1.Write a program to find out a factorial of given number.

```
def factorial_iterative(n):
    """ Function to calculate factorial of a number iteratively """
    factorial = 1
    for i in range(1, n + 1):
        factorial *= i
    return factorial
def factorial_recursive(n):
    """ Function to calculate factorial of a number recursively """
    if n == 0:
        return 1
    else:
        return n * factorial_recursive(n - 1)
num = int(input("Enter a number to find its factorial: "))
factorial_iter = factorial_iterative(num)
print(f"Factorial of {num} (using iterative method) is: {factorial_iter}")
factorial_rec = factorial_recursive(num)
print(f"Factorial of {num} (using recursive method) is: {factorial_rec}")
```

2.Write a program to find out odd and even number up to given number.

```
def find_odd_even_numbers(n):
    odd_numbers = []
    even_numbers = []
 for num in range(1, n + 1):
        if num % 2 == 0:
            even_numbers.append(num)
        else:
            odd_numbers.append(num)
    return odd_numbers, even_numbers
def main():
    n = int(input("Enter a number: "))
```

```
    odd_nums, even_nums = find_odd_even_numbers(n)
    print(f"Odd numbers up to {n}: {odd_nums}")
    print(f"Even numbers up to {n}: {even_nums}")
  if __name__ == "__main__":
    main()
```

3.Write a program to find out the given number is prime or not.

```
def is_prime(number):
    if number <= 1:
        return False
    elif number == 2:
        return True
    elif number % 2 == 0:
        return False
    else:
         import math
        sqrt_num = int(math.sqrt(number)) + 1
        for divisor in range(3, sqrt_num, 2):
            if number % divisor == 0:
                return False
        return True
def main():
      num = int(input("Enter a number: "))
   if is_prime(num):
        print(f"{num} is a prime number.")
    else:
        print(f"{num} is not a prime number.")
if __name__ == "__main__":
    main()
```

4.5write a program to find out the given number is parindrome or not.

```
def is_palindrome(number):
    number_str = str(number)
    if number_str == number_str[::-1]:
        return True
    else:
        return False
num = int(input("Enter a number: "))
if is_palindrome(num):
    print(f"{num} is a palindrome")
else:
    print(f"{num} is not a palindrome")
```

5.Write a program to find out the given number is Armstrong or not.

```
def is_armstrong(number):
    num_str = str(number)
    num_digits = len(num_str)
    sum_of_powers = sum(int(digit) ** num_digits for digit in num_str)
    if sum_of_powers == number:
        return True
    else:
        return False
num = int(input("Enter a number: "))
if is_armstrong(num):
    print(f"{num} is an Armstrong number.")
else:
    print(f"{num} is not an Armstrong number.")
```

6.Write a program to demonstrate class, object, Inheritance.

```
class Animal:
```

```
    def __init__(self, name):
        self.name = name
      def speak(self):
        raise NotImplementedError("Subclass must implement abstract method")


class Dog(Animal):
    def speak(self):
        return f"{self.name} says Woof!"
class Cat(Animal):
    def speak(self):
        return f"{self.name} says Meow!"
class Duck:
    def __init__(self, name):
        self.name = name
      def speak(self):
        return f"{self.name} says Quack!"
def main():
    dog = Dog("dog")
    cat = Cat(" cat")
    duck = Duck("duck")


     print(dog.speak())
    print(cat.speak())
    print(duck.speak())


if __name__ == "__main__":
    main()
```

7.Write a python program for Liner search.

```
def linear_search(arr, target):
```

```
    """
    Perform linear search to find the index of the target in the array.
  Parameters:
    arr (list): A list of elements to search through.
    target: The element to search for within the list.
  Returns:
    int: Index of the target element if found, otherwise -1.
    """
    for i in range(len(arr)):
        if arr[i] == target:
            return i
    return -1
if __name__ == "__main__":
    my_list = [4, 2, 7, 1, 9, 5]
    target_value = 5
    result_index = linear_search(my_list, target_value)
      if result_index != -1:
        print(f"Target {target_value} found at index {result_index}.")
    else:
        print(f"Target {target_value} not found in the list.")
```

8.Program to Create a file, Write in to file, read a file, append the file.

```
def create_file(filename):
    try:
        with open(filename, 'x') as f:
            print(f"File '{filename}' created successfully.")
    except FileExistsError:
        print(f"File '{filename}' already exists.")
def write_to_file(filename, content):
    with open(filename, 'w') as f:
```

```
        f.write(content)
        print(f"Content written to '{filename}' successfully.")
def read_file(filename):
    try:
        with open(filename, 'r') as f:
            content = f.read()
            print(f"Content of '{filename}':")
            print(content)
    except FileNotFoundError:
        print(f"File '{filename}' not found.")
def append_to_file(filename, additional_content):
    with open(filename, 'a') as f:
        f.write(additional_content)
        print(f"Additional content appended to '{filename}' successfully.")
def main():
    filename = "example_file.txt"
    content = "Hello, this is some content.\n"
    additional_content = "This is additional content to append.\n"

     create_file(filename)
     write_to_file(filename, content)
     read_file(filename)
     append_to_file(filename, additional_content)
     read_file(filename)

if __name__ == "__main__":
    main()
```

9.Demonstrate List and Dictionary with its in-rpoftant function (minimum 3).

```
def demonstrate_lists():
```

```
    my_list = [3, 1, 4, 1, 5, 9, 2, 6, 5]
    print("Original List:", my_list)
    print("First element:", my_list[0])
    print("Last element:", my_list[-1])
    my_list.append(8)
    print("After append:", my_list)
    my_list.remove(1)
    print("After remove(1):", my_list)
    my_list.sort()
    print("Sorted List:", my_list)
demonstrate_lists()
```

10.Develop program to demonstrate GUI programming using Tkinter.

```
import tkinter as tk
from tkinter import messagebox
def update_label():
    label.config(text="Button clicked!")
root = tk.Tk()
root.title("Tkinter Demo")

label = tk.Label(root, text="Hello, Tkinter!", padx=20, pady=20)
label.pack()

button = tk.Button(root, text="Click me!", command=update_label)
button.pack()

root.mainloop()
```

11.write a program to calculate the addition of odd nutnber up to given range.

```
def sum_of_odd_numbers(up_to):
```

```
    total_sum = 0
    for num in range(1, up_to + 1, 2):
        total_sum += num
    return total_sum
up_to = int(input("Enter the upper limit: "))
result = sum_of_odd_numbers(up_to)
print(f"The sum of odd numbers up to {up_to} is: {result}")
```

12.Write a program to calculate the addition of even number up to given range.

```
def sum_of_even_numbers(up_to):
    total_sum = 0
    for num in range(2, up_to + 1, 2):
        total_sum += num
    return total_sum
up_to = int(input("Enter the upper limit: "))
result = sum_of_even_numbers(up_to)
print(f"The sum of even numbers up to {up_to} is: {result}")
```

13.Develop program for data structure algorithm using python for sorting.

```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]


def selection_sort(arr):
    n = len(arr)
    for i in range(n)
 min_idx = i
```

```
        for j in range(i+1, n):
            if arr[j] < arr[min_idx]:
                min_idx = j
        arr[i], arr[min_idx] = arr[min_idx], arr[i]

def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return quick_sort(left) + middle + quick_sort(right)

arr1 = [64, 34, 25, 12, 22, 11, 90]
arr2 = arr1.copy()
arr3 = arr1.copy()

bubble_sort(arr1)
print("Bubble Sort:", arr1)

selection_sort(arr2)
print("Selection Sort:", arr2)

arr3 = quick_sort(arr3)
print("Quick Sort:", arr3)
```