, 33.8]

plt.pie(market_share, labels=brands, autopct='%1.1f%%', startangle=140)
plt.title('Market Share of Smartphone Brands (Q2 2024)')
plt.show()
```



Question 36

In [2]:
```
# An array representing the ages of a group of people:
# ages = np.array([18, 22, 21, 19, 22, 24, 20, 25, 30, 32, 21, 20, 18, 19, 23]
# 1. Filter the array to find the ages that fall between 20 and 30, inclusive.
# 2. Calculate the mean and standard deviation of filtered_ages.
# 3. Create a new array adjusted_ages by subtracting the mean of filtered_ages
# 4. Find the indices of the elements in adjusted_ages that are negative, indi

import numpy as np

ages = np.array([18, 22, 21, 19, 22, 24, 20, 25, 30, 32, 21, 20, 18, 19, 23])

# Filter ages between 20 and 30
filtered_ages = ages[(ages >= 20) & (ages <= 30)]
```

```
print("Filtered Ages:", filtered_ages)

# Calculate mean and standard deviation
mean_age = np.mean(filtered_ages)
std_dev_age = np.std(filtered_ages)
print("Mean Age:", mean_age)
print("Standard Deviation:", std_dev_age)

# Create adjusted_ages array
adjusted_ages = ages - mean_age
print("Adjusted Ages:", adjusted_ages)

# Find indices of negative elements
negative_indices = np.where(adjusted_ages < 0)
print("Indices of Ages Below Mean:", negative_indices)
```

```
Filtered Ages: [22 21 22 24 20 25 30 21 20 23]
Mean Age: 22.8
Standard Deviation: 2.85657137141714
Adjusted Ages: [-4.8 -0.8 -1.8 -3.8 -0.8  1.2 -2.8  2.2  7.2  9.2 -1.8 -2.8
-4.8 -3.8
  0.2]
Indices of Ages Below Mean: (array([ 0,  1,  2,  3,  4,  6, 10, 11, 12, 13]),)
```

Question 37

In [1]:
```
# Assume a 3*3 array of your choice. Write a program that sorts all the rows i
# Reference example:
# Array [(3,1,2),(9,5,6),(4,8,7)]
# Rows sorted : [[1 2 3] [5 6 9] [4 7 8]]
# Added in column-wise fashion: [10 15 20]

import numpy as np

array = np.array([[3, 1, 2], [9, 5, 6], [4, 8, 7]])

# Sort rows
sorted_array = np.sort(array, axis=1)
print("Rows Sorted:\n", sorted_array)

# Add column-wise
column_sum = np.sum(sorted_array, axis=0)
print("Column-wise Sum:", column_sum)
```

```
Rows Sorted:
 [[1 2 3]
 [5 6 9]
 [4 7 8]]
Column-wise Sum: [10 15 20]
```