**C.H.M.E.S**

**Dr. Moonje Institute of Management & Computer Studies**

**Nasik-5**

**A.Y 2024-2025**

**MCA I**

**Term 1 sem 1**

**Practical List -Python Programming**

**2024 Pattern**

**Practical No:1**

**Aim:** WAP to Display all types of pyramids of star

**Code:**

def print\_right\_angle\_triangle(n):

    print("Right-Angle Triangle:")

*for* i *in* range(1, n *+* 1):

        print('\*' *\** i)

    print()

def print\_isosceles\_triangle(n):

    print("Isosceles Triangle:")

*for* i *in* range(1, n *+* 1):

        print(' ' *\** (n *-* i) *+* '\*' *\** (2 *\** i *-* 1))

    print()

def print\_inverted\_triangle(n):

    print("Inverted Triangle:")

*for* i *in* range(n, 0, *-*1):

        print('\*' *\** i)

    print()

def print\_full\_pyramid(n):

    print("Full Pyramid:")

*for* i *in* range(1, n *+* 1):

        print(' ' *\** (n *-* i) *+* '\*' *\** (2 *\** i *-* 1))

    print()

def print\_diamond(n):

    print("Diamond Shape:")

*# Upper part*

*for* i *in* range(1, n *+* 1):

        print(' ' *\** (n *-* i) *+* '\*' *\** (2 *\** i *-* 1))

*# Lower part*

*for* i *in* range(n *-* 1, 0, *-*1):

        print(' ' *\** (n *-* i) *+* '\*' *\** (2 *\** i *-* 1))

    print()

*# Set the height of the pyramids*

n *=* 5

print\_right\_angle\_triangle(n)

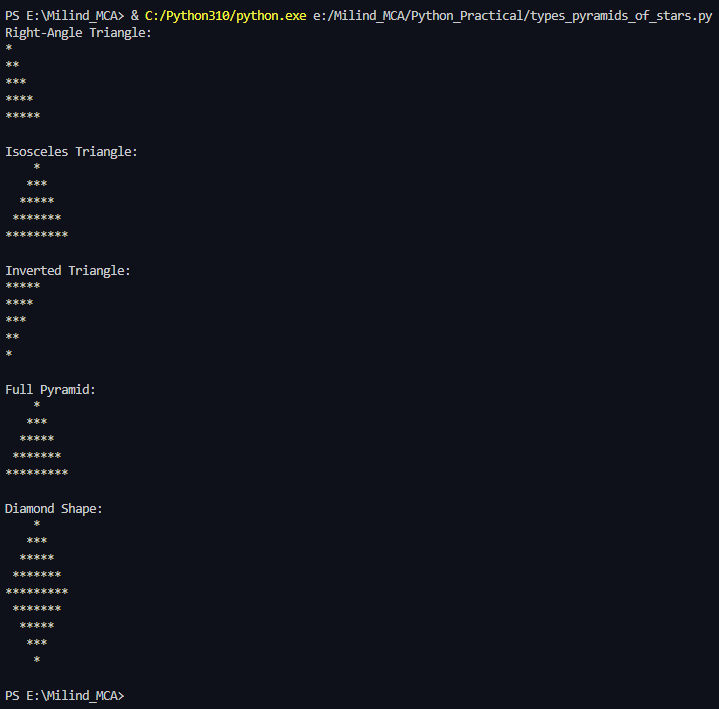
print\_isosceles\_triangle(n)

print\_inverted\_triangle(n)

print\_full\_pyramid(n)

print\_diamond(n)

**Output:**



**Practical No:2**

**Aim:** WAP to Display multiplication tables of all numbers.

**Code:**

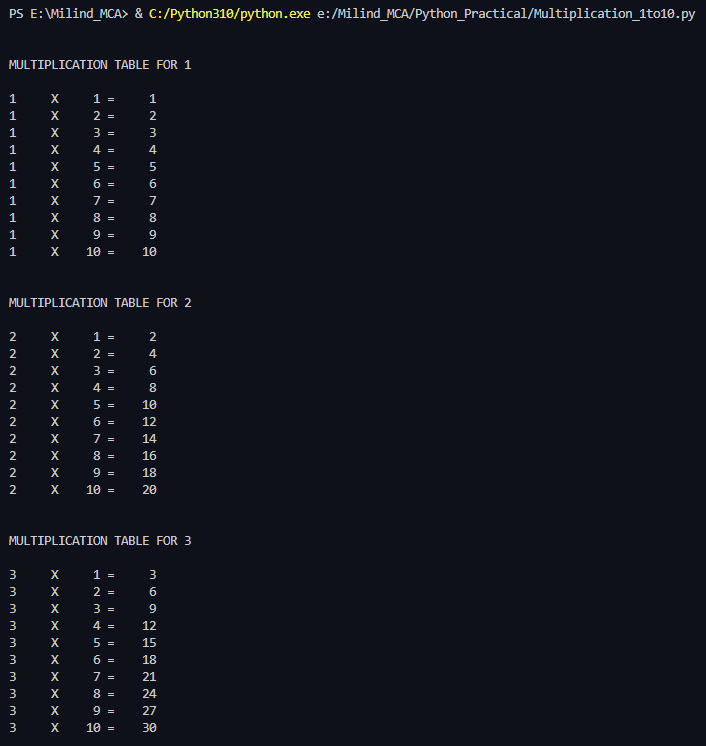
*for* i *in* range(1,11):

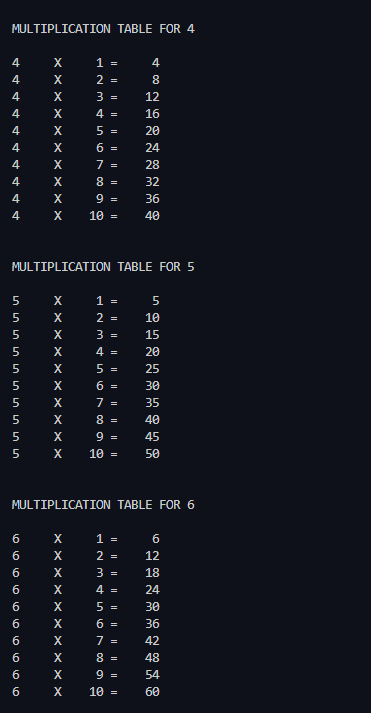
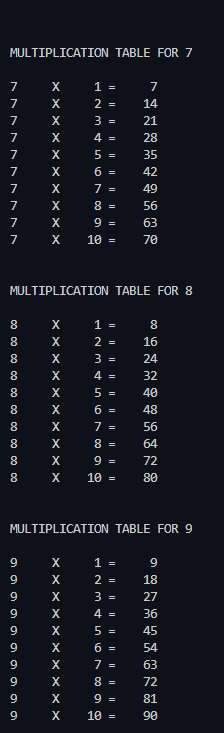
    print("\n\nMULTIPLICATION TABLE FOR %d\n" *%*(i))

*for* j *in* range(1,11):

        print("%-5d X %5d = %5d" *%* (i, j, i*\**j))

**Output:**



**Practical No:3**

**Aim:** WAP to implement tower of Hanoi.

**Code:**

**Output:**

**Practical No:4**

**Aim:** WAP to calculate simple intrest using a user defined function. Accept amount, duration from user. Set Intrest rate as default parameter.

**Code:**

def calculate\_simple\_interest(principal, rate, duration):

*return* (principal *\** rate *\** duration) */* 100

principal *=* float(input("Enter the principal amount: "))

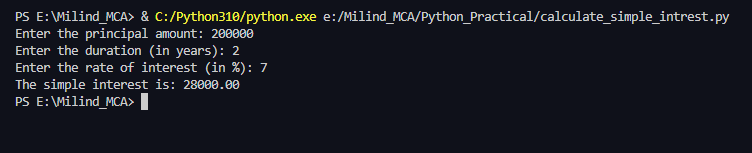
duration *=* float(input("Enter the duration (in years): "))

rate *=* float(input("Enter the rate of interest (in %): "))

simple\_interest *=* calculate\_simple\_interest(principal, rate, duration)

print(f"The simple interest is: {simple\_interest:.2f}")

**Output:**



**Practical No:5**

**Aim:** WAP to count even and odd numbers in a list

**Code:**

def count\_even\_odd(numbers):

    even\_count *=* 0

    odd\_count *=* 0

*for* number *in* numbers:

*if* number *%* 2 *==* 0:

            even\_count *+=* 1

*else*:

            odd\_count *+=* 1

*return* even\_count, odd\_count

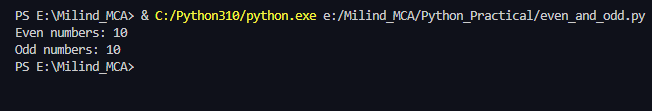
numbers *=* [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

even\_count, odd\_count *=* count\_even\_odd(numbers)

print(f"Even numbers: {even\_count}")

print(f"Odd numbers: {odd\_count}")

**Output:**



**Practical No:6**

**Aim:** WAP to find sum of all numbers, min, max, mean, mode of numbers in a list.

**Code:**

*from* collections *import* Counter

def calculate\_statistics(numbers):

*if* *not* numbers:

*return* None, None, None, None, None

*# Calculate sum*

    total\_sum *=* sum(numbers)

*# Calculate minimum*

    minimum *=* min(numbers)

*# Calculate maximum*

    maximum *=* max(numbers)

*# Calculate mean*

    mean *=* total\_sum */* len(numbers)

*# Calculate mode*

    frequency *=* Counter(numbers)

    mode\_data *=* frequency.most\_common()

    mode *=* [num *for* num, freq *in* mode\_data *if* freq *==* mode\_data[0][1]]

*return* total\_sum, minimum, maximum, mean, mode

*# Example usage*

numbers *=* [1, 2, 2, 3, 4, 4, 4, 5]

total\_sum, minimum, maximum, mean, mode *=* calculate\_statistics(numbers)

print(f"Sum: {total\_sum}")

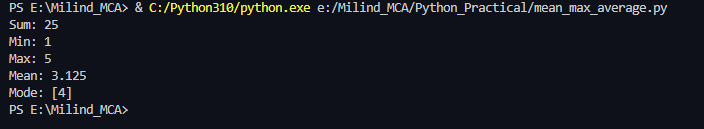
print(f"Min: {minimum}")

print(f"Max: {maximum}")

print(f"Mean: {mean}")

print(f"Mode: {mode}")

**Output:**



**Practical No:7**

**Aim:** WAP to find sum & multiplication of two matrices implement using list.

**Code:**

def add\_matrices(matrix1, matrix2):

*# Check if dimensions are the same*

*if* len(matrix1) *!=* len(matrix2) *or* len(matrix1[0]) *!=* len(matrix2[0]):

*raise* ValueError("Matrices must have the same dimensions for addition.")

    result *=* []

*for* i *in* range(len(matrix1)):

        row *=* []

*for* j *in* range(len(matrix1[0])):

            row.append(matrix1[i][j] *+* matrix2[i][j])

        result.append(row)

*return* result

def multiply\_matrices(matrix1, matrix2):

*# Check if matrices can be multiplied*

*if* len(matrix1[0]) *!=* len(matrix2):

*raise* ValueError("Number of columns in the first matrix must be equal to the number of rows in the second matrix.")

    result *=* []

*for* i *in* range(len(matrix1)):

        row *=* []

*for* j *in* range(len(matrix2[0])):

            sum\_product *=* 0

*for* k *in* range(len(matrix2)):

                sum\_product *+=* matrix1[i][k] *\** matrix2[k][j]

            row.append(sum\_product)

        result.append(row)

*return* result

*# Example usage:*

matrix1 *=* [

    [1, 2, 3],

    [4, 5, 6],

]

matrix2 *=* [

    [7, 8, 9],

    [1, 2, 3],

]

*# Matrix Addition*

sum\_result *=* add\_matrices(matrix1, matrix2)

print("Sum of matrices:")

*for* row *in* sum\_result:

    print(row)

*# Matrix Multiplication*

matrix3 *=* [

    [1, 2],

    [3, 4],

    [5, 6],

]

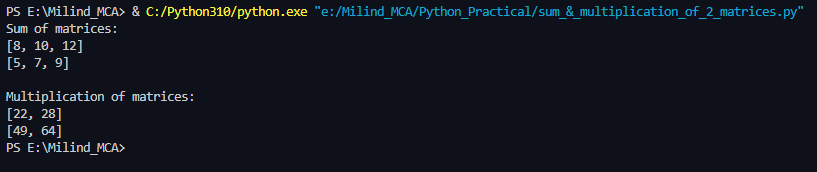
multiplication\_result *=* multiply\_matrices(matrix1, matrix3)

print("\nMultiplication of matrices:")

*for* row *in* multiplication\_result:

    print(row)

**Output:**



**Practical No:1**

**Aim:** WAP to Display all types of pyramids of star

**Code:**

def print\_right\_angle\_triangle(n):

    print("Right-Angle Triangle:")

*for* i *in* range(1, n *+* 1):

        print('\*' *\** i)

    print()

def print\_isosceles\_triangle(n):

    print("Isosceles Triangle:")

*for* i *in* range(1, n *+* 1):

        print(' ' *\** (n *-* i) *+* '\*' *\** (2 *\** i *-* 1))

    print()

def print\_inverted\_triangle(n):

    print("Inverted Triangle:")

*for* i *in* range(n, 0, *-*1):

        print('\*' *\** i)

    print()

def print\_full\_pyramid(n):

    print("Full Pyramid:")

*for* i *in* range(1, n *+* 1):

        print(' ' *\** (n *-* i) *+* '\*' *\** (2 *\** i *-* 1))

    print()

def print\_diamond(n):

    print("Diamond Shape:")

*# Upper part*

*for* i *in* range(1, n *+* 1):

        print(' ' *\** (n *-* i) *+* '\*' *\** (2 *\** i *-* 1))

*# Lower part*

*for* i *in* range(n *-* 1, 0, *-*1):

        print(' ' *\** (n *-* i) *+* '\*' *\** (2 *\** i *-* 1))

    print()

*# Set the height of the pyramids*

n *=* 5

print\_right\_angle\_triangle(n)

print\_isosceles\_triangle(n)

print\_inverted\_triangle(n)

print\_full\_pyramid(n)

print\_diamond(n)

**Output:**

