Q1) Write a program to calculate f(x) = sin(x) where x is in degree.

```
import math

def sin_degrees(x):
    # Convert degrees to radians
    radians = math.radians(x)

    # Calculate sine of radians
    sine = math.sin(radians)

    return sine
```

Q2) Create a function that accepts a list of numbers and returns the sum of elements on the list.
For example:
If a list **[1, 3, 6, 8, 12]** is passed as an argument to a function, the function should return a value of **30**.

```
def sum_list(numbers):
 return sum(numbers)  #The sum() function is a built-in Python function that calculates the sum of a list of numbers.
my_list = [1, 3, 6, 8, 12]
total = sum_list(my_list)  #The sum_list() function simply calls this function with the numbers argument and returns the result.
print(total)
```

```
30
```

Q3) Write a program to calculate the sum of digit of a number using fuction.

```python
def sum_of_digits(number):
    # Convert the number to a string and loop through each character
    # in the string, converting each character back to an integer and
    # adding it to a running total.
    total = 0
    for digit in str(number):
        total += int(digit)
    return total

# Prompt the user to enter a number and call the function to calculate
# the sum of its digits.
number = int(input("Enter a number: "))
digit_sum = sum_of_digits(number)
print("The sum of the digits in {} is {}".format(number, digit_sum))
```

```
Enter a number: 2
The sum of the digits in 2 is 2
```

Q4) Write a program to calculate the smallest divisor of a

number using function. for example: samllest divisor of 15 is 3.

```python
def smallest_divisor(number):
    # Loop through all numbers from 2 to the square root of the input number.
    # If the input number is divisible by the current number, return the current
    # number as the smallest divisor. Otherwise, continue looping.
    for i in range(2, int(number**0.5) + 1):
        if number % i == 0:
            return i
    # If no divisor is found between 2 and the square root of the input number,
    # the input number must be prime, so return the input number itself.
    return number

# Prompt the user to enter a number and call the function to calculate
# the smallest divisor.
number = int(input("Enter a number: "))
divisor = smallest_divisor(number)
print("The smallest divisor of {} is {}".format(number, divisor))
```

```
Enter a number: 34
The smallest divisor of 34 is 2
```

Q5) Write a program to check a given number is perfect number or not using function.

A positive integer is called a perfect number if it is equal to the sum of all of its divisors, including 1 but excluding the number itself. For example, 6 = 1 + 2 + 3.

```
def is_perfect_number(number):
    # Initialize a variable to hold the sum of divisors.
    divisor_sum = 0
    # Loop through all numbers from 1 to the input number (excluding the input number itself).
    # If the current number is a divisor of the input number, add it to the divisor sum.
    for i in range(1, number):
        if number % i == 0:
            divisor_sum += i
    # If the divisor sum is equal to the input number, the input number is a perfect number.
    if divisor_sum == number:
        return True
    else:
        return False

# Prompt the user to enter a number and call the function to check if it's a perfect number.
number = int(input("Enter a number: "))
if is_perfect_number(number):
    print("{} is a perfect number".format(number))
else:
    print("{} is not a perfect number".format(number))
```

```
Enter a number: 23
23 is not a perfect number
```

Q6) Write a function that takes two integers m and n as arguments and prints out an **m×n** box consisting of asterisks. for example: **rectangle(3,4)** should print following output:

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

```python
def rectangle(m, n):
    # Loop through all rows from 1 to m.
    for i in range(1, m+1):
        # Initialize an empty string to hold the current row.
        row = ""
        # Loop through all columns from 1 to n.
        for j in range(1, n+1):
            # Add an asterisk to the current row.
            row += "* "
        # Print the current row to the console.
        print(row)

# Call the function to print a 3 x 4 box of asterisks.
rectangle(3, 4)
```

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

Q7) Write a python program to create a matrix of dimensions m x n without using any additional libraries and display the values.

```python
def create_matrix(m, n):
    # Initialize an empty list to hold the matrix.
    matrix = []
    # Loop through all rows from 1 to m.
    for i in range(m):
        # Initialize an empty list to hold the current row.
        row = []
        # Loop through all columns from 1 to n.
        for j in range(n):
            # Append the current column number to the current row.
            row.append(j+1)
        # Append the current row to the matrix.
        matrix.append(row)
    # Return the matrix.
    return matrix

# Call the function to create a 3 x 4 matrix and print the values.
matrix = create_matrix(3, 4)
for row in matrix:
    print(row)
```

```
[1, 2, 3, 4]
[1, 2, 3, 4]
[1, 2, 3, 4]
```

Q8) Write a program for addition of two matrices.

```
def add_matrices(matrix1, matrix2):
    # Get the dimensions of the matrices.
    m = len(matrix1)
    n = len(matrix1[0])
    # Initialize an empty list to hold the result.
    result = []
    # Loop through all rows from 0 to m-1.
    for i in range(m):
        # Initialize an empty list to hold the current row.
        row = []
        # Loop through all columns from 0 to n-1.
        for j in range(n):
            # Add the corresponding values in the two matrices and append the result to the current row.
            row.append(matrix1[i][j] + matrix2[i][j])
        # Append the current row to the result.
        result.append(row)
    # Return the result.
    return result

# Define the matrices to add.
matrix1 = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
matrix2 = [[9, 8, 7], [6, 5, 4], [3, 2, 1]]
```

```
# Call the function to add the matrices and print the result.
result = add_matrices(matrix1, matrix2)
for row in result:
    print(row)
```

```
[10, 10, 10]
[10, 10, 10]
[10, 10, 10]
```

Q9) Write a program to identify the given matrix is diagonal matrix or not.

```python
def is_diagonal(matrix):
    # Get the dimensions of the matrix.
    m = len(matrix)
    n = len(matrix[0])
    # Loop through all rows from 0 to m-1.
    for i in range(m):
        # Loop through all columns from 0 to n-1.
        for j in range(n):
            # Check if the current element is non-diagonal and not zero.
            if i != j and matrix[i][j] != 0:
                # If so, the matrix is not diagonal.
                return False
    # If we haven't returned False yet, the matrix is diagonal.
    return True

# Define a matrix to check.
matrix = [[1, 0, 0], [0, 2, 0], [0, 0, 3]]
```

```python
# Call the function to check if the matrix is diagonal and print the result.
if is_diagonal(matrix):
    print("The matrix is diagonal.")
else:
    print("The matrix is not diagonal.")
```

```
The matrix is diagonal.
```

Q10) Write a program to calculate trace of a matrix

```python
def matrix_trace(matrix):
    # Get the dimensions of the matrix.
    m = len(matrix)
    n = len(matrix[0])
    # Initialize a variable to hold the trace.
    trace = 0
    # Loop through the diagonal elements.
    for i in range(min(m, n)):
        trace += matrix[i][i]
    # Return the trace.
    return trace

# Define a matrix to calculate the trace of.
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

# Call the function to calculate the trace and print the result.
trace = matrix_trace(matrix)
print("The trace of the matrix is", trace)
```

```
The trace of the matrix is 15
```