Additional Assignments

Assignment 1: Triangle Pattern

Problem Statement:

Write a Python program to print the following triangle pattern:

1

12

123

1234

12345

1234

123

12

1

Solution:

```
# Triangle Pattern
for i in range(1, 6):
    row = ""
    for j in range(1, i + 1):
        row += str(j)
    print(row)

for i in range(4, 0, -1):
    row = ""
    for j in range(1, i + 1):
        row += str(j)
```

```
print(row)
```

Assignment 2: Matrix Multiplication

Problem Statement: Write a Python program to perform matrix multiplication for two given matrices.

Sample Input:

Enter the number of rows for matrix A: 2

Enter the number of columns for matrix A: 3

Enter the elements of matrix A:

1 2 3

4 5 6

Enter the number of rows for matrix B: 3

Enter the number of columns for matrix B: 2

Enter the elements of matrix B:

7 8

9 10

11 12

Sample Output:

Matrix A:

1 2 3

4 5 6

Matrix B:

7 8

9 10

11 12

Resultant Matrix:

58 64

139 154

Solution:

```
# Matrix Multiplication
rows_A = int(input("Enter the number of rows for matrix A: "))
cols_A = int(input("Enter the number of columns for matrix A: "))

matrix_A = [list(map(int, input().split())) for _ in range(rows_A)]

rows_B = int(input("Enter the number of rows for matrix B: "))
cols_B = int(input("Enter the number of columns for matrix B: "))

matrix_B = [list(map(int, input().split())) for _ in range(rows_B)]

result_matrix = [[0] * cols_B for _ in range(rows_A)]

for i in range(rows_A):
    for j in range(cols_B):
        for k in range(cols_A):
            result_matrix[i][j] += matrix_A[i][k] * matrix_B[k][j]

# Display matrices and result
```

```
print("\nMatrix A:")
for row in matrix_A:
    print(' '.join(map(str, row)))

print("\nMatrix B:")
for row in matrix_B:
    print(' '.join(map(str, row)))

print("\nResultant Matrix:")
for row in result_matrix:
    print(' '.join(map(str, row)))
```

Assignment 3: Pascal's Triangle

Problem Statement: Write a Python program to generate Pascal's Triangle up to n rows.

Sample Input:

Enter the number of rows for Pascal's Triangle: 5

Sample Output:

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

Solution:

```
# Pascal's Triangle
```

```
n = int(input("Enter the number of rows for Pascal's Triangle: "))


for i in range(n):
    coefficient = 1
    for j in range(1, n - i + 1):
        print(" ", end="")
    for k in range(0, i + 1):
        print(coefficient, end=" ")
        coefficient = coefficient * (i - k) // (k + 1)
    print()
```

Assignment 4: Caesar Cipher

Problem Statement: Write a Python program to implement the Caesar Cipher. Allow the user to input a message and a shift value, then encrypt and print the resulting message.

Sample Input:

Enter the message: Hello, World!

Enter the shift value: 3

Sample Output:

Encrypted Message: Khoor, Zruog!

Solution:

```
# Caesar Cipher
message = input("Enter the message: ")
shift = int(input("Enter the shift value: "))

encrypted_message = ""

for char in message:
    if 'A' <= char <= 'Z':
        encrypted_char = chr(((ord(char) - ord('A') + shift) % 26) + ord('A'))
        #the ord() function is used to get the ASCII value of a character.
            #The ord() function takes a character as an argument and returns its corresponding ASCII value.
        encrypted_message += encrypted_char
    elif 'a' <= char <= 'z':
        encrypted_char = chr(((ord(char) - ord('a') + shift) % 26) + ord('a'))
        encrypted_message += encrypted_char
    else:
        encrypted_message += char

print(f"\nEncrypted Message: {encrypted_message}")
```

Assignment 5: Diamond Pattern

Problem Statement: Write a Python program to print a diamond pattern of asterisks given the number of rows.

Sample Input:

Enter the number of rows: 5

Sample Output:

```
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
```

Solution:

```
# Diamond Pattern
n = int(input("Enter the number of rows: "))

# Upper half of the diamond
for i in range(1, n + 1, 2):
    spaces = " " * ((n - i) // 2)
    stars = "*" * i
    print(spaces + stars)

# Lower half of the diamond
for i in range(n - 2, 0, -2):
    spaces = " " * ((n - i) // 2)
    stars = "*" * i
    print(spaces + stars)
```

Assignment 6: Hollow Pyramid Pattern

Problem Statement: Write a Python program to print a hollow pyramid pattern of asterisks given the number of rows.

Sample Input:

Enter the number of rows: 5

Sample Output:

```
    *
   * *
  *   *
 *     *
*********
```

Solution:

```
# Hollow Pyramid Pattern
n = int(input("Enter the number of rows: "))

# Top part of the pyramid
for i in range(1, n):
    spaces = " " * (n - i)
    if i == 1:
        print(spaces + "*")
    else:
        stars = " " + " " * (2 * (i - 1) - 1) + " "
        print(spaces + "*" + stars + "*")
```

```
# Bottom part of the pyramid
print("*" * (2 * n - 1))
```

Assignment 7: Number Pattern

Problem Statement: Write a Python program to print the following number pattern.

Sample Output:

1

22

333

4444

55555

666666

7777777

88888888

999999999

Solution:

```
# Number Pattern
for i in range(1, 10):
    print(str(i) * i)
```

Assignment 8: Diamond Pattern with Numbers

Problem Statement: Write a Python program to print a diamond pattern of numbers, where each row contains consecutive numbers.

Sample Input:

Enter the number of rows: 5

Sample Output:

```
    1
   232
  34543
 4567654
567898765
 4567654
  34543
   232
    1
```

Solution:

```
# Diamond Pattern with Numbers
n = int(input("Enter the number of rows: "))

# Upper half of the diamond
for i in range(1, n + 1):
    spaces = " " * (n - i)
    numbers = "".join(str(j) for j in range(i, 2 * i))
    print(spaces + numbers + numbers[-2::-1])

# Lower half of the diamond
for i in range(n - 1, 0, -1):
    spaces = " " * (n - i)
```

```
    numbers = "".join(str(j) for j in range(i, 2 * i))
    print(spaces + numbers + numbers[-2::-1])
```

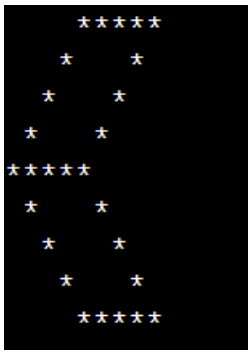
Assignment 9: Hollow Rhombus Pattern

Problem Statement: Write a Python program to print a hollow rhombus pattern of asterisks given the number of rows.

Sample Input:

Enter the number of rows: 5

Sample Output:



Solution:

```
# Hollow Rhombus Pattern
n = int(input("Enter the number of rows: "))

# Upper half of the rhombus
for i in range(1, n + 1):
    spaces = " " * (n - i)
    if i == 1 or i == n:
        stars = "*" * n
    else:
```

```
        stars = "*" + " " * (n - 2) + "*"
    print(spaces + stars)

# Lower half of the rhombus
for i in range(n - 1, 0, -1):
    spaces = " " * (n - i)
    if i == 1 or i == n:
        stars = "*" * n
    else:
        stars = "*" + " " * (n - 2) + "*"
    print(spaces + stars)
```

Assignment 10: Spiral Pattern

Problem Statement: Write a Python program to print a spiral pattern of numbers from 1 to n x n, where n is an odd number.

Sample Input:

Enter the size of the spiral: 5

Sample Output:

1 2 3 4 5

16 17 18 19 6

15 24 25 20 7

14 23 22 21 8

13 12 11 10 9

Solution:

```
# Spiral Pattern
n = int(input("Enter the size of the spiral: "))
matrix = [[0] * n for _ in range(n)]

# Fill the matrix in spiral order
num = 1
top, bottom, left, right = 0, n - 1, 0, n - 1

while num <= n * n:
    for i in range(left, right + 1):
        matrix[top][i] = num
        num += 1
    top += 1

    for i in range(top, bottom + 1):
        matrix[i][right] = num
        num += 1
    right -= 1

    for i in range(right, left - 1, -1):
        matrix[bottom][i] = num
        num += 1
```

```
    bottom -= 1

    for i in range(bottom, top - 1, -1):
        matrix[i][left] = num
        num += 1
    left += 1

# Print the spiral pattern
for row in matrix:
    for elem in row:
        print(f"{elem:3}", end=" ")
    print()
```

Assignment 11: Cross Pattern

Problem Statement: Write a Python program to print a cross pattern of asterisks given the number of rows and columns.

Sample Input:

Enter the number of rows: 7

Enter the number of columns: 7

Sample Output:

```
*     *
 *   *
  * *
   *
  * *
 *   *
```

* *

Solution:

```
# Cross Pattern
rows = int(input("Enter the number of rows: "))
cols = int(input("Enter the number of columns: "))

for i in range(rows):
    for j in range(cols):
        if i == j or i + j == rows - 1:
            print("*", end=" ")
        else:
            print(" ", end=" ")
    print()
```

Assignment 12: Word Frequency Counter

Problem Statement: Write a Python program to count the frequency of each word in a given text.

Sample Input:

Enter the text: This is a sample text. This text is used for testing. Sample text for counting word frequency.

Sample Output:

Word Frequency:

This: 2

is: 2

a: 1

sample: 2

text.: 2

This: 2

text: 2

is: 2

used: 1

for: 1

testing.: 1

Sample: 1

text: 2

for: 1

counting: 1

word: 1

frequency.: 1

Solution:

```
# Word Frequency Counter
text = input("Enter the text: ")
words = text.split()

word_frequency = {}
for word in words:
    if word not in word_frequency:
        word_frequency[word] = 1
    else:
```

```
        word_frequency[word] += 1


print("\nWord Frequency:")
for word, frequency in word_frequency.items():
    print(f"{word}: {frequency}")
```

Assignment 13: Shopping Cart with Discounts

Problem Statement: Create a shopping cart program using dictionaries to store information about items and their prices. Implement the following operations:

1. Add an item to the cart.
2. Display the contents of the cart.
3. Calculate the total amount with discounts based on the number of items.

Sample Input:

1. Add item to cart
2. Display cart contents
3. Calculate total amount
4. Exit

Enter your choice: 1

Enter item name: Laptop

Enter item price: 1000

Enter your choice: 1

Enter item name: Mouse

Enter item price: 20

Enter your choice: 2

Sample Output:

Cart Contents:

Item: Laptop, Price: $1000

Item: Mouse, Price: $20

Solution:

```
# Shopping Cart with Discounts
cart = {}

while True:
    print("1. Add item to cart\n2. Display cart contents\n3. Calculate total amount\n4. Exit")
    choice = int(input("Enter your choice: "))

    if choice == 1:
        item_name = input("Enter item name: ")
        item_price = float(input("Enter item price: "))
        cart[item_name] = item_price
    elif choice == 2:
        print("\nCart Contents:")
        for item, price in cart.items():
            print(f"Item: {item}, Price: ${price}")
    elif choice == 3:
        total_amount = sum(cart.values())
```

```
        if len(cart) >= 5:
            total_amount *= 0.9  # 10% discount for 5 or more items
        print(f"\nTotal Amount: ${total_amount:.2f}")
    elif choice == 4:
        break
```

Assignment 14: Password Validator

Problem Statement: Write a Python program to validate passwords based on the following rules:

- The password must be at least 8 characters long.
- The password must contain at least one uppercase letter, one lowercase letter, and one digit.

Implement the following operations:

1. Enter a password.
2. Check if the password is valid according to the rules.

Sample Input:

1. Enter a password

2. Check password validity

3. Exit

Enter your choice: 1

Enter the password: MySecureP@ss

Sample Output:

Password entered successfully.

Enter your choice: 2

Solution:

```
# Password Validator
password = ""

while True:
    print("1. Enter a password\n2. Check password validity\n3. Exit")
    choice = int(input("Enter your choice: "))

    if choice == 1:
        password = input("Enter the password: ")
        print("Password entered successfully.")
    elif choice == 2:
        if len(password) >= 8:
            has_uppercase = any('A' <= char <= 'Z' for char in password)
            has_lowercase = any('a' <= char <= 'z' for char in password)
            has_digit = any('0' <= char <= '9' for char in password)

            if has_uppercase and has_lowercase and has_digit:
                print("Password is valid.")
            else:
                print("Password is invalid. Please follow the rules.")
        else:
            print("Password is too short. Please enter a password with at least 8 characters.")
    elif choice == 3:
        break
```

Assignment 15: Anagram Checker

Problem Statement: Write a Python program to check if two given strings are anagrams or not. An anagram is a word or phrase formed by rearranging the letters of another.

Implement the following operations:

1. Enter two strings.
2. Check if the strings are anagrams.

Sample Input:

1. Enter the first string

2. Enter the second string

3. Check if they are anagrams

4. Exit

Enter your choice: 1

Enter the first string: listen

Sample Output:

First string entered successfully.

Enter your choice: 2

Solution:

```
# Anagram Checker

string1 = ""

string2 = ""


while True:
```

```
    print("1. Enter the first string\n2. Enter the second string\n3. Check if they are anagrams\n4. Exit")
    choice = int(input("Enter your choice: "))

    if choice == 1:
        string1 = input("Enter the first string: ")
        print("First string entered successfully.")
    elif choice == 2:
        string2 = input("Enter the second string: ")
        print("Second string entered successfully.")
    elif choice == 3:
        # Remove spaces and convert to lowercase for comparison
        cleaned_string1 = ''.join(char.lower() for char in string1 if char.isalnum())
        cleaned_string2 = ''.join(char.lower() for char in string2 if char.isalnum())

        # Check if the strings are anagrams
        if sorted(cleaned_string1) == sorted(cleaned_string2):
            print("The strings are anagrams.")
        else:
            print("The strings are not anagrams.")
    elif choice == 4:
        break
```