***Write a program to take two numbers from the user and perform Addition, Subtraction, Multiplication and print all the results.***

## Source Code:

```
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
addition = num1 + num2
subtraction = num1 - num2
multiplication = num1 * num2
print("Addition:", addition)
print("Subtraction:", subtraction)
print("Multiplication:", multiplication)
```

## Output:

```
Enter first number: 5
Enter second number: 3
Addition: 8.0
Subtraction: 2.0
Multiplication: 15.0
```



***Write a program to input a string from the user and display its first character, last character, first three characters and last three characters.***

## Source Code:

```
text = input("Enter a string: ")
print("First character:", text[0])
print("Last character:", text[-1])
print("First three characters:", text[:3])
print("Last three characters:", text[-3:])
```

## Output:

```
Enter a string: Ashesh
First character: A
Last character: h
First three characters: Ash
Last three characters: esh
```



## *Write a program to find a character in the string.*

### Source Code:

```
text = input("Enter a string: ")
char = input("Enter a character to find: ")
if char in text:
    print("Character found in the string.")
else:
    print("Character not found in the string.")
```

### Output:

```
Enter a string: Ashesh
Enter a character to find: y
Character not found in the string.
```



***Write a program to create a list that contains string, boolean value, float, integer and a list of numbers using list constructor and also using [ ] brackets. Access the item at index 2 and append [2, 6] to the list.***

## Source Code:

```
# Creating list using list constructor
list1 = list(["Hello", True, 3.14, 10, [1, 2, 3]])
# Creating list using square brackets
list2 = ["World", False, 2.71, 20, [4, 5, 6]]
# Access item at index 2
print("From list1, item at index 2:", list1[2])
print("From list2, item at index 2:", list2[2])
# Append [2, 6] to both lists
list1.append([2, 6])
list2.append([2, 6])
# Display updated lists
print("Updated list1:", list1)
print("Updated list2:", list2)
```

## Output:

```
From list1, item at index 2: 3.14
From list2, item at index 2: 2.71
Updated list1: ['Hello', True, 3.14, 10, [1, 2, 3], [2, 6]]
Updated list2: ['World', False, 2.71, 20, [4, 5, 6], [2, 6]]
```



## *Write a program to create a dictionary with keys and values of different data types. Access keys, values and items of the dictionary.*

### Source Code:

```
# Create a dictionary with different data types
my_dict = {
    "name": "Ashesh",       # string
    "age": 21,              # integer
    "height": 6.2,          # float
    "is_student": True,     # boolean
    "marks": [80, 90, 85]   # list
}
# Access keys, values, and items
print("Keys:", my_dict.keys())
print("Values:", my_dict.values())
print("Items:", my_dict.items())
```

### Output:

```
Keys: dict_keys(['name', 'age', 'height', 'is_student', 'marks'])
Values: dict_values(['Ashesh', 21, 6.2, True, [80, 90, 85]])
Items: dict_items([('name', 'Ashesh'), ('age', 21), ('height', 6.2), ('is_student', True), ('marks', [80, 90, 85])])
```



***Write a program that accepts marks in five subjects. (Useif - else)***

- ***If any subject has marks less than 40 → Fail***
- ***Else calculate average and display:***
  - ***≥ 75 → Distinction***
  - ***≥ 60 → First Division***
  - ***≥ 50 → Second Division***
  - ***Else → Pass***

**Source Code:**

```
m1 = float(input("Enter marks for subject 1: "))
m2 = float(input("Enter marks for subject 2: "))
m3 = float(input("Enter marks for subject 3: "))
m4 = float(input("Enter marks for subject 4: "))
m5 = float(input("Enter marks for subject 5: "))
if (m1 < 40 or m2 < 40 or m3 < 40 or m4 < 40 or m5 < 40):
    print("Result: Fail")
else:
    avg = (m1 + m2 + m3 + m4 + m5) / 5
    print("Average marks:", avg)
    if avg >= 75:
        print("Division: Distinction")
    elif avg >= 60:
        print("Division: First Division")
    elif avg >= 50:
        print("Division: Second Division")
    else:
        print("Division: Pass")
```



## Output:

```
Enter marks for subject 1: 50
Enter marks for subject 2: 70
Enter marks for subject 3: 75
Enter marks for subject 4: 82
Enter marks for subject 5: 90
Average marks: 73.4
Division: First Division
```



## *Write a program to check whether a number is divisible by 2 and 3 both.*

### Source Code:

```
num = int(input("Enter a number: "))
if num % 2 == 0 and num % 3 == 0:
    print("The number is divisible by both 2 and 3.")
else:
    print("The number is not divisible by both 2 and 3.")
```

### Output:

```
Enter a number: 6
The number is divisible by both 2 and 3.
```



***Create a calculator using match-case that:***

- ***Accepts two numbers***
- ***Allows the user to choose an operation (+, -,*, /, %)***
- ***Displays appropriate error message for invalid choice or division by zero.***

**Source Code:**

```
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
op = input("Enter operation (+, -, *, /, %): ")
match op:
    case "+":
        print("Result:", num1 + num2)
    case "-":
        print("Result:", num1 - num2)
    case "*":
        print("Result:", num1 * num2)
    case "/":
        if num2 == 0:
            print("Error: Division by zero!")
        else:
            print("Result:", num1 / num2)
    case "%":
        if num2 == 0:
            print("Error: Modulus by zero!")
        else:
            print("Result:", num1 % num2)
    case _:
        print("Error: Invalid operation!")
```



## Output:

```
Enter first number: 5
Enter second number: 2
Enter operation (+, -, *, /, %): +
Result: 7.0
```



***Write a program that accepts a single character and checks whether it is: (Use if-elif-else)***

- ***Uppercase letter***
- ***Lowercase letter***
- ***Digit***
- ***Special character***

**Source Code:**

```
ch = input("Enter a single character: ")
if ch.isupper():
    print("Uppercase letter")
elif ch.islower():
    print("Lowercase letter")
elif ch.isdigit():
    print("Digit")
else:
    print("Special character")
```

**Output:**

```
Enter a single character: @
Special character
```



***Write a program to print the following pattern:***

**Source Code:**

```
rows = 5
for i in range(1, rows + 1):
    print(" " * (rows - i), end="")
    print("* " * i)
```

***Write a program to print the following pattern:***

**N**
**NE**
**NEP**
**NEPA**
**NEPAL**

**Source Code:**

```
word = "NEPAL"

for i in range(1, len(word) + 1):

    print(word[:i])
```



***Write a program to print the following pattern:***

**1**
**2 3**
**4 5 6**
**7 8 9 10**

**Source Code:**

```
num = 1
rows = 4
for i in range(1, rows + 1):
    for j in range(i):
        print(num, end=" ")
        num += 1
    print()
```



## *Write a program to print the sum of all digits of user input numbers.*

### Source Code:

```
num_str = input("Enter a number: ")
total = 0
for digit in num_str:
    if digit.isdigit():
        total += int(digit)
print("Sum of digits:", total)
```

### Output:

```
Enter a number: 123
Sum of digits: 6
```



## *Write a program to count even numbers and odd numbers stored in a list.*

### Source Code:

```
numbers = [10, 15, 22, 33, 40, 55, 60]
even_count = 0
odd_count = 0
for num in numbers:
    if num % 2 == 0:
        even_count += 1
    else:
        odd_count += 1
print("Even numbers count:", even_count)
print("Odd numbers count:", odd_count)
```

### Output:

```
Even numbers count: 4
Odd numbers count: 3
```



## *Write a program to print factorials of a number using a for loop.*

### Source Code:

```
num = int(input("Enter a number: "))
factorial = 1
for i in range(1, num + 1):
    factorial *= i
print(f"Factorial of {num} is {factorial}")
```

### Output:

```
Enter a number: 5
Factorial of 5 is 120
```



## *Write a program to find the sum and average of 6 numbers using for loop.*

### Source Code:

```
total = 0
for i in range(1, 7):
    num = float(input(f"Enter number {i}: "))
    total += num
average = total / 6
print("Sum of numbers:", total)
print("Average of numbers:", average)
```

### Output:

```
Enter number 1: 1
Enter number 2: 2
Enter number 3: 3
Enter number 4: 4
Enter number 5: 5
Enter number 6: 6
Sum of numbers: 21.0
Average of numbers: 3.5
```



***Write a program of while loop that keeps asking the user to enter a number until they enter 0.***

## Source Code:

```
num = None
while num != 0:
    num = int(input("Enter a number (0 to quit): "))
    if num != 0:
        print("You entered:", num)
print("Program Ended !")
```

## Output:

```
Enter a number (0 to quit): 1
You entered: 1
Enter a number (0 to quit): 2
You entered: 2
Enter a number (0 to quit): 0
Program Ended !
```



## *Write a program to find the area of a circle using the lambda function.*

### Source Code:

```
area = lambda r: 3.14159 * r * r
radius = float(input("Enter the radius of the circle: "))
result = area(radius)
print("Area of the circle is:", result)
```

### Output:

```
Enter the radius of the circle: 5
Area of the circle is: 78.53975
```



## *Write a program to define a function to find the sum of elements at an even index in a list.*

### Source Code:

```
def sum_even_index(lst):
    total = 0
    for i in range(0, len(lst), 2):
        total = total + lst[i]
    return total
numbers = [1, 2, 3, 4, 5, 6]
result = sum_even_index(numbers)
print("Sum of elements at even indices is:", result)
```

### Output:

```
Sum of elements at even indices is: 9
```



## *Write a program to define a function that accepts marks of 5 subjects and returns percentage and grade.*

### Source Code:

```
def result(m1, m2, m3, m4, m5):
    total = m1 + m2 + m3 + m4 + m5
    percentage = total / 5
    if percentage >= 80:
        grade = "A"
    elif percentage >= 60:
        grade = "B"
    elif percentage >= 50:
        grade = "C"
    else:
        grade = "Fail"
    return percentage, grade
m1 = int(input("Enter marks of subject 1: "))
m2 = int(input("Enter marks of subject 2: "))
m3 = int(input("Enter marks of subject 3: "))
m4 = int(input("Enter marks of subject 4: "))
m5 = int(input("Enter marks of subject 5: "))
per, grade = result(m1, m2, m3, m4, m5)
print("Percentage:", per)
print("Grade:", grade)
```

### Output:

```
Enter marks of subject 1: 90
Enter marks of subject 2: 88
Enter marks of subject 3: 83
Enter marks of subject 4: 86
Enter marks of subject 5: 82
Percentage: 85.8
Grade: A
```

## *Write a program to define a function to append squares of numbers from 1 to 10 into a list.*

### Source Code:

```
def square_list():
    lst = []
    for i in range(1, 11):
        lst.append(i * i)
    return lst
result = square_list()
print("List of squares:", result)
```

### Output:

```
List of squares: [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```



## *Write a program to define a function to count vowels in a string.*

### Source Code:

```
def count_vowels(s):
    count = 0
    for char in s:
        if char in 'aeiouAEIOU':
            count = count + 1
    return count
text = input("Enter a string: ")
vowel_count = count_vowels(text)
print("Number of vowels in the string:", vowel_count)
```

### Output:

```
Enter a string: AsheshNeupane
Number of vowels in the string: 6
```



***Write a program to define a function to remove all the spaces from a string taken from the user.***

## Source Code:

```
def remove_spaces(s):
    result = ""
    for char in s:
        if char != " ":
            result = result + char
    return result
text = input("Enter a string: ")
new_text = remove_spaces(text)
print("String without spaces:", new_text)
```

## Output:

```
Enter a string: Ashesh Neupane
String without spaces: AsheshNeupane
```



## *Write a program to define a function to find common elements in two lists.*

### Source Code:

```
def common_elements(list1, list2):
    result = []
    for item in list1:
        if item in list2 and item not in result:
            result = result + [item]
    return result
list1 = [1, 2, 3, 4, 5]
list2 = [4, 5, 6, 7, 8]
common = common_elements(list1, list2)
print("Common elements:", common)
```

### Output:

```
Common elements: [4, 5]
```



## *Write a program to define a function that checks whether a list is sorted in ascending order.*

### Source Code:

```
def is_sorted(lst):
    for i in range(len(lst) - 1):
        if lst[i] > lst[i + 1]:
            return False
    return True
numbers = [1, 2, 3, 4, 5]
result = is_sorted(numbers)
if result:
    print("The list is sorted in ascending order.")
else:
    print("The list is not sorted in ascending order.")
```

### Output:

```
The list is sorted in ascending order.
```



## *Write a program to define a function that keeps asking the user for numbers and appends them until the user enters 0.*

### Source Code:

```
def collect_numbers():
    numbers = []
    while True:
        n = int(input("Enter a number (0 to stop): "))
        if n == 0:
            break
        numbers = numbers + [n]
    return numbers
result = collect_numbers()
print("Numbers entered:", result)
```

### Output:

```
Enter a number (0 to stop): 5
Enter a number (0 to stop): 4
Enter a number (0 to stop): 0
Numbers entered: [5, 4]
```



## *Write a program to define a function to remove all even numbers from a given list.*

### Source Code:

```
def remove_even(lst):
    result = []
    for num in lst:
        if num % 2 != 0:
            result = result + [num]
    return result
numbers = [1, 2, 3, 4, 5, 6, 7, 8]
new_list = remove_even(numbers)
print("List after removing even numbers:", new_list)
```

### Output:

```
List after removing even numbers: [1, 3, 5, 7]
```



***Create a 1D NumPy array containing numbers from 1 to 12.***

- ***Reshape the array into a 3×4 matrix and display it.***
- ***Reshape the same array into a 4×3 matrix and display it.***

**Source Code:**

```
import numpy as np
# Create a 1D array from 1 to 12
arr = np.arange(1, 13)
print("1D Array:")
print(arr)
# Reshape into 3x4 matrix
matrix_3x4 = arr.reshape(3, 4)
print("\n3x4 Matrix:")
print(matrix_3x4)
# Reshape into 4x3 matrix
matrix_4x3 = arr.reshape(4, 3)
print("\n4x3 Matrix:")
print(matrix_4x3)
```

**Output:**

```
1D Array:
[ 1  2  3  4  5  6  7  8  9 10 11 12]


3x4 Matrix:
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]

4x3 Matrix:
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
```

```
[10 11 12]]
```



## *Write a program to generate random integers using numpy and find sum & average of the array.*

### Source Code:

```
import numpy as np

# Generate an array of random integers

# Parameters: low, high, size

arr = np.random.randint(1, 100, size=10)

print("Generated Array:", arr)

# Calculate sum

total_sum = np.sum(arr)

print("Sum of array:", total_sum)

# Calculate average

average = np.mean(arr)

print("Average of array:", average)
```

### Output:

```
Generated Array: [46 11 37 87 99 51  4 47 49 53]

Sum of array: 484

Average of array: 48.4
```



## *Write a program to count the number of elements greater than the mean of the array.*

### Source Code:

```
import numpy as np
arr = np.array([10, 20, 30, 40, 50])
print("Array:", arr)
# Calculate mean
mean_value = np.mean(arr)
print("Mean:", mean_value)
# Count elements greater than mean
count = np.sum(arr > mean_value)
print("Number of elements greater than mean:", count)
```

### Output:

```
Array: [10 20 30 40 50]
Mean: 30.0
Number of elements greater than mean: 2
```



## *Write a function that accepts a NumPy array and counts how many even and odd numbers are present.*

### Source Code:

```
import numpy as np
def count_even_odd(arr):
    """
    Counts the number of even and odd elements in a NumPy array.
    Parameters:
    arr (numpy.ndarray): Input array of integers.
    Returns:
    tuple: A tuple containing two integers:
        (even_count, odd_count)
    """
    even_count = np.sum(arr % 2 == 0)
    odd_count = np.sum(arr % 2 != 0)
    return even_count, odd_count
# Example usage
arr = np.array([1, 2, 3, 4, 5, 6])
even, odd = count_even_odd(arr)
print("Array:", arr)
print("Even numbers:", even)
print("Odd numbers:", odd)
```

### Output:

```
Array: [1 2 3 4 5 6]
Even numbers: 3
Odd numbers: 3
```



## *Write a program to extract all the even numbers from a NumPy array.*

### Source Code:

```
import numpy as np

arr = np.array([10, 15, 22, 33, 40, 55, 60])

print("Original Array:", arr)

even_numbers = arr[arr % 2 == 0]

print("Even Numbers:", even_numbers)
```

### Output:

```
Original Array: [10 15 22 33 40 55 60]

Even Numbers: [10 22 40 60]
```



## *Write a program to create a class Rectangle with methods to calculate area and perimeter.*

### Source Code:

```
class Rectangle:
    """
    This class represents a rectangle.
    It provides methods to calculate area and perimeter.
    """
    def __init__(self, length, width):
        """
        Constructor to initialize length and width
        """
        self.length = length
        self.width = width
    def area(self):
        """
        Method to calculate area of rectangle
        Formula: length * width
        """
        return self.length * self.width
    def perimeter(self):
        """
        Method to calculate perimeter of rectangle
        Formula: 2 * (length + width)
        """
        return 2 * (self.length + self.width)
# Create object
r = Rectangle(10, 5)
# Display results
print("Area:", r.area())
print("Perimeter:", r.perimeter())
```

## Output:

```
Area: 50
Perimeter: 30
```



***Write a program to create a class Student with attributes: name, roll_no, and marks (list of subjects). Write methods to perform following tasks :***

***i. Calculate average marks.***

***ii. Assign grades based on average.***

***iii. Display full student details.***

***iv. Create at least 3 student objects and display their results.***

**Source Code:**

```
class Student:
    # Constructor to initialize student details
    def __init__(self, name, roll_no, marks):
        self.name = name
        self.roll_no = roll_no
        self.marks = marks    # list of marks
    # i. Calculate average marks
    def calculate_average(self):
        return sum(self.marks) / len(self.marks)
    # ii. Assign grade based on average
    def assign_grade(self):
        avg = self.calculate_average()
        if avg >= 80:
            return "A"
        elif avg >= 60:
            return "B"
        elif avg >= 50:
            return "C"
        else:
            return "Fail"
    # iii. Display student details
    def display(self):
        print("\nName:", self.name)
```

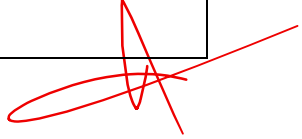
```
        print("Roll No:", self.roll_no)
        print("Marks:", self.marks)
        print("Average:", self.calculate_average())
        print("Grade:", self.assign_grade())
# iv. Create at least 3 student objects
s1 = Student("Ashesh", 1, [80, 85, 78])
s2 = Student("Bishow", 2, [60, 65, 70])
s3 = Student("Nirajan", 3, [45, 50, 48])
# Display results
s1.display()
s2.display()
s3.display()
```

## Output:

```
Name: Ashesh
Roll No: 1
Marks: [80, 85, 78]
Average: 81.0
Grade: A

Name: Bishow
Roll No: 2
Marks: [60, 65, 70]
Average: 65.0
Grade: B

Name: Nirajan
Roll No: 3
Marks: [45, 50, 48]
Average: 47.666666666666664
Grade: Fail
```



***Write a program to create a base class Person with attributes name and age, then derive a class Teacher that adds subject attribute, overrides the display_info() method, and uses super() to call the parent class method.***

## Source Code:

```
# Base class
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def display_info(self):
        print("Name:", self.name)
        print("Age:", self.age)
# Derived class
class Teacher(Person):
    def __init__(self, name, age, subject):
        super().__init__(name, age)   # Calling parent constructor
        self.subject = subject
    def display_info(self):
        super().display_info()        # Calling parent method
        print("Subject:", self.subject)
# Creating object
t = Teacher("Hari Bahadur Baniya", 50, "Mathematics")
# Calling method
t.display_info()
```

## Output:

```
Name: Hari Bahadur Baniya
Age: 50
Subject: Mathematics
```

***Write a program to create a class Academic with study-related methods and a class Sports with sports-related methods, then create a class Student that inherits from both and demonstrates combined behavior by accessing methods of both parent classes.***

## Source Code:

```
class Academic:
    def study(self):
        print("Student is studying subjects.")
class Sports:
    def play(self):
        print("Student is playing sports.")
class Student(Academic, Sports):
    def show(self):
        print("Student performs both academic and sports activities.")
# Creating object
s1 = Student()
# Accessing methods from both parent classes
s1.study()
s1.play()
s1.show()           .
```

## Output:

```
Student is studying subjects.
Student is playing sports.
Student performs both academic and sports activities.
```

***Write a program to create a class Result that stores marks of multiple subjects, calculates the percentage, and determines pass/fail along with division or class based on the percentage.***

**Source Code:**

```
class Result:
    def __init__(self, marks):
        self.marks = marks
        self.total_subjects = len(marks)
        self.total_marks = sum(marks)
        self.percentage = 0
    def calculate_percentage(self):
        if self.total_subjects == 0:
            self.percentage = 0
        else:
            self.percentage = self.total_marks / self.total_subjects
        return self.percentage
    def get_result(self):
        # Check fail condition (pass mark = 40)
        for mark in self.marks:
            if mark < 40:
                return "Fail"
        # Division based on percentage
        if self.percentage >= 80:
            return "Distinction"
        elif self.percentage >= 60:
            return "First Division"
        elif self.percentage >= 50:
            return "Second Division"
        elif self.percentage >= 40:
            return "Third Division"
        else:
            return "Fail"
```

```
    def display(self):
        self.calculate_percentage()
        result = self.get_result()
        print("Marks:", self.marks)
        print("Total Marks:", self.total_marks)
        print("Percentage:", self.percentage)
        print("Result:", result)
marks = [78, 85, 67, 90, 72]
student = Result(marks)
student.display()
```

## Output:

```
Marks: [78, 85, 67, 90, 72]
Total Marks: 392
Percentage: 78.4
Result: First Division
```



***Write a program to create a base class Shape with a method area(), then derive classes Circle and Rectangle that override the area() method to calculate their respective areas.***

## Source Code:

```
import math
# Base class
class Shape:
    def area(self):
        print("Area method should be overridden in subclasses")
# Derived class: Circle
class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius
    def area(self):
        return math.pi * self.radius ** 2
# Derived class: Rectangle
class Rectangle(Shape):
    def __init__(self, length, width):
        self.length = length
        self.width = width
    def area(self):
        return self.length * self.width
circle = Circle(5)
rectangle = Rectangle(4, 6)
print("Area of Circle:", circle.area())
print("Area of Rectangle:", rectangle.area())
```

## Output:

```
Area of Circle: 78.53981633974483
Area of Rectangle: 24
```

***Write a program to create an abstract class Vehicle with an abstract method start(). Derive classes Car and Bike and implement start() differently in each class.***

## Source Code:

```
from abc import ABC, abstractmethod
# Abstract class
class Vehicle(ABC):
    @abstractmethod
    def start(self):
        pass
# Car class
class Car(Vehicle):
    def start(self):
        print("Car starts with key")
# Bike class
class Bike(Vehicle):
    def start(self):
        print("Bike starts with kick")
c = Car()
b = Bike()
c.start()
b.start()
```

## Output:

```
Car starts with key
Bike starts with kick
```

## *Write a program to create a class Employee with name and salary, then derive Manager using super() to add department and display details.*

### Source Code:

```
# Base class
class Employee:
    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
# Derived class
class Manager(Employee):
    def __init__(self, name, salary, department):
        super().__init__(name, salary)
        self.department = department
    def display(self):
        print("Name:", self.name)
        print("Salary:", self.salary)
        print("Department:", self.department)
m = Manager("Ashesh", 50000, "IT")
m.display()
```

### Output:

```
Name: Ashesh
Salary: 50000
Department: IT
```



***Write a program to create an abstract class Transport with a method calculate_fare(distance), then implement classes Bus, Train, and Taxi that override the method to calculate fare differently.***

## Source Code:

```
from abc import ABC, abstractmethod
# Abstract class
class Transport(ABC):
    @abstractmethod
    def calculate_fare(self, distance):
        pass
# Bus class
class Bus(Transport):
    def calculate_fare(self, distance):
        return distance * 5   # Rs. 5 per km
# Train class
class Train(Transport):
    def calculate_fare(self, distance):
        return distance * 3   # Rs. 3 per km
# Taxi class
class Taxi(Transport):
    def calculate_fare(self, distance):
        return distance * 10  # Rs. 10 per km
d = 10  # distance in km
b = Bus()
t = Train()
tx = Taxi()
print("Bus fare:", b.calculate_fare(d))
print("Train fare:", t.calculate_fare(d))
print("Taxi fare:", tx.calculate_fare(d))
```

## Output:

```
Bus fare: 50
Train fare: 30
Taxi fare: 100
```



***Use sqlite3 module in python and follow the given instructions :***

1. ***Create a SQLite database called students.db***
2. ***Create two tables: students and courses with attributes student_id, first_name, last_name, email and enrollment_date whereas courses table has attributes course_id, course_name, instructor and credits.***

**Source Code:**

```
import sqlite3
# Create SQLite database
conn = sqlite3.connect("students.db")
cursor = conn.cursor()
# Create students table with proper constraints
cursor.execute("""
CREATE TABLE students (
    student_id INTEGER PRIMARY KEY,
    first_name TEXT NOT NULL,
    last_name TEXT NOT NULL,
    email TEXT UNIQUE NOT NULL,
    enrollment_date TEXT NOT NULL
)
""")
# Create courses table with proper constraints
cursor.execute("""
CREATE TABLE courses (
    course_id INTEGER PRIMARY KEY,
    course_name TEXT NOT NULL UNIQUE,
    instructor TEXT NOT NULL,
    credits INTEGER NOT NULL CHECK (credits > 0)
)
""")
conn.commit()
conn.close()
```

```
print("Database and tables created successfully.")
```

## Output:

```
Database and tables created successfully.
```



## *Perform basic INSERT operations and handle multiple data insertions.*

## *1. Insert at least 5 student records*

## *2. Insert at least 3 course records*

### Source Code:

```
import sqlite3
# Connect database
conn = sqlite3.connect("students.db")
cursor = conn.cursor()
# 1. Insert 5 student records
students_data = [
    (1, "Ashesh", "Neupane", "an@gmail.com", "2025-01-10"),
    (2, "Bishow", "Magar", "bm@gmail.com", "2025-02-15"),
    (3, "Sanyam", "Kothari", "sk@gmail.com", "2025-03-20"),
    (4, "Nishant", "Lamichhane", "nl@gmail.com", "2025-04-05"),
    (5, "Abhishek", "Karki", "ak@gmail.com", "2025-05-01")
]
cursor.executemany("""
INSERT INTO students (student_id, first_name, last_name, email, enrollment_date)
VALUES (?, ?, ?, ?, ?)
""", students_data)
# 2. Insert 3 course records
courses_data = [
    (101, "Python", "Nabin", 4),
    (102, "Database", "Anil", 3),
    (103, "Web Development", "Manoj", 4)
]
cursor.executemany("""
INSERT INTO courses (course_id, course_name, instructor, credits)
VALUES (?, ?, ?, ?)
""", courses_data)
conn.commit()
```

```
# Display inserted data
print("Students Table:")
for row in cursor.execute("SELECT * FROM students"):
    print(row)
print("\nCourses Table:")
for row in cursor.execute("SELECT * FROM courses"):
    print(row)
conn.close()
```

## Output:

```
Students Table:
(1, 'Ashesh', 'Neupane', 'an@gmail.com', '2025-01-10')
(2, 'Bishow', 'Magar', 'bm@gmail.com', '2025-02-15')
(3, 'Sanyam', 'Kothari', 'sk@gmail.com', '2025-03-20')
(4, 'Nishant', 'Lamichhane', 'nl@gmail.com', '2025-04-05')
(5, 'Abhishek', 'Karki', 'ak@gmail.com', '2025-05-01')

Courses Table:
(101, 'Python', 'Nabin', 4)
(102, 'Database', 'Anil', 3)
(103, 'Web Development', 'Manoj', 4)
```



## *Perform various SELECT queries with different clauses.*

## *1. Write queries to retrieve all students*

## *2. Select students enrolled after a specific date*

## *3. Use ORDER BY clause*

**Source Code:**

```
import sqlite3
# Connect database
conn = sqlite3.connect("students.db")
cursor = conn.cursor()
# 1. Retrieve all students
print("All Students:")
cursor.execute("SELECT * FROM students")
for row in cursor.fetchall():
    print(row)
# 2. Students enrolled after a specific date
print("\nStudents enrolled after 2025-02-01:")
cursor.execute("""
SELECT * FROM students
WHERE enrollment_date > '2025-02-01'
""")
for row in cursor.fetchall():
    print(row)
# 3. ORDER BY clause (sort by first_name)
print("\nStudents ordered by first name:")
cursor.execute("""
SELECT * FROM students
ORDER BY first_name ASC
""")
for row in cursor.fetchall():
    print(row)
```

```
conn.close()
```

## Output:

```
All Students:
(1, 'Ashesh', 'Neupane', 'an@gmail.com', '2025-01-10')
(2, 'Bishow', 'Magar', 'bm@gmail.com', '2025-02-15')
(3, 'Sanyam', 'Kothari', 'sk@gmail.com', '2025-03-20')
(4, 'Nishant', 'Lamichhane', 'nl@gmail.com', '2025-04-05')
(5, 'Abhishek', 'Karki', 'ak@gmail.com', '2025-05-01')

Students enrolled after 2025-02-01:
(2, 'Bishow', 'Magar', 'bm@gmail.com', '2025-02-15')
(3, 'Sanyam', 'Kothari', 'sk@gmail.com', '2025-03-20')
(4, 'Nishant', 'Lamichhane', 'nl@gmail.com', '2025-04-05')
(5, 'Abhishek', 'Karki', 'ak@gmail.com', '2025-05-01')

Students ordered by first name:
(5, 'Abhishek', 'Karki', 'ak@gmail.com', '2025-05-01')
(1, 'Ashesh', 'Neupane', 'an@gmail.com', '2025-01-10')
(2, 'Bishow', 'Magar', 'bm@gmail.com', '2025-02-15')
(4, 'Nishant', 'Lamichhane', 'nl@gmail.com', '2025-04-05')
(3, 'Sanyam', 'Kothari', 'sk@gmail.com', '2025-03-20')
```



## *Perform UPDATE and DELETE operations safely.*

## *1. Update a student's email address*

## *2. Update multiple records with a condition*

## *3. Delete a record and implement precautions.*

**Source Code:**

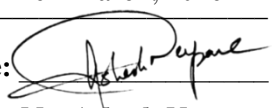
```
import sqlite3
# Connect database
conn = sqlite3.connect("students.db")
cursor = conn.cursor()
# 1. Update a student's email address
cursor.execute("""
UPDATE students
SET email = 'asheshneupane@gmail.com'
WHERE student_id = 1
""")
# 2. Update multiple records with a condition
cursor.execute("""
UPDATE students
SET enrollment_date = '2025-06-01'
WHERE enrollment_date < '2025-03-01'
""")
# 3. Delete a record safely (using condition)
cursor.execute("""
DELETE FROM students
WHERE student_id = 5
""")
conn.commit()
# Display updated table
print("Updated Students Table:")
cursor.execute("SELECT * FROM students")
for row in cursor.fetchall():
```

```
    print(row)
conn.close()
```

## Output:

```
Updated Students Table:
(1, 'Ashesh', 'Neupane', 'asheshneupane@gmail.com', '2025-06-01')
(2, 'Bishow', 'Magar', 'bm@gmail.com', '2025-06-01')
(3, 'Sanyam', 'Kothari', 'sk@gmail.com', '2025-03-20')
(4, 'Nishant', 'Lamichhane', 'nl@gmail.com', '2025-04-05')
```

**Date of Issue:** 26th March, 2026

**Signature:** 

**Issued By:** Mr. Ashesh Neupane

**Designation:** Founder / Admin