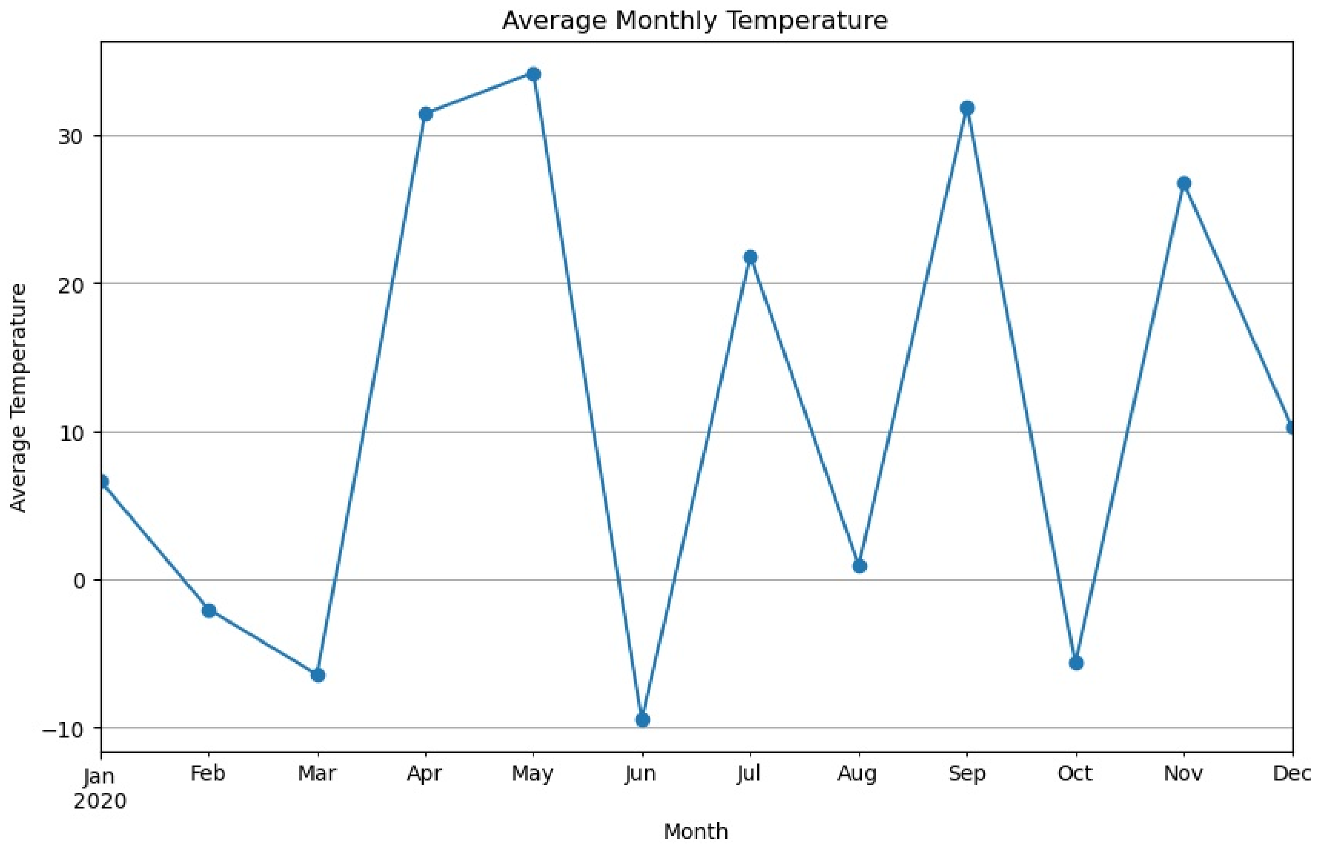
In [ ]:
```
Program 8:
Develop a program to read a CSV file containing daily temperature data with columns like 'Date' and 'Temperatur
average temperature for each month. Visualize the average monthly temperatures using a line chart.
```

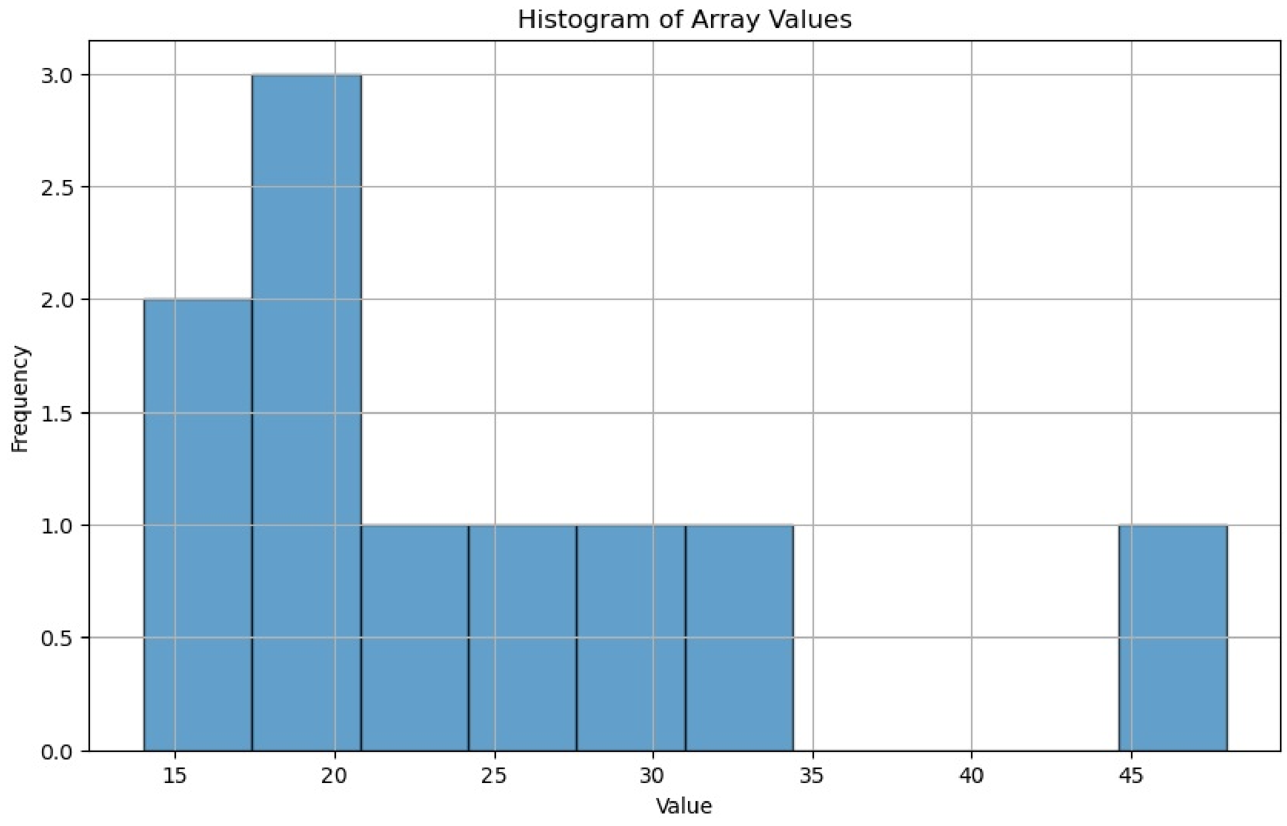
In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt

def compute_average_monthly_temperatures (file_path):
    df = pd.read_csv(file_path, parse_dates=['Date'])
    df['Date'] = pd.to_datetime(df['Date'])
    df['YearMonth'] = df ['Date'].dt.to_period('M')
    monthly_avg_temp = df.groupby ('YearMonth') ['Temperature'].mean()

    return monthly_avg_temp

def visualize_average_monthly_temperatures (monthly_avg_temp):
    plt.figure(figsize=(10, 6))
    monthly_avg_temp.plot(kind='line', marker='o')

    plt.title('Average Monthly Temperature')
    plt.xlabel('Month')
    plt.ylabel('Average Temperature')
    plt.grid(True)

    plt.show()

file_path = input("Enter the path to the CSV file containing temperature data: ")

monthly_avg_temp = compute_average_monthly_temperatures (file_path)

visualize_average_monthly_temperatures (monthly_avg_temp)
```



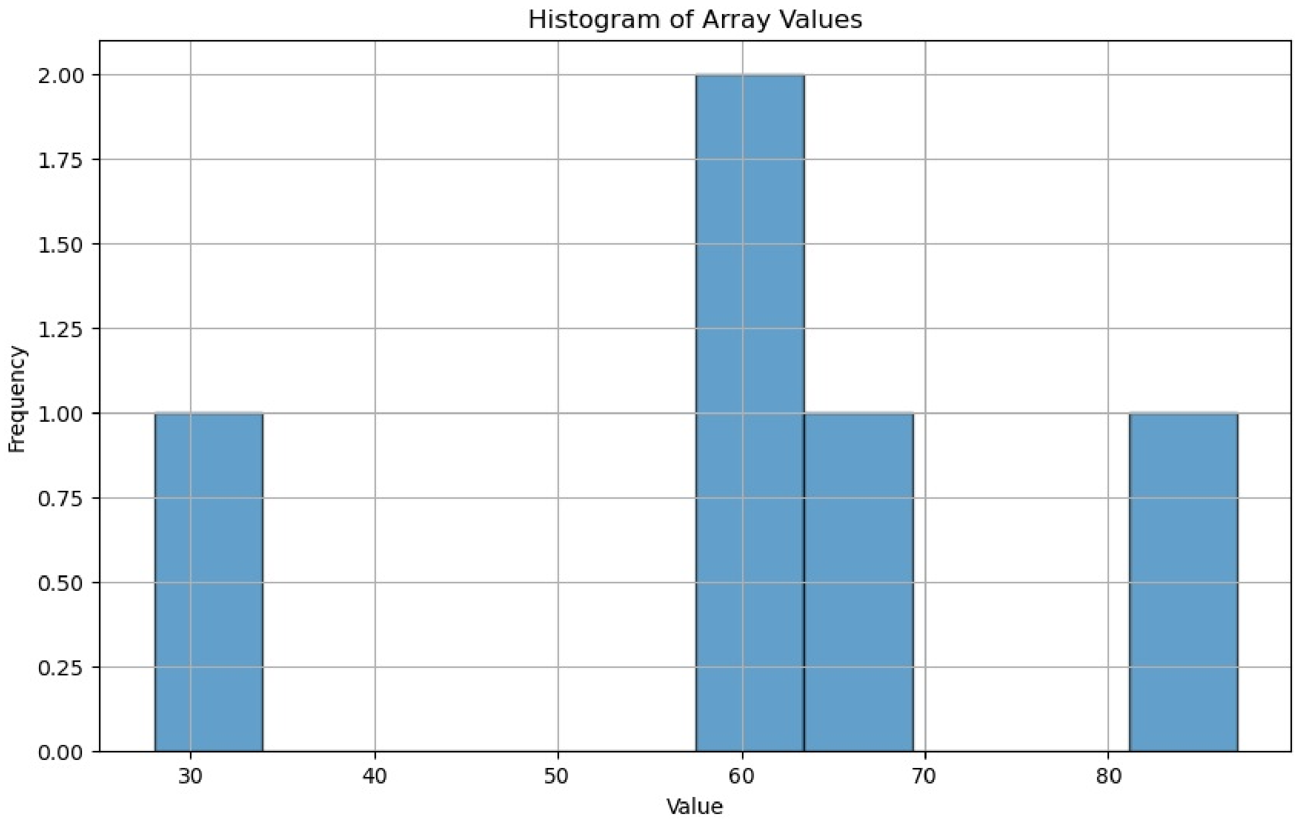
In [ ]:
```
Program 9:
Develop a program to generate a NumPy array of random integers. Perform basic array operations such as finding
deviation. Visualize the array values using a histogram.
```

In [3]:
```
import numpy as np
import matplotlib.pyplot as plt

def generate_random_array(size, low, high):
  return np.random.randint(low, high, size)

def array_operations(arr):
  mean = np.mean(arr)
  median = np.median(arr)
  variance = np.var(arr)
  std_deviation = np.std(arr)
  return mean, median, variance, std_deviation

def visualize_array_histogram(arr):
  plt.figure(figsize=(10, 6))
  plt.hist(arr, bins=10, edgecolor='black', alpha=0.7)
  plt.title('Histogram of Array Values')
  plt.xlabel('Value')
  plt.ylabel('Frequency')
  plt.grid(True)
  plt.show()

size = int(input("Enter the size of the array: "))
low = int(input("Enter the lower bound of the random range: "))
high = int(input("Enter the upper bound of the random range: "))

arr = generate_random_array(size, low, high)
print("\n Random array of integers is:\n",arr)

mean, median, variance, std_deviation = array_operations(random_array)

print("Mean:", mean)
print("Median:", median)
print("Variance:", variance)
print("Standard Deviation:", std_deviation)

visualize_array_histogram(random_array)

size = int(input("Enter the size of the array: "))
low = int(input("Enter the lower bound of the random integers: "))
high = int(input("Enter the upper bound of the random integers: "))

arr = generate_random_array(size, low, high)
print("\n Random array of integers is: \n", arr)

mean, median, variance, std_deviation = array_operations(arr)

print(f"\nArray Operations Results:")
print(f"Mean: {mean}")
print(f"Median: {median}")
print(f"Variance: {variance}")
print(f"Standard Deviation: {std_deviation}")

visualize_array_histogram(arr)
```

```
Mean: 24.5
Median: 21.0
Variance: 95.25
Standard Deviation: 9.759610647971568
```



```
Random array of integers is:
[28 87 60 58 69]


Array Operations Results:
Mean: 60.4
Median: 60.0
Variance: 367.44000000000005
Standard Deviation: 19.16872452720838
```



In [ ]:

```
Program 10:
Design a basic calculator that can perform addition, subtraction, multiplication, and division. Extend it to ha
roots,exponents, and trigonometric functions.
```

In [9]:

```
import math

def basic_operations():
    print("\n--- Basic Operations ---")
    num1 = float(input("Enter the first number: "))
    operator = input("Enter the operation (+, -, *, /): ")
    num2 = float(input("Enter the second number: "))

    if operator == '+':
        result = num1 + num2
    elif operator == '-':
        result = num1 - num2
    elif operator == '*':
        result = num1 * num2
    elif operator == '/':
        if num2 != 0:
            result = num1 / num2
        else:
            result = "Error: Division by zero is undefined."
    else:
        result = "Invalid operation"

    print(f"Result: {result}")

def advanced_operations():
    print("\n--- Advanced Operations ---")
    print("1. Square Root")
    print("2. Exponentiation")
    print("3. Sine")
    print("4. Cosine")
    print("5. Tangent")

    choice = int(input("Choose an operation (1-5): "))

    if choice == 1:
        num = float(input("Enter the number: "))
        result = math.sqrt(num)
```

```python
            print(f"Square Root of {num} is {result}")
        elif choice == 2:
            base = float(input("Enter the base: "))
            exp = float(input("Enter the exponent: "))
            result = math.pow(base, exp)
            print(f"{base} raised to the power of {exp} is {result}")
        elif choice == 3:
            angle = float(input("Enter the angle in degrees: "))
            result = math.sin(math.radians(angle))
            print(f"Sine of {angle} degrees is {result}")
        elif choice == 4:
            angle = float(input("Enter the angle in degrees: "))
            result = math.cos(math.radians(angle))
            print(f"Cosine of {angle} degrees is {result}")
        elif choice == 5:
            angle = float(input("Enter the angle in degrees: "))
            result = math.tan(math.radians(angle))
            print(f"Tangent of {angle} degrees is {result}")
        else:
            print("Invalid choice")

while True:
    print("\n--- Calculator ---")
    print("1. Basic Operations")
    print("2. Advanced Operations")
    print("3. Exit")

    choice = int(input("Choose an option (1-3): "))

    if choice == 1:
        basic_operations()
    elif choice == 2:
        advanced_operations()
    elif choice == 3:
        print("Exiting the calculator. Goodbye!")
        break
    else:
        print("Invalid choice. Please try again.")
```

```
--- Calculator ---
1. Basic Operations
2. Advanced Operations
3. Exit
--- Basic Operations ---
Result: 2.0

--- Calculator ---
1. Basic Operations
2. Advanced Operations
3. Exit
--- Basic Operations ---
Result: 0.0

--- Calculator ---
1. Basic Operations
2. Advanced Operations
3. Exit
--- Basic Operations ---
Result: 5.0

--- Calculator ---
1. Basic Operations
2. Advanced Operations
3. Exit
--- Basic Operations ---
Result: 5.0

--- Calculator ---
1. Basic Operations
2. Advanced Operations
3. Exit
--- Advanced Operations ---
1. Square Root
2. Exponentiation
3. Sine
4. Cosine
5. Tangent
Square Root of 4.0 is 2.0

--- Calculator ---
1. Basic Operations
2. Advanced Operations
3. Exit
```

```
--- Advanced Operations ---
1. Square Root
2. Exponentiation
3. Sine
4. Cosine
5. Tangent
2.0 raised to the power of 3.0 is 8.0

--- Calculator ---
1. Basic Operations
2. Advanced Operations
3. Exit
--- Advanced Operations ---
1. Square Root
2. Exponentiation
3. Sine
4. Cosine
5. Tangent
Sine of 90.0 degrees is 1.0

--- Calculator ---
1. Basic Operations
2. Advanced Operations
3. Exit
--- Advanced Operations ---
1. Square Root
2. Exponentiation
3. Sine
4. Cosine
5. Tangent
Cosine of 45.0 degrees is 0.7071067811865476

--- Calculator ---
1. Basic Operations
2. Advanced Operations
3. Exit
--- Advanced Operations ---
1. Square Root
2. Exponentiation
3. Sine
4. Cosine
5. Tangent
Tangent of 90.0 degrees is 1.633123935319537e+16

--- Calculator ---
1. Basic Operations
2. Advanced Operations
3. Exit
Invalid choice. Please try again.

--- Calculator ---
1. Basic Operations
2. Advanced Operations
3. Exit
Exiting the calculator. Goodbye!
```

In [ ]:

```
Program 11:
Develop a to-do list application where users can add, delete, and view their tasks. Extend it by adding feature
mark tasks as completed.
```

In [15]:

```python
from datetime import datetime

class Task:
    def __init__(self, description, due_date=None, priority=None):
        self.description = description
        self.due_date = due_date
        self.priority = priority
        self.completed = False

    def __str__(self):
        status = "Completed" if self.completed else "Incomplete"
        return f"Task: {self.description}, Due: {self.due_date}, Priority: {self.priority}, Status: {status}"

class ToDoList:
    def __init__(self):
        self.tasks = []

    def add_task(self, description, due_date=None, priority=None):
        task = Task(description, due_date, priority)
        self.tasks.append(task)
        print(f"Task added: {description}")

    def delete_task(self, task_id):
        if 0 <= task_id < len(self.tasks):
            removed_task = self.tasks.pop(task_id)
            print(f"Task deleted: {removed_task.description}")
        else:
            print("Invalid task ID")

    def view_tasks(self):
        if not self.tasks:
            print("No tasks in the list.")
        else:
            for idx, task in enumerate(self.tasks):
                print(f"{idx}. {task}")

    def mark_task_completed(self, task_id):
        if 0 <= task_id < len(self.tasks):
            self.tasks[task_id].completed = True
            print(f"Task marked as completed: {self.tasks[task_id].description}")
        else:
            print("Invalid task ID")

todo_list = ToDoList()

while True:
    print("\n--- To-Do List Menu ---")
    print("1. Add Task")
    print("2. Delete Task")
    print("3. View Tasks")
    print("4. Mark Task as Completed")
    print("5. Exit")

    choice = input("Choose an option (1-5): ")

    if choice == "1":
        description = input("Enter the task description: ")
        due_date_input = input("Enter the due date (YYYY-MM-DD) or leave blank: ")
        due_date = due_date_input if due_date_input else None
        priority = input("Enter the priority (High, Medium, Low) or leave blank: ")
        todo_list.add_task(description, due_date, priority)
    elif choice == "2":
        task_id = int(input("Enter the task ID to delete: "))
        todo_list.delete_task(task_id)
    elif choice == "3":
        todo_list.view_tasks()
    elif choice == "4":
        task_id = int(input("Enter the task ID to mark as completed: "))
        todo_list.mark_task_completed(task_id)
    elif choice == "5":
        print("Exiting the to-do list application. Goodbye!")
        break
    else:
        print("Invalid choice. Please try again.")
```

```
--- To-Do List Menu ---
1. Add Task
2. Delete Task
3. View Tasks
4. Mark Task as Completed
5. Exit
Task added: shopping

--- To-Do List Menu ---
1. Add Task
2. Delete Task
3. View Tasks
4. Mark Task as Completed
5. Exit
Task added: cleaning

--- To-Do List Menu ---
1. Add Task
2. Delete Task
3. View Tasks
4. Mark Task as Completed
5. Exit
0. Task: shopping, Due: 2024-12-02, Priority: medium, Status: Incomplete
1. Task: cleaning, Due: 2023-12-05, Priority: high, Status: Incomplete

--- To-Do List Menu ---
1. Add Task
2. Delete Task
3. View Tasks
4. Mark Task as Completed
5. Exit
```

```
Task marked as completed: cleaning

--- To-Do List Menu ---
1. Add Task
2. Delete Task
3. View Tasks
4. Mark Task as Completed
5. Exit
Task deleted: cleaning

--- To-Do List Menu ---
1. Add Task
2. Delete Task
3. View Tasks
4. Mark Task as Completed
5. Exit
0. Task: shopping, Due: 2024-12-02, Priority: medium, Status: Incomplete

--- To-Do List Menu ---
1. Add Task
2. Delete Task
3. View Tasks
4. Mark Task as Completed
5. Exit
Exiting the to-do list application. Goodbye!
```

In [ ]:
```
Program 12:
Develop a hangman game where users try to guess a hidden word by suggesting letters within a certain number of
```

In [19]:
```python
import random

WORDS = ["python", "java", "kotlin", "javascript", "hangman", "programming", "development"]

def choose_word():
    return random.choice(WORDS)

def display_word(word, guessed_letters):
    return ''.join([letter if letter in guessed_letters else '_' for letter in word])

def play_hangman():
    word = choose_word()
    guessed_letters = set()
    attempts = 6

    print("Welcome to Hangman!")

    while attempts > 0:
        print(f"\nWord: {display_word(word, guessed_letters)}")
        print(f"Attempts left: {attempts}")
        print(f"Guessed letters: {', '.join(sorted(guessed_letters))}")

        guess = input("Guess a letter: ").lower()

        if len(guess) != 1 or not guess.isalpha():
            print("Please enter a single letter.")
            continue

        if guess in guessed_letters:
            print("You have already guessed that letter. Try again.")
            continue

        guessed_letters.add(guess)

        if guess in word:
            print("Good guess!")
            if all(letter in guessed_letters for letter in word):
                print(f"Congratulations! You guessed the word: {word}")
                break
        else:
            print("Wrong guess.")
            attempts -= 1

        if attempts == 0:
            print(f"Game over! The word was: {word}")

while True:
    play_hangman()
    play_again = input("Do you want to play again? (yes/no): ").lower()
    if play_again != 'yes':
        print("Thanks for playing! Goodbye!")
        break
```

```
Welcome to Hangman!

Word: _______
Attempts left: 6
Guessed letters: 
Good guess!

Word: _a___a_
Attempts left: 6
Guessed letters: a
Wrong guess.

Word: _a___a_
Attempts left: 5
Guessed letters: a, e
Good guess!

Word: ha___a_
Attempts left: 5
Guessed letters: a, e, h
Please enter a single letter.

Word: ha___a_
Attempts left: 5
Guessed letters: a, e, h
Good guess!

Word: han__an
Attempts left: 5
Guessed letters: a, e, h, n
Good guess!

Word: hang_an
Attempts left: 5
Guessed letters: a, e, g, h, n
Wrong guess.

Word: hang_an
Attempts left: 4
Guessed letters: a, e, g, h, n, t
Please enter a single letter.

Word: hang_an
Attempts left: 4
Guessed letters: a, e, g, h, n, t
Good guess!
Congratulations! You guessed the word: hangman
Thanks for playing! Goodbye!
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js