1.Write a program to find whether a number is a prime number

```
import math
n = int(input("Enter a number:"))
for i in range(2, int(math.sqrt(n)) + 1):
    if n % i == 0:
        print(n, "is not a prime number")
        break
else:
    print(n, "is a prime number")
```

2. Write a program to print m raise to power n, where m and n are read from the user.

```
m = int(input("Enter the base (m): "))
n = int(input("Enter the exponent (n): "))
result = m ** n
print(m, "raised to the power", n, "is:", result)
```

3. Write a program having a parameterized function that returns True or False depending on whether the parameter passed is even or odd.

```
def is_even(number):
    if number % 2 == 0:
        return True
    else:
        return False

num = int(input("Enter a number: "))

if is_even(num):
    print(num, "is even.")
else:
    print(num, "is odd.")
```

4. Write a program to print the summation of the following series upto n terms: 1-2+3-4+5-6+7-------n.

```
n = int(input("Enter the number of terms (n): "))

total = 0


for i in range(1, n + 1):
    if i % 2 == 0:
        total -= i
    else:
        total += i


print("Sum of the series up to", n, "terms is:", total)
```

5. Write a menu driven program to perform the following operations on strings using string built in functions. A. Find the frequency of a character in a string. B. Replace a character by another character in a string. C. Remove the first occurrence of a character from a string. D. Remove all occurrences of a character from a string.

```
def frequency(string, char):
    return string.count(char)


def replace_char(string, old_char, new_char):
    return string.replace(old_char, new_char)


def remove_first_occurrence(string, char):
    return string.replace(char, '', 1)


def remove_all_occurrences(string, char):
    return string.replace(char, '')


# Main program
while True:
    print("\nMenu:")
```

```
print("a. Find the frequency of a character in a string.")
print("b. Replace a character by another character in a string.")
print("c. Remove the first occurrence of a character from a string.")
print("d. Remove all occurrences of a character from a string.")
print("e. Exit")

choice = input("Enter your choice (a/b/c/d/e): ").lower()

if choice == 'e':
    print("Exiting the program.")
    break

string = input("Enter the string: ")

if choice == 'a':
    char = input("Enter the character to find frequency: ")
    print(f"Frequency of '{char}' in the string is: {frequency(string, char)}")

elif choice == 'b':
    old_char = input("Enter the character to be replaced: ")
    new_char = input("Enter the new character: ")
    print(f"Modified string: {replace_char(string, old_char, new_char)}")

elif choice == 'c':
    char = input("Enter the character to remove (first occurrence): ")
    print(f"Modified string: {remove_first_occurrence(string, char)}")

elif choice == 'd':
    char = input("Enter the character to remove (all occurrences): ")
    print(f"Modified string: {remove_all_occurrences(string, char)}")
```

```
    else:
        print("Invalid choice, please try again.")
```

6. Write a program that accept two strings and returns and returns the indices of all the occurrences of the second string in the first string as a list. If the second string is not present in the first string, the it should returns -1.

```
def find_all_occurrences(main_string, sub_string):
    indices = []
    index = main_string.find(sub_string)
    while index != -1:
        indices.append(index)
        index = main_string.find(sub_string, index + 1)
    return indices if indices else -1

main_str = input("Enter the main string: ")
sub_str = input("Enter the substring to search for: ")
result = find_all_occurrences(main_str, sub_str)
print("Indices:", result)
```

7.Using Numpy module write menu driven program to do following. a. Create an array fille with 1's. b. Find maximum and minimum values from an array. c. Dot product of 2 arrays. d. Reshape a 1-D array to 2-D array.

```
import numpy as np

def create_ones_array():
    shape = tuple(map(int, input("Enter dimensions separated by space (e.g. 2 3 for 2x3): ").split()))
    arr = np.ones(shape)
    print("Array filled with 1's:\n", arr)

def find_max_min():
    arr = np.array(list(map(int, input("Enter array elements separated by space: ").split())))
```

```
    print("Array:", arr)
    print("Max value:", np.max(arr))
    print("Min value:", np.min(arr))

def dot_product():
    arr1 = np.array(list(map(int, input("Enter elements of first array: ").split())))
    arr2 = np.array(list(map(int, input("Enter elements of second array: ").split())))
    if arr1.size != arr2.size:
        print("Arrays must be of the same size for dot product.")
    else:
        print("Dot product:", np.dot(arr1, arr2))

def reshape_array():
    arr = np.array(list(map(int, input("Enter elements of 1-D array: ").split())))
    rows = int(input("Enter number of rows: "))
    cols = int(input("Enter number of columns: "))
    if rows * cols != arr.size:
        print("Cannot reshape array: total elements do not match.")
    else:
        reshaped = arr.reshape((rows, cols))
        print("Reshaped 2-D array:\n", reshaped)

while True:
    print("\nMenu:")
    print("a. Create an array filled with 1's")
    print("b. Find maximum and minimum values of an array")
    print("c. Dot product of 2 arrays")
    print("d. Reshape a 1-D array to 2-D array")
    print("e. Exit")

    choice = input("Enter your choice (a/b/c/d/e): ").lower()
```

```
    if choice == 'a':
        create_ones_array()
    elif choice == 'b':
        find_max_min()
    elif choice == 'c':
        dot_product()
    elif choice == 'd':
        reshape_array()
    elif choice == 'e':
        print("Exiting the program.")
        break
    else:
        print("Invalid choice. Please try again.")
```

8. Write a function that takes a sentence as input from the user and calculates the frequency of each letter. Use a variable of dictionary type to maintain the count.

```
def letter_frequency():
    sentence = input("Enter a sentence: ")
    frequency = {}

    for char in sentence:
        if char.isalpha():
            char = char.lower()
            frequency[char] = frequency.get(char, 0) + 1

    print("Letter frequencies:")
    for letter, count in sorted(frequency.items()):
        print(letter, ":", count)

letter_frequency()
```

9. Consider a tuple t1=(1,2,5,7,9,2,4,6,8,10). Write a program to perform the following operations. a. Print contents pf t1 in two separate lines that half values come on one line and other half in the next line. b. Print all even values of t1 as another tuple t2. c. Concatenate a tuple t2=(11,13,15) with t1. d. Return maximum and minimum value from t1.

```
t1 = (1, 2, 5, 7, 9, 2, 4, 6, 8, 10)


half = len(t1) // 2

print("First half of t1:", t1[:half])

print("Second half of t1:", t1[half:])


t2 = tuple(x for x in t1 if x % 2 == 0)

print("Tuple of even values (t2):", t2)


t3 = (11, 13, 15)

concatenated = t1 + t3

print("Concatenated tuple (t1 + t3):", concatenated)


print("Maximum value in t1:", max(t1))

print("Minimum value in t1:", min(t1))
```

10. Write a function that reads a file, File1 and copies only alternative lines to another file, File2. Alternative lines copied should be the odd numbered lines. File1.txt is shown below. apples are red cherries are brown mangoes are yellow grapes are purple

```
def copy_alternate_lines(File1, File2):
   try:
      with open(File1, 'r') as f1, open(File2, 'w') as f2:
         for line_number, line in enumerate(f1, start=1):
            if line_number % 2 != 0:  # Copy odd-numbered lines
               f2.write(line)
      print("Alternate (odd-numbered) lines copied from", File1, "to", File2)
   except FileNotFoundError:
      print("Error: The file", File1, "does not exist.")
```

```
copy_alternate_lines('File1.txt', 'File2.txt')
```

11. Write a python program to handle ZeroDivisionError exception when dividing a number by zero.

```
def divide_numbers():
    try:
        numerator = float(input("Enter the numerator: "))
        denominator = float(input("Enter the denominator: "))
        result = numerator / denominator
        print("Result:", result)
    except ZeroDivisionError:
        print("Error: Division by zero is not allowed. Please try again.")
        divide_numbers()

divide_numbers()
```

12. Write a program that reads a list of integers from the user and throws an exception if any numbers are duplicates.

```
def read_unique_integers():
    try:
        numbers = list(map(int, input("Enter a list of integers separated by space: ").split()))
        unique_numbers = []

        for num in numbers:
            if num in unique_numbers:
                raise ValueError(f"Duplicate value found: {num}")
            unique_numbers.append(num)

        print("All numbers are unique:", unique_numbers)

    except ValueError as e:
```

```
        print("Error:", e)


read_unique_integers()
```

13. Write a program that makes use of a function to display sine, cosine, polynomial and exponential curves.

```
import numpy as np
import matplotlib.pyplot as plt


def plot_curves():
    x = np.linspace(-10, 10, 400)
    y_sin = np.sin(x)
    y_cos = np.cos(x)
    y_poly = x**3 - 4*x**2 + x + 6
    y_exp = np.exp(x)

    plt.figure(figsize=(12, 8))

    # Sine curve
    plt.subplot(2, 2, 1)
    plt.plot(x, y_sin, color='blue')
    plt.title("Sine Curve")
    plt.grid(True)

    # Cosine curve
    plt.subplot(2, 2, 2)
    plt.plot(x, y_cos, color='green')
    plt.title("Cosine Curve")
    plt.grid(True)
```

```
    # Polynomial curve
    plt.subplot(2, 2, 3)
    plt.plot(x, y_poly, color='red')
    plt.title("Polynomial Curve")
    plt.grid(True)

    # Exponential curve
    plt.subplot(2, 2, 4)
    plt.plot(x, y_exp, color='purple')
    plt.title("Exponential Curve")
    plt.ylim(0, 1000)
    plt.grid(True)

    plt.tight_layout()
    plt.show()

plot_curves()
```

14. Take as input in months and profits made by company ABC over a year. Represent this data using a line plot. Generated line plot

```
import matplotlib.pyplot as plt

def plot_profits():
    months = list(range(1, 13))
    profits = []

    print("Enter the profits for each of the 12 months:")
    for i in range(1, 13):
        profit = float(input(f"Profit for month {i}: "))
        profits.append(profit)
```

```
    plt.figure(figsize=(10, 5))
    plt.plot(months, profits, marker='o', linestyle='-', color='blue')
    plt.xlabel("Month Number")
    plt.ylabel("Total Profit")
    plt.title("Monthly Profit of Company ABC")
    plt.grid(True)
    plt.xticks(months)
    plt.show()

plot_profits()
```