

APP WEEK 11

PYTHON PROGRAM USING CONTROL STRUCTURES AND ARRAYS

1. Q1) Implement a python program to find the first largest and second largest numbers in an Array. Note: should not use any built-in sorting functions or libraries.

```
def find_largest_and_second_largest(arr):
    if len(arr) < 2:
        print("Array should have at least two elements")
    return
    first_largest = second_largest = float('-inf')
    for num in arr:
        if num > first_largest:
            second_largest = first_largest
            first_largest = num
        elif num > second_largest and num != first_largest:
            second_largest = num
    if second_largest == float('-inf'):
        print("There is no second largest element in the array.")
    else:
        print("The first largest element is:", first_largest)
    print("The second largest element is:", second_largest)
# Input an array from the user
arr = []
n = int(input("Enter the number of elements in the array: "))
for i in range(n):
    num = int(input(f"Enter element {i+1}: "))
    arr.append(num)
    find_largest_and_second_largest(arr)
```

```
Q1.PY > find_largest_and_second_largest
1 def find_largest_and_second_largest(arr):
2     if len(arr) < 2:
3         print("Array should have at least two elements")
4         return
5
6     first_largest = second_largest = float('-inf')
7
8     for num in arr:
9         if num > first_largest:
10             second_largest = first_largest
11             first_largest = num
12         elif num > second_largest and num != first_largest:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - [] ... ^ X

● PS C:\Users\Asus\Desktop\week 11> & C:/Python311/python.exe "c:/Users/Asus/Desktop/week 11/
Q1.PY"
Enter the number of elements in the array: 4
Enter element 1: 2
Enter element 2: 3
Enter element 3: 4
Enter element 4: 5
The first largest element is: 5
```

2. **Q2)** Write a Python program to calculate the sum of even numbers and the sum of odd numbers in an array.

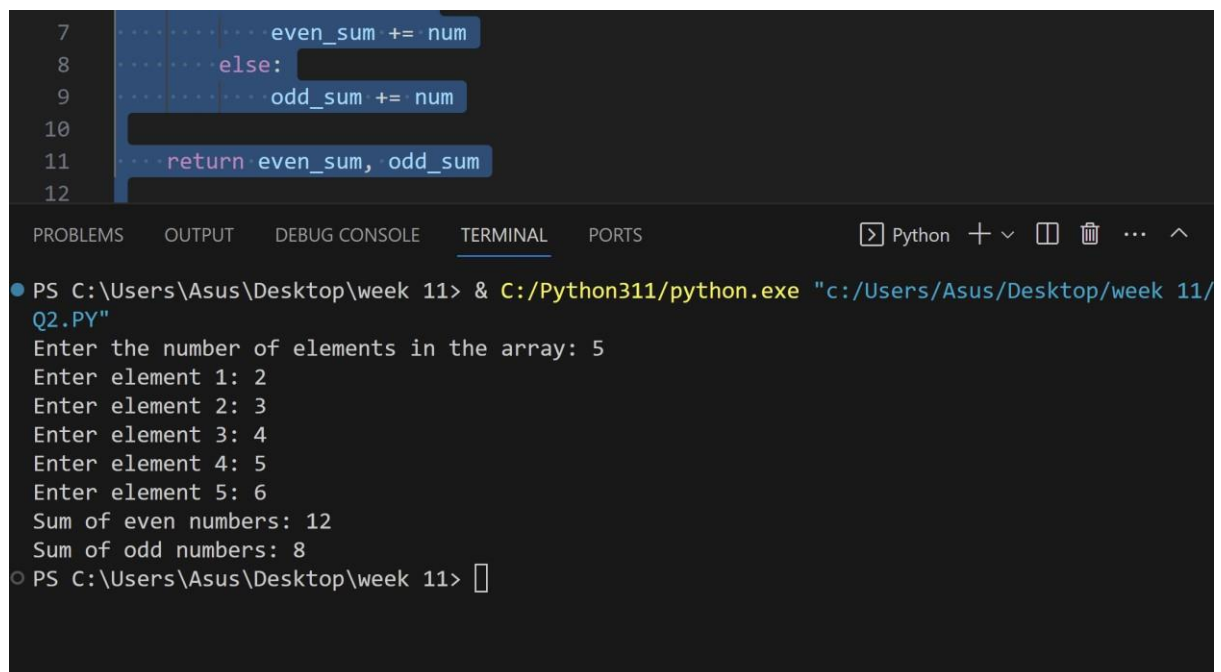
```

def sum_even_and_odd(arr):
    even_sum = 0
    odd_sum = 0
    for num in arr:
        if num % 2 == 0:
            even_sum += num
        else:
            odd_sum += num
    return even_sum,
    odd_sum

# Input an array from the user arr = [] n = int(input("Enter
the number of elements in the array: ")) for i in range(n):
num = int(input(f"Enter element {i+1}: "))
arr.append(num)

even_sum, odd_sum = sum_even_and_odd(arr)
print("Sum of even numbers:",
even_sum) print("Sum of odd numbers:",
odd_sum)

```



The screenshot shows a code editor with a Python program and a terminal window below it. The code in the editor is as follows:

```

7 ..... even_sum += num
8 ..... else:
9 ..... odd_sum += num
10 .....
11 ..... return even_sum, odd_sum
12 .....

```

The terminal window shows the execution of the program. The command prompt is at `PS C:\Users\Asus\Desktop\week 11>`. The user runs the command `C:/Python311/python.exe "c:/Users/Asus/Desktop/week 11/Q2.PY"`. The program prompts the user to enter the number of elements in the array (5), and then prompts for each element (2, 3, 4, 5, 6). The program then outputs the sum of even numbers (12) and the sum of odd numbers (8).

```

PS C:\Users\Asus\Desktop\week 11> & C:/Python311/python.exe "c:/Users/Asus/Desktop/week 11/Q2.PY"
Enter the number of elements in the array: 5
Enter element 1: 2
Enter element 2: 3
Enter element 3: 4
Enter element 4: 5
Enter element 5: 6
Sum of even numbers: 12
Sum of odd numbers: 8
PS C:\Users\Asus\Desktop\week 11>

```

3. Q3) Write a python program to count the Occurrences of a Specific Element in an Array.

```

def count_occurrences(arr, element):
    count = 0
    for num in arr:
        if num == element:
            count += 1
    return count

# Input an array from the user
arr = []
n = int(input("Enter the number of elements in the array: "))
for i in range(n):
    num = int(input(f"Enter element {i+1}: "))
    arr.append(num)

element_to_count = int(input("Enter the element to count its occurrences: "))

occurrences = count_occurrences(arr, element_to_count)

print(f"The element {element_to_count} appears {occurrences} times in the array.")

```

The screenshot shows a code editor with the following code:

```

6         count += 1
7
8     return count
9
10 # Input an array from the user
11 arr = []
12 n = int(input("Enter the number of elements in the array: "))

```

Below the code editor is a terminal window showing the execution of the program:

```

PS C:\Users\Asus\Desktop\week 11> & C:/Python311/python.exe "c:/Users/Asus/Desktop/week 11/Q3.PY"
Enter the number of elements in the array: 5
Enter element 1: 1
Enter element 2: 2
Enter element 3: 34
Enter element 4: 44
Enter element 5: 1
Enter the element to count its occurrences: 1
The element 1 appears 2 times in the array.
PS C:\Users\Asus\Desktop\week 11>

```

4. **Q4)** Write a Python program that takes a sentence as input and identifies and prints all the palindromic words in the sentence. Use an array to store the palindromic words.

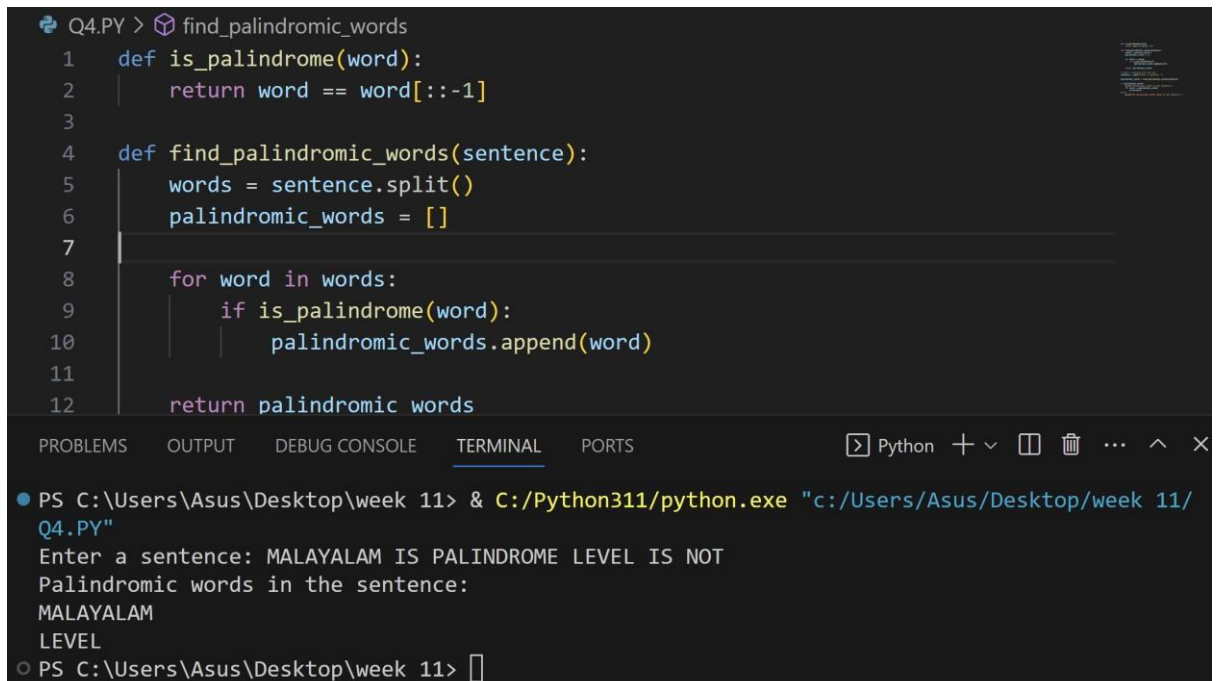
```

def is_palindrome(word):
    return word == word[::-1]
def
find_palindromic_words(sentence):
    words = sentence.split()
    palindromic_words = []
    for word in words:
        if is_palindrome(word):
            palindromic_words.append(word)
    return
palindromic_words

# Input a sentence from the user
sentence = input("Enter a sentence: ")

palindromic_words = find_palindromic_words(sentence)
if
palindromic_words:
    print("Palindromic words in the sentence:")
    for word in palindromic_words:
        print(word)
else:
    print("No palindromic words found in the sentence.")

```



The screenshot shows a code editor with a Python script named Q4.PY. The script defines two functions: `is_palindrome` and `find_palindromic_words`. The `is_palindrome` function checks if a word is a palindrome by comparing it to its reverse. The `find_palindromic_words` function takes a sentence, splits it into words, and returns a list of palindromic words. The terminal window shows the execution of the script, where the user enters the sentence "MALAYALAM IS PALINDROME LEVEL IS NOT". The program outputs the palindromic words "MALAYALAM" and "LEVEL".

```

Q4.PY > find_palindromic_words
1  def is_palindrome(word):
2      return word == word[::-1]
3
4  def find_palindromic_words(sentence):
5      words = sentence.split()
6      palindromic_words = []
7
8      for word in words:
9          if is_palindrome(word):
10             palindromic_words.append(word)
11
12     return palindromic words

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - [] [] ... ^ X

```

● PS C:\Users\Asus\Desktop\week 11> & C:/Python311/python.exe "c:/Users/Asus/Desktop/week 11/
Q4.PY"
Enter a sentence: MALAYALAM IS PALINDROME LEVEL IS NOT
Palindromic words in the sentence:
MALAYALAM
LEVEL
○ PS C:\Users\Asus\Desktop\week 11>

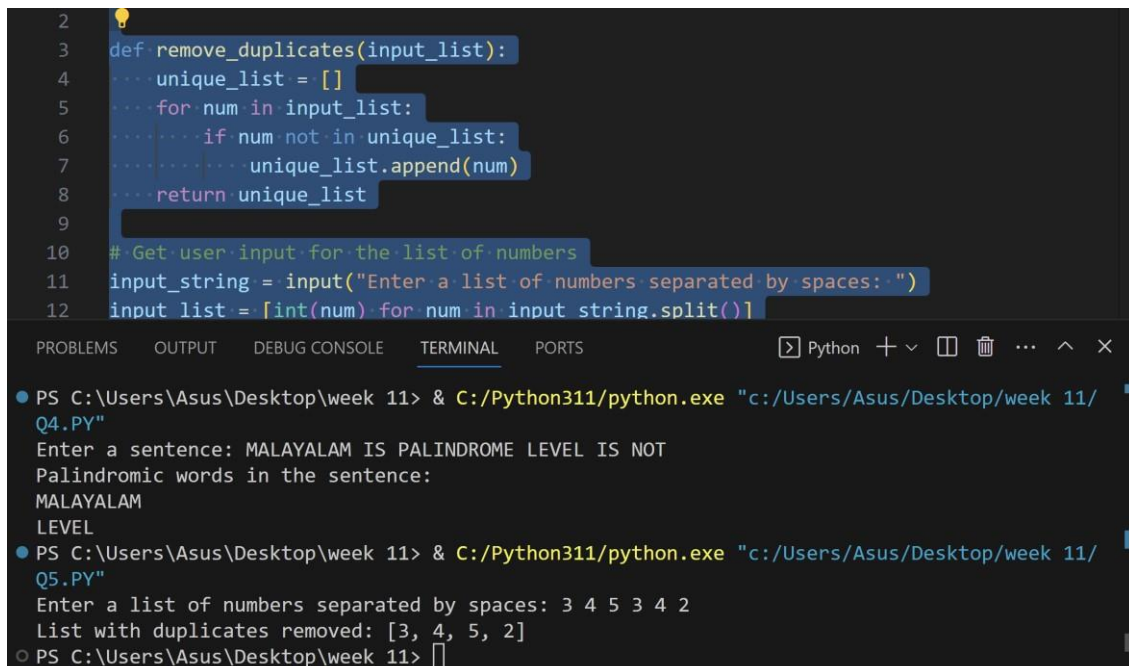
```

5. Q5) Write a Python program that takes a list of numbers and removes all duplicates from the list, preserving the original order of elements.

```
def remove_duplicates(input_list):
    unique_list = []
    for num in input_list:
        if num not in unique_list:
            unique_list.append(num)
    return unique_list

# Get user input for the list of numbers
input_string = input("Enter a list of numbers separated by spaces: ")
input_list = [int(num) for num in input_string.split()]

result_list = remove_duplicates(input_list)
print("List with duplicates removed:", result_list)
```



The screenshot shows a code editor with the Python program for Q5. The code is as follows:

```
2
3 def remove_duplicates(input_list):
4     unique_list = []
5     for num in input_list:
6         if num not in unique_list:
7             unique_list.append(num)
8     return unique_list
9
10 # Get user input for the list of numbers
11 input_string = input("Enter a list of numbers separated by spaces: ")
12 input_list = [int(num) for num in input_string.split()]

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - [ ] [X] ... ^ X
```

The terminal output shows the execution of the program:

```
PS C:\Users\Asus\Desktop\week 11> & C:/Python311/python.exe "c:/Users/Asus/Desktop/week 11/Q4.PY"
Enter a sentence: MALAYALAM IS PALINDROME LEVEL IS NOT
Palindromic words in the sentence:
MALAYALAM
LEVEL
PS C:\Users\Asus\Desktop\week 11> & C:/Python311/python.exe "c:/Users/Asus/Desktop/week 11/Q5.PY"
Enter a list of numbers separated by spaces: 3 4 5 3 4 2
List with duplicates removed: [3, 4, 5, 2]
PS C:\Users\Asus\Desktop\week 11>
```

6. **Q6)** Write a Python program that performs matrix multiplication. Ask the user to input two matrices as lists of lists (2D arrays) and then multiply them if possible. Make sure to check if the matrices are compatible for multiplication and handle errors gracefully.

```
def matrix_multiply(mat1, mat2):
    # Check if matrices are compatible for multiplication
    if len(mat1[0]) != len(mat2):
        return None # Matrices cannot be multiplied

    # Initialize the result matrix with zeros
    result = [[0 for _ in range(len(mat2[0]))] for _ in range(len(mat1))]
    # Perform matrix multiplication
    for i in range(len(mat1)):
        for j in range(len(mat2[0])):
            for k in range(len(mat2)):
                result[i][j] += mat1[i][k] * mat2[k][j]
    return result

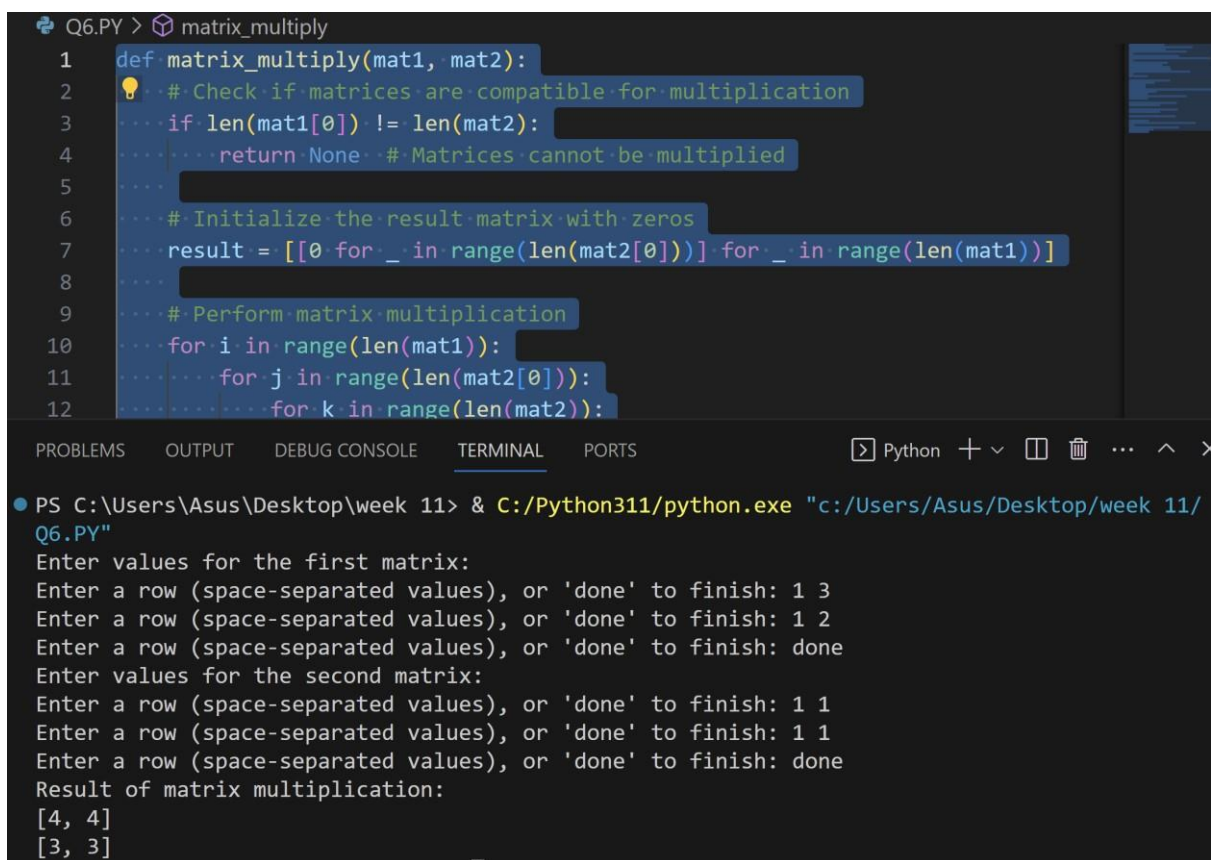
# Get user input for the first matrix
def get_matrix_input():
    matrix = []
    while True:
        row = input("Enter a row (space-separated values), or 'done' to finish: ")
        if row == 'done':
            break
        matrix.append([int(x) for x in row.split()])
    return matrix

print("Enter values for the first matrix:")
```

```

matrix1 = get_matrix_input()
print("Enter values for the second
matrix:") matrix2 = get_matrix_input()
result = matrix_multiply(matrix1,
matrix2)
if result is
None:
    print("Matrices are not compatible for multiplication.")
else:
    print("Result of matrix multiplication:")
for row in result:
    print(row)

```



The screenshot shows a code editor with a Python file named Q6.PY. The code defines a function `matrix_multiply(mat1, mat2)` that checks for matrix compatibility, initializes a result matrix with zeros, and performs the multiplication using nested loops. Below the code editor, the terminal window shows the execution of the program. It prompts the user to enter values for two matrices. The first matrix is entered as `1 3` and `1 2`, and the second matrix is entered as `1 1` and `1 1`. The final output shows the result of the matrix multiplication as `[4, 4]` and `[3, 3]`.

```

Q6.PY > matrix_multiply
1 def matrix_multiply(mat1, mat2):
2     # Check if matrices are compatible for multiplication
3     if len(mat1[0]) != len(mat2):
4         return None # Matrices cannot be multiplied
5
6     # Initialize the result matrix with zeros
7     result = [[0 for _ in range(len(mat2[0]))] for _ in range(len(mat1))]
8
9     # Perform matrix multiplication
10    for i in range(len(mat1)):
11        for j in range(len(mat2[0])):
12            for k in range(len(mat2)):

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - [] [X] ... ^ >

```

PS C:\Users\Asus\Desktop\week 11> & C:/Python311/python.exe "c:/Users/Asus/Desktop/week 11/Q6.PY"
Enter values for the first matrix:
Enter a row (space-separated values), or 'done' to finish: 1 3
Enter a row (space-separated values), or 'done' to finish: 1 2
Enter a row (space-separated values), or 'done' to finish: done
Enter values for the second matrix:
Enter a row (space-separated values), or 'done' to finish: 1 1
Enter a row (space-separated values), or 'done' to finish: 1 1
Enter a row (space-separated values), or 'done' to finish: done
Result of matrix multiplication:
[4, 4]
[3, 3]

```


7. Q7) Write a python program to print diamond number pattern using Nested Loops.

```
1
123
12345
123
1
```

```
def print_diamond_pattern(n):
    # Print the top half of the diamond
    for i in range(1, n + 1, 2):
        # Print spaces
        spaces = " " * ((n - i) // 2)
        # Print numbers
        numbers = "".join(str(num)
                           for num in range(1, i + 1))
        print(spaces + numbers)

    # Print the bottom half of the diamond
    for i in range(n - 2, 0, -2):
        # Print spaces
        spaces = " " * ((n - i) // 2)
        # Print numbers
        numbers = "".join(str(num)
                           for num in range(1, i + 1))
        print(spaces + numbers)

    n = int(input("Enter the number of rows (odd number):
    "))
    if n % 2 == 0:
        print("Please enter an odd number.")
    else:
        print_diamond_pattern(n)
```

```
Q7.PY > print_diamond_pattern

10     # Print the bottom half of the diamond
11     for i in range(n - 2, 0, -2):
12         # Print spaces
13         spaces = " " * ((n - i) // 2)
14         # Print numbers
15         numbers = "".join(str(num) for num in range(1, i + 1))
16         print(spaces + numbers)
17
18     n = int(input("Enter the number of rows (odd number): "))
19     if n % 2 == 0:
20         print("Please enter an odd number.")
21     else:
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - [] [X] ... ^

```
PS C:\Users\Asus\Desktop\week 11> & C:/Python311/python.exe "c:/Users/Asus/Desktop/week 11/Q7.PY"
Enter the number of rows (odd number): 5
1
123
12345
123
1
```

8. **Q8)** Write a Python program that simulates a simple guessing game. Generate a random number and have the user guess it. Provide hints like "too high" or "too low" until they guess correctly.

```
import random

# Generate a random number between 1 and 100
secret_number = random.randint(1, 100)

# Initialize the number of guesses
guesses = 0
while
True:
    # Prompt the user for a guess    guess =
    int(input("Guess the number between 1 and 100: "))
    guesses += 1
```

```

    # Check if the guess is correct
if guess == secret_number:
    print(f"Congratulations! You guessed the number {secret_number} in {guesses} attempts.")
    break
elif guess < secret_number:
    print("Too low! Try again.")
else:
    print("Too high! Try again.")

```

The screenshot shows a code editor with a Python script named Q8.PY. The script generates a random number between 1 and 100 and allows the user to guess it. The terminal output shows the program running with several guesses and feedback messages.

```

Q8.PY > ...
1 import random
2
3 # Generate a random number between 1 and 100
4 secret_number = random.randint(1, 100)
5
6 # Initialize the number of guesses
7 guesses = 0
8
9 while True:
10     # Prompt the user for a guess
11     guess = int(input("Guess the number between 1 and 100: "))
12     guesses += 1

```

Terminal Output:

```

PS C:\Users\Asus\Desktop\week 11> & C:/Python311/python.exe "c:/Users/Asus/Desktop/week 11/Q8.PY"
Guess the number between 1 and 100: 97
Too high! Try again.
Guess the number between 1 and 100: 22
Too low! Try again.
Guess the number between 1 and 100: 50
Too low! Try again.
Guess the number between 1 and 100: 32
Too low! Try again.
Guess the number between 1 and 100: 

```

Q9) Write a Python program that checks the strength of a password entered by a user. The program should assess the password based on criteria like length, use of uppercase and lowercase letters, digits, and special characters. Use control structures and arrays to provide a detailed evaluation

```
import string

# Function to check if a password meets specific criteria def
check_password_strength(password):
    criteria_met = [False, False, False, False, False]

    # Check the length of the password
    if len(password) >= 8:
        criteria_met[0] = True

    # Check for lowercase letters if
    any(char.islower() for char in password):
        criteria_met[1] = True
```

```

    # Check for uppercase letters    if
any(char.isupper() for char in password):
    criteria_met[2] = True

    # Check for digits    if
any(char.isdigit() for char in password):
    criteria_met[3] = True

    # Check for special characters    if any(char in
string.punctuation for char in password):
    criteria_met[4] = True
    return
criteria_met

# Get user input for the password
password = input("Enter your password: ")

# Check the password strength criteria_met =
check_password_strength(password)

# Provide a detailed evaluation
print("Password Strength Evaluation:")
print("1. Minimum length of 8 characters:", "Met" if criteria_met[0] else "Not
Met")
print("2. At least one lowercase letter:", "Met" if criteria_met[1] else "Not
Met")
print("3. At least one uppercase letter:", "Met" if criteria_met[2] else "Not
Met") print("4. At least one digit:", "Met" if criteria_met[3] else "Not
Met") print("5. At least one special character:", "Met" if criteria_met[4]
else "Not Met")

# Check if all criteria are met if
all(criteria_met):
    print("Congratulations! Your password is strong.")
else:
    print("Your password does not meet all the criteria for a strong password.")

```

```
Q9.PY > ...
1  import string
2
3  def check_password_strength(password):
4      criteria_met = [False, False, False, False, False]
5
6      # Check the length of the password
7      if len(password) >= 8:
8          criteria_met[0] = True
9
10     # Check for lowercase letters
11     if any(char.islower() for char in password):
12         criteria_met[1] = True
13
14     # Check for uppercase letters
15     if any(char.isupper() for char in password):
16         criteria_met[2] = True
17
18     # Check for digits
19     if any(char.isdigit() for char in password):
20         criteria_met[3] = True
21
22     # Check for special characters
23     if any(char in string.punctuation for char in password):
24         criteria_met[4] = True
25
26     return criteria_met
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - [] [X] ... ^ X

```
PS C:\Users\Asus\Desktop\week 11> & C:/Python311/python.exe "c:/Users/Asus/Desktop/week 11/Q9.PY"
Enter your password: SymSys@2026
Password Strength Evaluation:
1. Minimum length of 8 characters: Met
2. At least one lowercase letter: Met
3. At least one uppercase letter: Met
4. At least one digit: Met
5. At least one special character: Met
Congratulations! Your password is strong.
```

9. **Q10)** Write a Python program that generates the Fibonacci sequence up to a specified number of terms using a loop and stores it in an array.

```
def generate_fibonacci_sequence(n):
    fibonacci_sequence = []

    # Handle the case for n=1 separately
    if n >= 1:
        fibonacci_sequence.append(0)

    # Handle the case for n=2 separately
    if n >= 2:
        fibonacci_sequence.append(1)

    # Generate the rest of the sequence using a loop
    while len(fibonacci_sequence) < n:
        next_number = fibonacci_sequence[-1] + fibonacci_sequence[-2]
        fibonacci_sequence.append(next_number)

    return fibonacci_sequence

# Get user input for the number of terms n = int(input("Enter the
number of Fibonacci terms to generate: "))
if n <=
0:
    print("Please enter a positive integer.")
else:
    result = generate_fibonacci_sequence(n)
print("Fibonacci sequence up to", n, "terms:", result)
```

```
q10.py > generate_fibonacci_sequence
1 def generate_fibonacci_sequence(n):
2     fibonacci_sequence = []
3
4     # Handle the case for n=1 separately
5     if n >= 1:
6         fibonacci_sequence.append(0)
7
8     # Handle the case for n=2 separately
9     if n >= 2:
10        fibonacci_sequence.append(1)
11
12    # Generate the rest of the sequence using a loop

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - [] [X] ... ^ >

PS C:\Users\Asus\Desktop\week 11> & C:/Python311/python.exe "c:/Users/Asus/Desktop/week 11/q10.py"
Enter the number of Fibonacci terms to generate: 5
Fibonacci sequence up to 5 terms: [0, 1, 1, 2, 3]
PS C:\Users\Asus\Desktop\week 11> 
```