1.write a program to demonstrate Hierarchical inheritance

```
class Animal:
    def speak(self):
        print("Animal makes a sound")
class Dog(Animal):
    def speak(self):
        print("Dog barks")
class Cat(Animal):
    def speak(self):
        print("Cat meows")

dog = Dog()
cat = Cat()
dog.speak()  # Outputs: Dog barks
cat.speak()  # Outputs: Cat meows
```

2.write a program to display all prime numbers within an interval

```
def is_prime(num):
    if num <= 1:
        return False
    for i in range(2, int(num ** 0.5) + 1):
        if num % i == 0:
            return False
    return True

def display_primes_in_interval(start, end):
    for num in range(start, end + 1):
        if is_prime(num):
            print(num, end=" ")

start = int(input("Enter the start of the interval: "))
end = int(input("Enter the end of the interval: "))

print(f"Prime numbers between {start} and {end} are:")
display_primes_in_interval(start, end)
```

3. write a program to print multiplication table of a given number

```
num = int(input("Enter a number: "))
for i in range(1, 11):
      print(f"{num} x {i} = {num * i}")
```

4.1 write a program to define a function with multiple return values

```
def calculate(a, b):
    sum_value = a + b
    diff_value = a - b
    product_value = a * b
    return sum_value, diff_value, product_value

x, y, z = calculate(10, 5)
print("Sum:", x)
print("Difference:", y)
print("Product:", z)
```

4.2 write a program to define a function with multiple return statements

```
def check_number(num):
    if num > 0:
        return "Positive"
    elif num < 0:
        return "Negative"
    else:
        return "Zero"

result = check_number(10)
print(result)

result = check_number(-5)
print(result)

result = check_number(0)
print(result)
```

5.1 write program to define a function using default arguments

```
def greet(name="Guest", age=25):
    print(f"Hello {name}, you are {age} years old.")

greet()
greet(name="Alice")
greet(name="Bob", age=30)
```

5.2 write program to define a function using keyword arguments

```
def greet(name, age):
    print(f"Hello {name}, you are {age} years old.")

greet(name="Alice", age=30)
greet(age=25, name="Bob")
```

6. Write a program to perform the given operations on a list like addition, insertion, slicing, deleting an element from any given position, changing a value in the list

```
my_list = [10, 20, 30, 40, 50]

my_list.append(60)
print("After addition:", my_list)

my_list.insert(2, 25)
print("After insertion:", my_list)

sliced_list = my_list[1:5]
print("Sliced list:", sliced_list)

del my_list[3]
print("After deletion:", my_list)

my_list[2] = 35
print("After changing a value:", my_list)
```

7.1 write a program to check if a given key exists in a dictionary or not

```
my_dict = {'name': 'Alice', 'age': 25, 'city': 'New York'}

key_to_check = 'age'

if key_to_check in my_dict:
    print(f"'{key_to_check}' exists in the dictionary.")
else:
    print(f"'{key_to_check}' does not exist in the dictionary.")
```

7.2 write a program to add a new key-value pair to an existing dictionary

```
my_dict = {'name': 'Alice', 'age': 25, 'city': 'New York'}

my_dict['country'] = 'USA'

print("Updated dictionary:", my_dict)
```

8.Write a python program to create a class that represents a shape. include
Methods to calculate its area and perimeter. Implement sub-classes for different shapes like circle, triangle and square.

```
import math

class Shape:
    def area(self):
        pass

    def perimeter(self):
        pass

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return math.pi * self.radius ** 2

    def perimeter(self):
        return 2 * math.pi * self.radius

class Triangle(Shape):
    def __init__(self, side1, side2, side3):
        self.side1 = side1
        self.side2 = side2
        self.side3 = side3

    def area(self):
        s = (self.side1 + self.side2 + self.side3) / 2
        return math.sqrt(s * (s - self.side1) * (s - self.side2) * (s - self.side3))

    def perimeter(self):
        return self.side1 + self.side2 + self.side3

class Square(Shape):
    def __init__(self, side):
        self.side = side

    def area(self):
        return self.side ** 2

    def perimeter(self):
        return 4 * self.side

circle = Circle(5)
triangle = Triangle(3, 4, 5)
square = Square(4)

print(f"Circle - Area: {circle.area()}, Perimeter: {circle.perimeter()}")
print(f"Triangle - Area: {triangle.area()}, Perimeter: {triangle.perimeter()}")
print(f"Square - Area: {square.area()}, Perimeter: {square.perimeter()}")
```

9. Python program to demonstrate Numpy arrays creation using array() function. And use of ndim, shape, size, d type.

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])

print("Array:", arr)
print("Number of dimensions (ndim):", arr.ndim)
print("Shape of the array:", arr.shape)
print("Size of the array:", arr.size)
print("Data type of the array elements (dtype):", arr.dtype)
```

10. Create a dictionary with at least five keys and each key represent value as a list where this list contains at least ten values and convert this dictionary as a pandas dataframe and explore the data through the data frame as follows:
a) Apply head() function to the pandas data frame
b) Perform various data selection operations on Data Frame

```
import pandas as pd

data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva', 'Frank', 'Grace', 'Helen', 'Ivan', 'Jack'],
    'Age': [24, 30, 22, 28, 25, 32, 21, 27, 29, 26],
    'City': ['New York', 'Los Angeles', 'Chicago', 'Houston', 'Phoenix', 'Philadelphia', 'San Antonio', 'San Diego', 'Dallas', 'Austin'],
    'Salary': [50000, 60000, 55000, 62000, 45000, 75000, 68000, 72000, 71000, 54000],
    'Department': ['HR', 'IT', 'Finance', 'Sales', 'Marketing', 'HR', 'Sales', 'IT', 'Finance', 'HR']
}

df = pd.DataFrame(data)

print("First 5 rows of the DataFrame:")
print(df.head())

print("\n'Name' column:")
print(df['Name'])

print("\n'Name' and 'Salary' columns:")
print(df[['Name', 'Salary']])

print("\nRows from index 2 to 5:")
print(df[2:6])

print("\nRows where Age > 25:")
print(df[df['Age'] > 25])

print("\nRow at index 3:")
print(df.iloc[3])
```