



ITM SKILLS
UNIVERSITY

**INSTITUTE OF TECHNOLOGY AND MANAGEMENT
SKILLS UNIVERSITY,
KHARGHAR, NAVI MUMBAI**

PYTHON PROGRAMMING LAB



Prepared by:

Name of Student: Aayush Ajit Chouunkar

Roll No: 04

Batch: 2023-27

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

exp. No	List of Experiment
	1.1 Write a program to compute Simple Interest.
	1.2 Write a program to perform arithmetic, Relational operators.
	1.3 Write a program to find whether a given no is even & odd.
	1.4 Write a program to print first n natural number & their sum.
	1.5 Write a program to determine whether the character entered is a Vowel or not .
	1.6 Write a program to find whether given number is an Armstrong Number.
	1.7 Write a program using for loop to calculate factorial of a No.
	1.8 Write a program to print the following pattern
	i) <pre> * * * * * * * * * * * * * * *</pre>
	ii) <pre> 1 2 2 3 3 3 4 4 4 4 5 5 5 5 5</pre>
	iii) <pre> * * * * *</pre>

	2.1 Write a program that define the list of defines the list of define countries that are in BRICS.
	2.2 Write a program to traverse a list in reverse order. 1.By using Reverse method. 2.By using slicing
	2.3 Write a program that scans the email address and forms a tuple of username and domain.
	2.4 Write a program to create a list of tuples from given list having number and add its cube in tuple. i/p: c= [2,3,4,5,6,7,8,9]
	2.5 Write a program to compare two dictionaries in Python? (By using == operator)
	2.6 Write a program that creates dictionary of cube of odd numbers in the range.
	2.7 Write a program for various list slicing operation. a= [10,20,30,40,50,60,70,80,90,100] i. Print Complete list ii. Print 4th element of list iii. Print list from 0th to 4th index. iv. Print list -7th to 3rd element v. Appending an element to list. vi. Sorting the element of list. vii. Popping an element. viii. Removing Specified element. ix. Entering an element at specified index. x. Counting the occurrence of a specified element. xi. Extending list. xii. Reversing the list.
	3.1 Write a program to extend a list in python by using given approach. i. By using + operator. ii. By using Append ()

	iii. By using extend ()
	3.2 Write a program to add two matrices.
	3.3 Write a Python function that takes a list and returns a new list with distinct elements from the first list.
	3.4 Write a program to Check whether a number is perfect or not.
	3.5 Write a Python function that accepts a string and counts the number of upper- and lower-case letters. string_test= 'Today is My Best Day'
	4.1 Write a program to Create Employee Class & add methods to get employee details & print.
	4.2 Write a program to take input as name, email & age from user using combination of keywords argument and positional arguments (*args and **kwargs) using function,
	4.3 Write a program to admit the students in the different Departments(pgdm/btech) and count the students. (Class, Object and Constructor).
	4.4 Write a program that has a class store which keeps the record of code and price of product display the menu of all product and prompt to enter the quantity of each item required and finally generate the bill and display the total amount.
	4.5 Write a program to take input from user for addition of two numbers using (single inheritance).
	4.6 Write a program to create two base classes LU and ITM and one derived class. (Multiple inheritance).
	4.7 Write a program to implement Multilevel inheritance, Grandfather□Father-□Child to show property inheritance from grandfather to child.
	4.8 Write a program Design the Library catalogue system using inheritance take base class (library item) and derived class (Book, DVD & Journal) Each derived

	class should have unique attribute and methods and system should support Check in and check out the system. (Using Inheritance and Method overriding)
	5.1 Write a program to create my_module for addition of two numbers and import it in main script.
	5.2 Write a program to create the Bank Module to perform the operations such as Check the Balance, withdraw and deposit the money in bank account and import the module in main file.
	5.3 Write a program to create a package with name cars and add different modules (such as BMW, AUDI, NISSAN) having classes and functionality and import them in main file cars.
	6.1 Write a program to implement Multithreading. Printing “Hello” with one thread & printing “Hi” with another thread.
	7.1 Write a program to use ‘whether API’ and print temperature of any city, also print the sunrise and sunset times for the same humidity of that area.
	7.2 Write a program to use the ‘API’ of crypto currency.

Name of Student: Aayush Ajit Choumkar

Roll Number: 04

Experiment No:1.1

Title: Write a program to compute Simple Interest.

Theory: Calculates the simple interest (SI) based on the given principle amount, rate of interest, and time period. Formula for SI = $P \times R \times T / 100$

Code:

```
p=int(input("Enter principal amount-"))
r=float(input("Enter rate of intrest-"))
t=int(input("Enter number of years-"))

simpleintrest=(p*r*t)/100

print("YOUR SIMPLE INTRESRT-",simpleintrest)
```

Output: (screenshot)

```
python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog1.1.py"
● aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoun
kar/ashpython/labmanauaipython/prog1.1.py"
Enter principal amount-1000
Enter rate of intrest-5
Enter number of years-2
YOUR SIMPLE INTRESRT- 100.0
○ aayushchoukar@Aayushs-MacBook-Air labmanauaipython % █
```

Test Case: Any two (screenshot)

```
python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog1.1.py"
● aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog1.1.py"
Enter principal amount-10000
Enter rate of intrest-2
Enter number of years-2
YOUR SIMPLE INTRESRT- 400.0
● aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog1.1.py"
Enter principal amount-0
Enter rate of intrest-2
Enter number of years-2
YOUR SIMPLE INTRESRT- 0.0
○ aayushchoukar@Aayushs-MacBook-Air labmanauaipython %
```

Conclusion:

Simple interest is calculated based on the values entered by the user with respect to the formula mentioned in the code.

Experiment 1.2

TITLE : Write a program to perform arithmetic, Relational operators.

THEORY:

Takes user input for two numbers x and y, and performs basic arithmetic operations on them like addition, subtraction, multiplication, division, modulus, and floor division

CODE:

```
# Arithmetic operators

def arithmetic_operations(a, b):

    print("Arithmetic Operations:")

    print(f"{a} + {b} = {a + b}")

    print(f"{a} - {b} = {a - b}")

    print(f"{a} * {b} = {a * b}")

    print(f"{a} / {b} = {a / b}")

    print(f"{a} % {b} = {a % b}")

    print(f"{a} // {b} = {a // b}")

    print(f"{a} ** {b} = {a ** b}")

# Relational operators

def relational_operations(a, b):

    print("\nRelational Operations:")

    print(f"{a} == {b}: {a == b}")

    print(f"{a} != {b}: {a != b}")

    print(f"{a} < {b}: {a < b}")

    print(f"{a} > {b}: {a > b}")

    print(f"{a} <= {b}: {a <= b}")
```



```

    print(f"{a} >= {b}: {a >= b}")

# Input values

num1 = int(input("Enter the first number: "))

num2 = int(input("Enter the second number: "))

# Perform arithmetic operations

arithmetic_operations(num1, num2)

# Perform relational operations

relational_operations(num1, num2)

```

OUTPUT:

```

aayushchoukar@Aayushs-MacBook-Air labmanaulpython % python3 -u "/Users/aayushchoukar/ashpython/labmanaulpython/prog1.2.py"
Enter the first number: 3
Enter the second number: 4
Arithmetic Operations:
3 + 4 = 7
3 - 4 = -1
3 * 4 = 12
3 / 4 = 0.75
3 % 4 = 3
3 // 4 = 0
3 ** 4 = 81

Relational Operations:
3 == 4: False
3 != 4: True
3 < 4: True
3 > 4: False
3 <= 4: True
3 >= 4: False

```

TESTCASE:

```

aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog1.2.py"
Enter the first number: 1
Enter the second number: 1
Arithmetic Operations:
1 + 1 = 2
1 - 1 = 0
1 * 1 = 1
1 / 1 = 1.0
1 % 1 = 0
1 // 1 = 1
1 ** 1 = 1

Relational Operations:
1 == 1: True
1 != 1: False
1 < 1: False
1 > 1: False
1 <= 1: True
1 >= 1: True
aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog1.2.py"
Enter the first number: 2
Enter the second number: 2
Arithmetic Operations:
2 + 2 = 4
2 - 2 = 0
2 * 2 = 4
2 / 2 = 1.0
2 % 2 = 0
2 // 2 = 1
2 ** 2 = 4

Relational Operations:
2 == 2: True
2 != 2: False
2 < 2: False
2 > 2: False
2 <= 2: True
2 >= 2: True
aayushchoukar@Aayushs-MacBook-Air labmanauaipython %

```

CONCLUSION:

On taking input from user, the code prints the answers of all arithmetic operations for the entered values with respect to the formulas mentioned in the code.

Experiment 1.3

TITLE: Write a program to find whether a given no is even & odd.

THEORY:

This program determines whether a given number is even or odd. It takes user input for a number, performs the modulo operation with 2, and checks if the result is equal to zero. If true, it concludes that the number is even; otherwise, it identifies the number as odd.

CODE:

```
nos=int(input("Enter a number="))

if nos%2==0:

    print(nos,"The number is even")

else:

    print(nos,"The number is odd")
```

OUTPUT:

```
python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog1.3.py"
aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog1.3.py"
Enter a number=2
2 The number is even
aayushchoukar@Aayushs-MacBook-Air labmanauaipython %
```

TESTCASE:

```
python3 -u "/Users/aayushchouunkar/ashpython/labmanauaipython/prog1.3.py"
● aayushchouunkar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoun
kar/ashpython/labmanauaipython/prog1.3.py"
Enter a number=2
2 The number is even
● aayushchouunkar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoun
kar/ashpython/labmanauaipython/prog1.3.py"
Enter a number=345
345 The number is odd
● aayushchouunkar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoun
kar/ashpython/labmanauaipython/prog1.3.py"
Enter a number=69
69 The number is odd
○ aayushchouunkar@Aayushs-MacBook-Air labmanauaipython % █
```

CONCLUSION

The code determines whether a user-inputted number is even or odd, with the use of the modulo operator to evaluate divisibility by 2.

Experiment1.4

TITLE:

Write a program to print first n natural number & their sum.

THEORY:

The program calculates the sum of natural numbers up to a given input 'n' using a while loop. It initialises variables for sum and 'i', continuously adds 'i' to the sum until 'i' reaches the value of 'n'.

CODE:

```
def print_natural_numbers_and_sum(n):  
  
    if n <= 0:  
  
        print("Please enter a positive integer.")  
  
        return  
  
    # Print the first n natural numbers  
  
    print(f"First {n} natural numbers:")  
  
    for i in range(1, n + 1):  
  
        print(i, end=" ")  
  
    # Calculate the sum of the first n natural numbers  
  
    sum_natural_numbers = n * (n + 1) // 2  
  
    # Print the sum  
  
    print(f"\nSum of the first {n} natural numbers: {sum_natural_numbers}")  
  
# Input value for n  
  
n = int(input("Enter a positive integer (n): "))
```

```
# Call the function to print natural numbers and their sum

print_natural_numbers_and_sum(n)
```

OUTPUT:

```
python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog1.4.py"
aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog1.4.py"
Enter a positive integer (n): 3
First 3 natural numbers:
1 2 3
Sum of the first 3 natural numbers: 6
aayushchoukar@Aayushs-MacBook-Air labmanauaipython %
```

TESTCASE:

```
python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog1.4.py"
aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog1.4.py"
Enter a positive integer (n): 5
First 5 natural numbers:
1 2 3 4 5
Sum of the first 5 natural numbers: 15
aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog1.4.py"
Enter a positive integer (n): 10
First 10 natural numbers:
1 2 3 4 5 6 7 8 9 10
Sum of the first 10 natural numbers: 55
aayushchoukar@Aayushs-MacBook-Air labmanauaipython %
```

CONCLUSION:

The program computes the sum of natural numbers up to the specified 'n' with respect to the formula mentioned in the code.

Experiment 1.5

TITLE:

Write a program to determine whether the character entered is a Vowel or not

THEORY:

This program determines whether a user-inputted character is a vowel or a consonant. It checks if character belongs to a predefined list of vowels, and then prints the answer.

CODE:

```
def is_vowel(char):  
    vowels = "aeiou AEIOU"  
  
    if char in vowels:  
        return True  
    else:  
        return False  
  
# Input a character  
character = input("Enter a character: ")  
  
# Check if the entered character is a vowel  
if len(character) == 1 and character.isalpha():  
    if is_vowel(character):  
        print(f"{character} is a vowel.")  
    else:  
        print(f"{character} is not a vowel.")  
else:  
    print("Please enter a valid single character.")
```

OUTPUT:

```
python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog1.5.py"
aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog1.5.py"
Enter a character: R
R is not a vowel.
aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog1.5.py"
Enter a character: E
E is a vowel.
aayushchoukar@Aayushs-MacBook-Air labmanauaipython %
```

TESTCASE:

```
python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog1.5.py"
aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog1.5.py"
Enter a character: AEIOU
Please enter a valid single character.
aayushchoukar@Aayushs-MacBook-Air labmanauaipython %
```

CONCLUSION:

This program is working accurately and takes a single character if the character entered is (a,e,i,o,u) or (A,E,I,O,U) it says it is vowel and it not matches its a consonant.

Experiment1.6

TITLE:

Write a program to find whether given number is an Armstrong Number.

THEORY:

This program takes user input, calculates the number of digits, and then computes sum of each digit raised to power of number of digits. It compares the calculated sum with the original number to check if it is an armstrong number or not

CODE:

```
def is_armstrong_number(number):  
  
    num_str = str(number)  
  
    num_digits = len(num_str)  
  
    sum_of_digits = sum(int(digit) ** num_digits for digit in num_str)  
  
    return sum_of_digits == number  
  
# Input a number  
num = int(input("Enter a number: "))  
  
# Check if the entered number is an Armstrong number  
if is_armstrong_number(num):  
    print(f"{num} is an Armstrong number.")  
else:  
    print(f"{num} is not an Armstrong number.")
```

OUTPUT:

```
python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog1.6.py"
● aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoun
kar/ashpython/labmanauaipython/prog1.6.py"
Enter a number: 9
9 is an Armstrong number.
○ aayushchoukar@Aayushs-MacBook-Air labmanauaipython % █
```

TESTCASE:

```
python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog1.6.py"
● aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoun
kar/ashpython/labmanauaipython/prog1.6.py"
Enter a number: 345
345 is not an Armstrong number.
● aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoun
kar/ashpython/labmanauaipython/prog1.6.py"
Enter a number: 6969
6969 is not an Armstrong number.
○ aayushchoukar@Aayushs-MacBook-Air labmanauaipython % █
```

CONCLUSION:

It defines the logic behind calculation of Armstrong number and identifies the same.

Experiment1.7

TITLE:

Write a program using for loop to calculate factorial of a No.

THEORY:

This code calculates the factorial of a given number using a for loop. It takes user input, initialises a variable factorial to 1, and then multiplies it by each integer from 1 to the input number.

CODE:

```
def calculate_factorial(n):  
    factorial = 1  
  
    # Check if the input is negative  
    if n < 0:  
        return "Factorial is not defined for negative numbers."  
  
    # Calculate factorial using a for loop  
    for i in range(1, n + 1):  
        factorial *= i  
  
    return factorial  
  
# Get user input for the number  
number = int(input("Enter a number: "))  
  
# Calculate and print the factorial  
result = calculate_factorial(number)  
print(f"The factorial of {number} is: {result}")
```

OUTPUT:

```
python3 -u "/Users/aayushchouunkar/ashpython/labmanauaipython/prog1.7.py"
aayushchouunkar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchouunkar/ashpython/labmanauaipython/prog1.7.py"
Enter a number: 6
The factorial of 6 is: 720
aayushchouunkar@Aayushs-MacBook-Air labmanauaipython %
```

TESTCASE:

```
python3 -u "/Users/aayushchouunkar/ashpython/labmanauaipython/prog1.7.py"
aayushchouunkar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchouunkar/ashpython/labmanauaipython/prog1.7.py"
Enter a number: 4
The factorial of 4 is: 24
aayushchouunkar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchouunkar/ashpython/labmanauaipython/prog1.7.py"
Enter a number: 10
The factorial of 10 is: 3628800
aayushchouunkar@Aayushs-MacBook-Air labmanauaipython %
```

CONCLUSION:

By using for loop, the program computes the factorial of the number with respect to the formula mentioned in the code.

Experiment1.8

TITLE:

PATTERN

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

THEORY:

This program generates a simple pattern of asterisks in the shape of a right-angled triangle. It prompts the user to input the number of rows, and then, using nested loops, prints asterisks in incremental order on each line.

CODE:

```
def print_pattern(rows):  
  
    for i in range(1, rows + 1):  
  
        for j in range(1, i + 1):  
  
            print("*", end=" ")  
  
        print()  
  
# Input the number of rows  
  
num_rows = int(input("Enter the number of rows: "))  
  
# Call the function to print the pattern  
  
print_pattern(num_rows)
```

OUTPUT:

```
python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog1.8.py"
aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog1.8.py"
Enter the number of rows: 10
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * *
* * * * * *
* * * * * *
* * * * * *
* * * * * *
aayushchoukar@Aayushs-MacBook-Air labmanauaipython %
```

CONCLUSION:

The program demonstrates a basic pattern printing technique using nested loops, creating a right-angled triangle with asterisks.

Experiment 1.9

TITLE:

PATTERN

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

THEORY:

This program generates a pattern where each row displays numbers incrementally up to the row number. It takes user input for the number of rows, and using nested loops, prints the row number repetitively on each line.

CODE:

```
def print_number_pattern(rows):

    for i in range(1, rows + 1):

        # Use nested loop to print repeated numbers

        for j in range(i):

            print(i, end=" ")

        print() # Move to the next line after printing each row

# Get user input for the number of rows

num_rows = int(input("Enter the number of rows for the pattern: "))

# Print the number pattern

print_number_pattern(num_rows)
```

OUTPUT:

```
python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog1.9.py"
● aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoun
kar/ashpython/labmanauaipython/prog1.9.py"
Enter the number of rows for the pattern: 5
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
○ aayushchoukar@Aayushs-MacBook-Air labmanauaipython %
```

CONCLUSION:

By nested for loop in def function,the program creates a simple numerical pattern, in right angle with the help of the concept of nested iteration.

Experiment1.9

TITLE:

PATTERN

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

THEORY:

This program generates a pattern of pyramid with asterisks. It takes user input for the number of rows, uses nested loops to control spacing and asterisk printing, creating the pyramid pattern.

CODE:

```
def print_pyramid(rows):

    for i in range(1, rows + 1):

        # Print spaces before the asterisks

        for j in range(rows - i):

            print(" ", end=" ")

        # Print asterisks

        for k in range(2 * i - 1):

            print("*", end=" ")

        # Move to the next line after printing each row

        print()

# Get user input for the number of rows
```

```

num_rows = int(input("Enter the number of rows for the pyramid: "))

# Print the pyramid pattern

print_pyramid(num_rows)

```

OUTPUT:

```

python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog1.10.py"
aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoun
kar/ashpython/labmanauaipython/prog1.10.py"
Enter the number of rows for the pyramid: 10
      *
     * * *
    * * * * *
   * * * * * * *
  * * * * * * * *
 * * * * * * * * *
* * * * * * * * * *
 * * * * * * * * * *
  * * * * * * * * * *
   * * * * * * * * * *
    * * * * * * * * * *
     * * * * * * * * *
      * * * * * * * *
aayushchoukar@Aayushs-MacBook-Air labmanauaipython %

```

CONCLUSION:

The code defines a `print_pyramid` function that takes user input for the number of rows and prints a pyramid pattern with spaces and asterisks. The pattern is formed by rows of increasing odd-numbered asterisks. The code effectively generates and displays a pyramid based on the user-specified number of rows.

Experiment 2.1

TITLE:

2.1 Write a program that define the list of defines the list of define countries that are in BRICS.

THEORY:

The program checks if a user inputted country is a member of BRICS GROUP .It uses list of brics countries and uses if else statement to display result.

CODE:

```
# Write a program that define the list of defines the list of define countries that are in
BRICS.

countries = ["brazil", "russia", "india", "china", "srilanka" ]

member = input("Enter the name of the country: ").lower()

if member in countries:

    print(member, " is the member of BRICS")

else:

    print(member, " is not a member of BRICS")
```

OUTPUT:

```
python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog2.1.py"
aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoun
kar/ashpython/labmanauaipython/prog2.1.py"
Enter the name of the country: India
india is the member of BRICS
aayushchoukar@Aayushs-MacBook-Air labmanauaipython %
```

CONCLUSION:

The program determines whether a given country is a member of BRICS, showcasing the use of lists, conditional statements, and user input processing

Experiment 2.2

TITLE:

2.2 Write a program to traverse a list in reverse order.

1.By using Reverse method.

2.By using slicing

THEORY:

The program takes user input to create a list of elements and then demonstrates two methods to traverse the list in reverse order: using the reversed method and by slicing.

CODE:

```
# Write a program to traverse a list in reverse order.

# 1.By using Reverse method.

# 2.By using slicing

mylist = input("Enter elements of the list: ").split()

print("List in reverse order using Reverse method:")

for item in reversed(mylist):

    print(item, end=" ")

print()
```

```
print("List in reverse order using Slicing:")  
  
for item in mylist[::-1]:  
    print(item, end=" ")
```

OUTPUT:

```
python3 -u "/Users/aayushchoukhar/ashpython/labmanaulpython/prog2.2.py"  
● aayushchoukhar@Aayushs-MacBook-Air labmanaulpython % python3 -u "/Users/aayushchoukhar/ashpython/labmanaulpython/prog2.2.py"  
Enter elements of the list: 1 2 3 4 5  
List in reverse order using Reverse method:  
5 4 3 2 1  
List in reverse order using Slicing:  
5 4 3 2 1 %  
○ aayushchoukhar@Aayushs-MacBook-Air labmanaulpython %
```

CONCLUSION:

By using both the reversed and slicing method the program showcases two approaches for traversing a list in reverse order.

Experiment2.3

TITLE:

Write a program that scans the email address and forms a tuple of username and domain.

THEORY:

This program takes user input of email addresses & then uses the split method to separate individual email address into username and domain parts using the '@' symbol.

CODE:

```
# Write a program that scans the email address and forms a tuple of username and domain.

emails = input("Enter the Email addresses separated by commas: ")
elist = emails.split(',')

for email in elist:
    username, domain = email.strip().split('@')
    print("Username: ", username, "Domain: ", domain)
```

OUTPUT:

```
python3 -u "/Users/aayushchoukar/ashpython/labmanaulpython/prog2.3.py"
● aayushchoukar@Aayushs-MacBook-Air labmanaulpython % python3 -u "/Users/aayushchoukar/ashpython/labmanaulpython/prog2.3.py"
Enter the Email addresses separated by commas: ayush123@gmail.com
Username: ayush123 Domain: gmail.com
○ aayushchoukar@Aayushs-MacBook-Air labmanaulpython % █
```

CONCLUSION:

The program extracts and prints the username and domain for each email from a list of strings, with the use of string manipulation and the split method.

Experiment2.4

TITLE:

Write a program to create a list of tuples from given list having number and add its cube in tuple.

i/p: c= [2,3,4,5,6,7,8,9]

THEORY:

The code takes user input to create a list of numbers and then uses a list comprehension to generate a list of tuples, each containing a number from the input list and its cube.

CODE:

```
# Write a program to create a list of tuples from given list having number and add its
cube in tuple.

# i/p: c= [2,3,4,5,6,7,8,9]

numbers = [int(n) for n in input("Enter numbers separated by spaces: ").split()]

cubes_tuples = [(num, num ** 3) for num in numbers]

print("List of tuples giving cube for num:", cubes_tuples)
```

OUTPUT:

```
python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog2.4.py"
aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoun
kar/ashpython/labmanauaipython/prog2.4.py"
Enter numbers separated by spaces: 4 5 6
List of tuples giving cube for num: [(4, 64), (5, 125), (6, 216)]
aayushchoukar@Aayushs-MacBook-Air labmanauaipython %
```

CONCLUSION:

By using list comprehension, the program efficiently creates a list of tuples with numbers and their respective cubes with help of formula written in the code.

Experiment2.5

TITLE:

Write a program to compare two dictionaries in Python?

(By using == operator)

THEORY:

The program compares two dictionaries using the == operator. The dictionaries, dict1 and dict2, are checked for equality, and the result is stored in the variable equal.

CODE:

```
# Write a program to compare two dictionaries in Python?
# (By using == operator)

dict1 = {"a": 1, "b": 2, "c": 3}
dict2 = {"a": 1, "b": 2, "c": 3}

equal = dict1 == dict2

print("Dictionaries are equal." if equal else "Dictionaries are not equal.")
```

OUTPUT:


```
python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog2.5.py"
● aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoun
kar/ashpython/labmanauaipython/prog2.5.py"
Dictionaries are equal.
○ aayushchoukar@Aayushs-MacBook-Air labmanauaipython %
```

CONCLUSION:

The program uses the `==` operator to compare two dictionaries. If the dictionaries have the same key-value pairs, they are considered equal, otherwise, they are considered not equal.

Experiment 2.6

TITLE:

Write a program that creates dictionary of cube of odd numbers in the range.

THEORY:

The program takes user input of starting and ending numbers. It then creates a dictionary containing cubes of odd numbers within the specified range.

CODE:

```
# Write a program that creates dictionary of cube of odd numbers in the range.

start = int(input("Enter starting number: "))
end = int(input("Enter ending number: "))

cube_dict = {num: num ** 3 for num in range(start, end + 1) if num % 2 != 0}
```

```
print("Dictionary of Cubes for Odd Numbers:", cube_dict)
```

OUTPUT:

```
python3 -u "/Users/aayushchouunkar/ashpython/labmanauaipython/prog.2.6.py"
● aayushchouunkar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoun
kar/ashpython/labmanauaipython/prog.2.6.py"
Enter starting number: 1
Enter ending number: 5
Dictionary of Cubes for Odd Numbers: {1: 1, 3: 27, 5: 125}
○ aayushchouunkar@Aayushs-MacBook-Air labmanauaipython % █
```

CONCLUSION:

By using dictionary comprehension, the program creates a dictionary of cubes for odd numbers in the range.

Experiment 2.7

TITLE:

Write a program for various list slicing operation.

`a= [10,20,30,40,50,60,70,80,90,100]`

- i. Print Complete list
- ii. Print 4th element of list
- iii. Print list from 0th to 4th index.
- iv. Print list -7th to 3rd element
- v. Appending an element to list.
- vi. Sorting the element of list.
- vii. Popping an element.
- viii. Removing Specified element.
- ix. Entering an element at specified index.
- x. Counting the occurrence of a specified element.
- xi. Extending list.
- xii. Reversing the list.

THEORY:

This program uses list slicing operations on given list. It covers printing specific elements, slicing, appending, sorting, popping, removing, inserting, counting occurrences, extending, and reversing the list.

CODE:

```
# Write a program for various list slicing operation.

# List

a = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

```
# i. Print list

print("i. List:", a)

# ii. Print 4th element of list

print("ii. 4th element of list:", a[3])

# iii. Print list from 0th to 4th index

print("iii. List from 0th to 4th index:", a[:5])

# iv. Print list -7th to 3rd element

print("iv. List from -7th to 3rd element:", a[-7:3])

# v. Appending an element to list

a.append(110)

print("v. List after appending 110:", a)

# vi. Sorting the elements of list

a.sort()

print("vi. Sorted list:", a)

# vii. Popping an element

popped_element = a.pop()

print("vii. Popped element:", popped_element, "Updated list:", a)

# viii. Removing specified element

a.remove(60)

print("viii. List after removing 60:", a)

# ix. Entering an element at specified index
```

```

a.insert(2, 35)

print("ix. List after inserting 35 at index 2:", a)

# x. Counting the occurrence of a specified element
count_30 = a.count(30)

print("x. Occurrence of 30 in the list:", count_30)

# xi. Extending list
a.extend([120, 130])

print("xi. Extended list:", a)

# xii. Reversing the list
a.reverse()

print("xii. Reversed list:", a)

```

OUTPUT:

```

kar/ashpython/labmanaulpython/prog2.7.py"
i. List: [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
ii. 4th element of list: 40
iii. List from 0th to 4th index: [10, 20, 30, 40, 50]
iv. List from -7th to 3rd element: []
v. List after appending 110: [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110]
vi. Sorted list: [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110]
vii. Popped element: 110 Updated list: [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
viii. List after removing 60: [10, 20, 30, 40, 50, 70, 80, 90, 100]
ix. List after inserting 35 at index 2: [10, 20, 35, 30, 40, 50, 70, 80, 90, 100]
x. Occurrence of 30 in the list: 1
xi. Extended list: [10, 20, 35, 30, 40, 50, 70, 80, 90, 100, 120, 130]
xii. Reversed list: [130, 120, 100, 90, 80, 70, 50, 40, 30, 35, 20, 10]
aayushchoukar@Aayushs-MacBook-Air labmanaulpython %

```

CONCLUSION:

The program shows a variety of list manipulation operations using slicing.

Experiment 3.1

TITLE:

3.1 Write a program to extend a list in python by using given approach.

THEORY:

- i. By using + operator.**
- ii. By using Append ()**
- iii. By using extend ()**

CODE:

```
# Write a program to extend a list in python by using given approach.

# i. By using + operator.

# ii. By using Append ()

# iii. By using extend ()

list = [1, 2, 3]

# i. By using + operator

extend1 = list + [4, 5, 6]

print("i. Extended list using + operator:", extend1)

# ii. By using Append()

list.append(4)

list.append(5)

list.append(6)

print("ii. Extended list using Append():", list)

# iii. By using extend()

list = [1, 2, 3]
```

```
list.extend([4, 5, 6])

print("iii. Extended list using extend():", list)
```

OUTPUT:

```
python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog3.1.py"
● aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog3.1.py"
i. Extended list using + operator: [1, 2, 3, 4, 5, 6]
ii. Extended list using Append(): [1, 2, 3, 4, 5, 6]
iii. Extended list using extend(): [1, 2, 3, 4, 5, 6]
○ aayushchoukar@Aayushs-MacBook-Air labmanauaipython %
```

CONCLUSION:

The program uses multiple methods to extend a given list in python.

Experiment 3.2

TITLE:

Write a program to add two matrices.

THEORY:

This program takes user input for number of rows and columns, then asks the user to enter elements for two matrices. It performs matrix addition and displays the result.

CODE:

```
# Write a program to add two matrices.

rows = int(input("Enter the Number of rows : "))
column = int(input("Enter the Number of Columns: "))

print("Enter the elements of First Matrix: ")
```

```
matrix1= [[int(input()) for i in range(column)] for i in range(rows)]

print("First Matrix is: ")

for n in matrix1:

    print(n)

print("Enter the elements of Second Matrix:")

matrix2= [[int(input()) for i in range(column)] for i in range(rows)]

for n in matrix2:

    print(n)

result=[[0 for i in range(column)] for i in range(rows)]

for i in range(rows):

    for j in range(column):

        result[i][j] = matrix1[i][j]+matrix2[i][j]

print("The Sum of Above two Matrices is : ")

for r in result:

    print(r)
```

OUTPUT:


```
python3 -u "/Users/aayushchoukhar/ashpython/labmanauaipython/prog3.2.py"
aayushchoukhar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoukhar/ashpython/labmanauaipython/prog3.2.py"
Enter the Number of rows : 3
Enter the Number of Columns: 2
Enter the elements of First Matrix:
2
2
3
3
4
5
First Matrix is:
[2, 2]
[3, 3]
[4, 5]
Enter the elements of Second Matrix:
2
2
3
3
4
4
[2, 2]
[3, 3]
[4, 4]
The Sum of Above two Matrices is :
[4, 4]
[6, 6]
[8, 9]
```

CONCLUSION:

The program show matrix addition by utilising nested lists to represent matrices.

Experiment 3.3

TITLE:

Write a Python function that takes a list and returns a new list with distinct elements from the first list.

THEORY:

The code defines a function `getelements` that takes input of list, goes through it and then creates a new list containing only unique elements. It does this by checking whether each element is already present in the result list before appending

CODE:

```
# Write a Python function that takes a list and returns a new list with distinct elements
from the first list.

def getelements(list):

    distinctlist = []

    for element in list:

        if element not in distinctlist:

            distinctlist.append(element)

    return distinctlist

a = input("Enter elements for list: ")

userlist = a.split()

userlist = [int(element) for element in userlist]

result = getelements(userlist)

print("List:", userlist)

print("List with distinct elements:", result)
```

OUTPUT:

```
python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog3.3.py"
● aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog3.3.py"
Enter elements for list: 3 4 5 6 8 7 6 5
List: [3, 4, 5, 6, 8, 7, 6, 5]
List with distinct elements: [3, 4, 5, 6, 8, 7]
○ aayushchoukar@Aayushs-MacBook-Air labmanauaipython %
```

CONCLUSION:

By using a this loop, the code removes duplicate elements from the list, and creates a new list with distinct elements.

Experiment 3.4

TITLE:

Write a program to Check whether a number is perfect or not.

THEORY:

The code defines a function `perfectnum` to check if a given number is perfect. A perfect number is one whose sum of divisors (excluding itself) equals the number. The code calculates the sum and checks if it matches the input number

CODE:

```
# Write a program to Check whether a number is perfect or not.

def perfectnum(number):

    if number <= 0:
```

```

        return False

    divsum = sum([divisor for divisor in range(1, number) if number % divisor == 0])

    return divsum == number

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

result = perfectnum(check)

if result:

    print(f"{check} is a perfect number.")

else:

    print(f"{check} is not a perfect number.")

```

OUTPUT:

```

python3 -u "/Users/aayushchoukar/ashpython/labmanualpython/prog3.4.py"
● aayushchoukar@Aayushs-MacBook-Air labmanualpython % python3 -u "/Users/aayushchoukar/ashpython/labmanualpython/prog3.4.py"
Enter a number: 28
28 is a perfect number.
● aayushchoukar@Aayushs-MacBook-Air labmanualpython % python3 -u "/Users/aayushchoukar/ashpython/labmanualpython/prog3.4.py"
Enter a number: 456
456 is not a perfect number.
○ aayushchoukar@Aayushs-MacBook-Air labmanualpython % █

```

CONCLUSION:

The code identifies whether the entered number is a perfect number or not, and provides the result of the evaluation.

Experiment 3.5

TITLE:

Write a Python function that accepts a string and counts the number of upper- and lower-case letters.

string_test= 'Today is My Best Day'

THEORY:

The function count takes a string as input and calculates the count of uppercase and lowercase letters in the string using isupper and islower string methods

CODE:

```
# Write a Python function that accepts a string and counts the number of upper and
lower-case letters.

# string_test = 'Today is My Best Day'.

def count(string):

    upper_count = sum(1 for char in string if char.isupper())

    lower_count = sum(1 for char in string if char.islower())

    return upper_count, lower_count

test = input("Enter a string: ")

upper, lower = count(test)

print(f"Uppercase letters: {upper}, Lowercase letters: {lower}")
```

OUTPUT:

```
python3 -u "/Users/aayushchoukar/ashpython/labmanaulpython/prog3.5.py"
● aayushchoukar@Aayushs-MacBook-Air labmanaulpython % python3 -u "/Users/aayushchoukar/ashpython/labmanaulpython/prog3.5.py"
Enter a string: HELLO MY NAME IS AAYUSH
Uppercase letters: 19, Lowercase letters: 0
○ aayushchoukar@Aayushs-MacBook-Air labmanaulpython %
```



CONCLUSION:

On applying the function to the provided string, it counts and prints the number of uppercase and lowercase letters.

Experiment 4.1

TITLE:

Write a program to Create Employee Class & add methods to get employee details & print.

THEORY:

The code defines a simple Employee class with attributes such as employee ID, name, gender, city, and salary. The class has an `__init__` method to initialize these attributes.

CODE:

```
# Write a program to Create Employee Class & add methods to get employee details & print.

class Employee:

    def __init__(self, emp_id, name, gender, city, salary):

        self.emp_id = emp_id

        self.name = name

        self.gender = gender

        self.city = city

        self.salary = salary

def main():

    emp_id = input("Enter Employee ID: ")

    name = input("Enter Name: ")

    gender = input("Enter Gender: ")

    city = input("Enter City: ")

    salary = float(input("Enter Salary: "))

    employee = Employee(emp_id, name, gender, city, salary)

    print("\nEmployee Details:")
```

```

print("ID:", employee.emp_id)

print("Name:", employee.name)

print("Gender:", employee.gender)

print("City:", employee.city)

print("Salary:", employee.salary)

if __name__ == "__main__":

    main()

```

OUTPUT:

```

python3 -u "/Users/aayushchoukar/ashpython/labmanaulpython/prog4.1.py"
aayushchoukar@Aayushs-MacBook-Air labmanaulpython % python3 -u "/Users/aayushchoun
kar/ashpython/labmanaulpython/prog4.1.py"
Enter Employee ID: 345
Enter Name: PREM
Enter Gender: FEMALE
Enter City: MUMBAI
Enter Salary: 50000

Employee Details:
ID: 345
Name: PREM
Gender: FEMALE
City: MUMBAI
Salary: 50000.0
aayushchoukar@Aayushs-MacBook-Air labmanaulpython %

```

CONCLUSION:

When executed, the code prompts the user to input details for a new employee, creates an instance of the Employee class, and displays the entered information. This structure allows for easy management and representation of employee data in a clear and organised manner.

Experiment 4.2

TITLE:

4.2 Write a program to take input as name, email & age from user using combination of keywords argument and positional arguments (*args and **kwargs) using function,

THEORY:

The program uses a function with a combination of positional arguments (*args) and keyword arguments (**kwargs) to obtain user details for name, email, and age. The user input is collected in the main function and passed to the flexible get_user_details function for processing

CODE:

```
# Write a program to take input as name, email & age from user using combination of
keywords argument and positional arguments (*args and **kwargs) using function,

def userdetails(*args, **kwargs):

    name = args[0] if args else kwargs.get('name', 'Unknown')

    email = kwargs.get('email', 'Unknown')

    age = kwargs.get('age', 'Unknown')

    return name, email, age

def main():

    name_input = input("Enter your name: ")

    email_input = input("Enter your email: ")

    age_input = input("Enter your age: ")

    details = userdetails(name=name_input, email=email_input, age=age_input)

    print("\nUser Details:")

    print("Name:", details[0])
```

```
print("Email:", details[1])

print("Age:", details[2])

if __name__ == "__main__":

    main()
```

OUTPUT:

```
python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog4.2.py"
aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog4.2.py"
Enter your name: AAYUSH
Enter your email: aayushchoukyy@gmail.com
Enter your age: 18

User Details:
Name: AAYUSH
Email: aayushchoukyy@gmail.com
Age: 18
aayushchoukar@Aayushs-MacBook-Air labmanauaipython %
```

CONCLUSION:

By leveraging both positional and keyword arguments, this program allows users to input their details conveniently, providing a flexible and readable way to handle varying sets of information. The result is a clear display of the user's name, email, and age.

Experiment4.3

TITLE:

Write a program to admit the students in the different Departments(pgdm/btech)and count the students. (Class, Object and Constructor).

THEORY:

The Python code defines a Student class representing student details. The user is prompted to input information for multiple students & categorizes them into PGDM or B.Tech departments.

CODE:

```
# Write a program to admit the students in the different Departments (pgdm/btech) and count
the students. (Class, Object and Constructor).

class Student:

    count = 0

    def __init__(self):

        self.name = input("Enter Student Name: ")

        self.age = int(input("Enter Student Age: "))

        self.department = input("Enter Student Department (PGDM(p)/B.Tech(b)): ")
        .capitalize()

        Student.count += 1

    def display(self):

        print("Name:", self.name, "Age:", self.age, "Department:", self.department)

print("""----- STUDENT ADMIT -----""")
```

```
pgdm_students = []

btech_students = []

num_students = int(input("Enter The Total Number Of Students: "))

for _ in range(num_students):

    new_student = Student()

    new_student.display()

    if new_student.department == 'P':

        pgdm_students.append(new_student)

    elif new_student.department == 'B':

        btech_students.append(new_student)

print("*****")

print("\nTotal PGDM Department Students:")

for student in pgdm_students:

    student.display()

print("\nTotal B.Tech Department Students:")

for student in btech_students:

    student.display()

print("\nTotal Number Of students:", Student.count)
```

```

python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog4.3.py"
aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog4.3.py"
----- STUDENT ADMIT -----
Enter The Total Number Of Students: 2
Enter Student Name: AAYUSH CHOUNKEYYY
Enter Student Age: 18
Enter Student Department (PGDM(p)/B.Tech(b)): b
Name: AAYUSH CHOUNKEYYY Age: 18 Department: B
Enter Student Name: PREM THATI
Enter Student Age: 18
Enter Student Department (PGDM(p)/B.Tech(b)): p
Name: PREM THATI Age: 18 Department: P
*****

Total PGDM Department Students:
Name: PREM THATI Age: 18 Department: P

Total B.Tech Department Students:
Name: AAYUSH CHOUNKEYYY Age: 18 Department: B

Total Number Of students: 2
aayushchoukar@Aayushs-MacBook-Air labmanauaipython %

```

CONCLUSION:

On running the code, it collects student information, categorizes them by department, and presents detailed statistics, including the total number of students in each department and in total.

Experiment4.4

TITLE:

Write a program that has a class store which keeps the record of code and price of product display the menu of all product and prompt to enter the quantity of each item required and finally generate the bill and display the total amount.

THEORY:

The Python code defines a Store class to simulate a store's inventory system. It allows adding products, displaying the menu, and generating a bill based on user input for product codes and quantities.

CODE:

```
# Write a program that has a class store which keeps the record of code and price of product display the menu of all product and prompt to enter the quantity of each item required and finally generate the bill and display the total amount.
```

```
class Store:

    def __init__(self):

        self.products = {}

    def add_product(self, code, name, price):

        self.products[code] = {'name': name, 'price': price}

    def display_menu(self):

        print("Menu:")

        print("Code\tName\tPrice")

        for code, product in self.products.items():

            print(f"{code}\t{product['name']}\t{product['price']}")

    def generate_bill(self, order):
```

```

        total_amount = 0

        print("\n----- RECEIPT -----")

        print("Code\tName\tPrice\tQuantity\tTotal")

        for code, quantity in order.items():

            product = self.products[code]

            item_total = quantity * product['price']

            total_amount += item_total

print(f"{code}\t{product['name']}\t₹{product['price']}\t{quantity}\t₹{item_total}")

        print("\nTotal Amount: ₹{:.2f}\n".format(total_amount))

def main():

    store = Store()

    store.add_product('001', 'Bread', 25.00)

    store.add_product('002', 'Chips', 15.00)

    store.add_product('003', 'KitKat', 10.00)

    store.add_product('004', 'Coke', 20.00)

    store.add_product('005', 'Biscuit', 12.00)

    store.add_product('006', 'Butter', 35.00)

    store.add_product('007', 'Rice', 30.00)

    store.add_product('008', 'Lentils', 35.00)

    store.add_product('009', 'Suji', 40.00)

    store.add_product('010', 'Spice', 20.00)

    store.display_menu()

    order = {}

```

```

while True:

    code = input("Enter the product code (or 'done' to finish): ")

    if code.lower() == 'done':

        break

    elif code in store.products:

        quantity = int(input(f"Enter the quantity for {store.products[code]['name']}:
"))

        order[code] = quantity

    else:

        print("Invalid product code. Please enter a valid code.")

store.generate_bill(order)

if __name__ == "__main__":

    main()

```

OUTPUT:

```

python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog4.4.py"
aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoun
kar/ashpython/labmanauaipython/prog4.4.py"
Menu:
Code    Name    Price
001     Bread  ₹25.0
002     Chips  ₹15.0
003     KitKat ₹10.0
004     Coke   ₹20.0
005     Biscuit ₹12.0
006     Butter ₹35.0
007     Rice   ₹30.0
008     Lentils ₹35.0
009     Suji   ₹40.0
010     Spice  ₹20.0
Enter the product code (or 'done' to finish): █

```


TESTCASE:

```
aayushchoukar@Aayushs-MacBook-Air labmanualpython % python3 -u "/Users/aayushchoukar/ashpython/labmanualpython/prog4.4.py"
Menu:
Code    Name    Price
001     Bread  ₹25.0
002     Chips  ₹15.0
003     KitKat  ₹10.0
004     Coke   ₹20.0
005     Biscuit ₹12.0
006     Butter ₹35.0
007     Rice   ₹30.0
008     Lentils ₹35.0
009     Suji   ₹40.0
010     Spice  ₹20.0
Enter the product code (or 'done' to finish): 001
Enter the quantity for Bread: 2
Enter the product code (or 'done' to finish): 002
Enter the quantity for Chips: 2
Enter the product code (or 'done' to finish): done

----- RECEIPT -----
Code    Name    Price    Quantity    Total
001     Bread  ₹25.0    2           ₹50.0
002     Chips  ₹15.0    2           ₹30.0
```

CONCLUSION:

The program initializes a store with predefined products, displays the menu, and prompts the user to input product codes and quantities for their order. It then generates a bill with a clear receipt.

Experiment 4.5

TITLE:

Write a program to take input from user for addition of two numbers using (single inheritance).

THEORY:

The code defines a class `addition` with a method `add` to perform addition. Another class `values` inherits from `addition` and includes a method `get_input` to collect two numbers from the user. The program then creates an instance of `values`, gets user input, performs addition, and prints the result.

CODE:

```
# Write a program to take input from user for addition of two numbers using
# (single inheritance)

class addition:

    def add(self, a, b):

        return a + b

class values(addition):

    def get_input(self):

        num1 = int(input("Enter the first number: "))

        num2 = int(input("Enter the second number: "))

        return num1, num2

add_values = values()

numbers = add_values.get_input()

result = add_values.add(*numbers)

print(f"The sum of {numbers[0]} and {numbers[1]} is: {result}")
```

OUTPUT:

```
python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog4.5.py"
aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog4.5.py"
Enter the first number: 3
Enter the second number: 4
The sum of 3 and 4 is: 7
aayushchoukar@Aayushs-MacBook-Air labmanauaipython %
```

TESTCASE:

```
python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog4.5.py"
aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog4.5.py"
Enter the first number: 0
Enter the second number: 0
The sum of 0 and 0 is: 0
aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog4.5.py"
Enter the first number: 56456565
Enter the second number: 456456
The sum of 56456565 and 456456 is: 56913021
aayushchoukar@Aayushs-MacBook-Air labmanauaipython %
```

CONCLUSION:

The code uses a class hierarchy where values inherits the addition functionality from the base class addition. It prompts the user for two numbers, adds them using the inherited method, and displays the result.

Experiment 4.6

TITLE:

Write a program to create two base classes LU and ITM and one derived class.
(Multiple inheritance).

THEORY:

The code defines classes LetsUpgrade and ITM, each representing their courses. The CourseSelector class inherits from both LetsUpgrade and ITM. It allows users to view available subjects, get details about a specific subject, and enrol in a course based on user input and eligibility criteria.

CODE:

```
# Write a program to create two base classes LU and ITM and one derived class.
# (Multiple inheritance).

class LetsUpgrade:

    lu_courses=[{'Subject': 'Maths', 'Trainer': 'Saikiran', 'Duration': 100},
                {'Subject': 'Python', 'Trainer': 'Saikiran', 'Duration': 150},
                {'Subject': 'Web design', 'Trainer': 'Prasad', 'Duration': 130}]

class ITM:

    itm_courses=[{'Subject': 'Maths', 'Trainer': 'Sheetal', 'Duration': 90},
                 {'Subject': 'DSA', 'Trainer': 'Sumit', 'Duration': 200},
                 {'Subject': 'Computer Fundamentals', 'Trainer': 'Sumit', 'Duration': 150}]

class CourseSelector(LetsUpgrade, ITM):

    def print_subjects(self, selected_class):

        if selected_class == 'LetsUpgrade':

            subjects = [course['Subject'] for course in self.lu_courses]

        elif selected_class == 'ITM':

            subjects = [course['Subject'] for course in self.itm_courses]

        else:

            subjects = []
```

```

if not subjects:

    print(f"No subjects available for {selected_class}")

    return

print(f"Available subjects for {selected_class}: {subjects}")

selected_subject = input("Enter the subject you want details for: ")

if selected_subject in subjects:

    if selected_class == 'LetsUpgrade':

        selected_course = next(course for course in self.lu_courses if
course['Subject'] == selected_subject)

        elif selected_class == 'ITM':

            selected_course = next(course for course in self.itm_courses if
course['Subject'] == selected_subject)

        print(f"\nDetails of {selected_subject} in {selected_class}:")

        print(f"Trainer: {selected_course['Trainer']} sir")

        print(f"Duration: {selected_course['Duration']} hours")

    else:

        print(selected_subject, " is not available in ", selected_class)

enroll_option = input("\nDo you wish to enroll? (yes/no): ").lower()

if enroll_option == 'yes':

    name = input("Enter your Name: ")

    dob = input("Enter you DOB (DD/MM/YYYY): ")

    age = int(input("Enter your Age: "))

    marks = int(input("Enter your 12th Marks: "))

    location = input("Enter your Location: ")

    if marks >= 60:

```

```

        print("\nCongratulations! You are enrolled in ", selected_subject)

        print("\n----- Student Details -----")

        print("Name: ", name)

        print("Age: ", age)

        print("Date Of Birth: ", dob)

        print("12th Marks: ", marks, "%")

        print("Location: ", location)

    else:

        print("\nSorry! You are not eligible for this course.")

else:

    print(selected_subject, " is not available in ", selected_class)

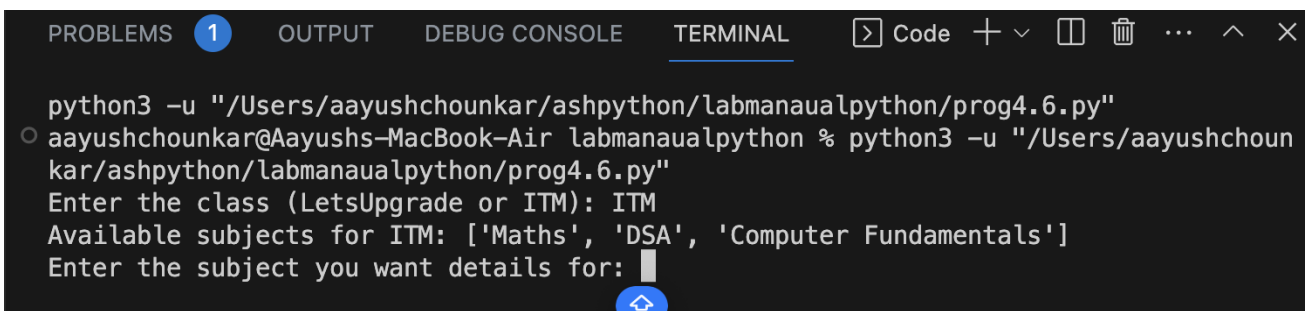
course_selector = CourseSelector()

selected_class = input("Enter the class (LetsUpgrade or ITM): ")

course_selector.print_subjects(selected_class)

```

OUTPUT:



```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL Code + - [ ] [ ] ... ^ X
python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog4.6.py"
aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoun
kar/ashpython/labmanauaipython/prog4.6.py"
Enter the class (LetsUpgrade or ITM): ITM
Available subjects for ITM: ['Maths', 'DSA', 'Computer Fundamentals']
Enter the subject you want details for: █


```

TESTCASE:

```
python3 -u "/Users/aayushchoukhar/ashpython/labmanauaipython/prog4.6.py"
aayushchoukhar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoukhar/ashpython/labmanauaipython/prog4.6.py"
Enter the class (LetsUpgrade or ITM): ITM
Available subjects for ITM: ['Maths', 'DSA', 'Computer Fundamentals']
Enter the subject you want details for: DSA

Details of DSA in ITM:
Trainer: Sumit sir
Duration: 200 hours

Do you wish to enroll? (yes/no): █
```



CONCLUSION:

The program provides a structured way for users to explore and enroll in courses offered by LetsUpgrade or ITM. It incorporates inheritance to reuse course information from the respective classes, demonstrates user interaction for course selection, and evaluates eligibility criteria for enrollment.

Experiment 4.7

TITLE:

Write a program to implement Multilevel inheritance,
Grandfather→Father→Child to show property inheritance from grandfather to child.

THEORY:

The code establishes a class hierarchy with Grandfather, Father, and Child classes representing successive generations. Each class has attributes related to assets, business, and education.

CODE:

```
# Write a program to implement Multilevel inheritance, Grandfather→Father→Child to show
property inheritance from grandfather to child.

class Grandfather:

    def __init__(self, assets):

        self.assets = assets

class Father(Grandfather):

    def __init__(self, assets, business):

        super().__init__(assets)

        self.business = business

class Child(Father):

    def __init__(self, assets, business, education):

        super().__init__(assets, business)

        self.education = education

grandfather_assets = 1000
```



```

father_assets = 5000

business_info = "Family Business"

child_education = "Computer Science Engineer"

#instance

grandfather = Grandfather(assets=grandfather_assets)

father = Father(assets=father_assets, business=business_info)

child = Child(assets=None, business=None, education=None)


child.assets = grandfather.assets + father.assets

child.business = father.business

child.education = child_education


print(f"\nGrandfather's assets: ₹{grandfather.assets}")

print(f"Father's assets: ₹{father.assets}")

print(f"Child's assets: ₹{child.assets}")

print(f"\nChild's business: {child.business}")

print(f"\nChild's education: {child.education}\n")

```

OUTPUT:

```

python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog4.7.py"
aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog4.7.py"

Grandfather's assets: ₹1000
Father's assets: ₹5000
Child's assets: ₹6000

Child's business: Family Business

Child's education: Computer Science Engineer

aayushchoukar@Aayushs-MacBook-Air labmanauaipython %

```

CONCLUSION:

The program creates instances of the classes, initializing and inheriting attributes from Grandfather to Father, and to Child. It demonstrates inheritance, encapsulation, & method overriding.

Experiment4.8

TITLE:

4.8 Write a program Design the Library catalogue system using inheritance take base class (library item) and derived class (Book, DVD & Journal) Each derived class should have unique attribute and methods and system should support Check in and check out the system. (Using Inheritance and Method overriding)

THEORY:

The code establishes a library management system with classes like LibraryItem, Book, DVD, and Journal. Each class models a type of library item with specific attributes.

CODE:

```
class LibraryItem:

    def __init__(self, title, author, item_id, copies_sold):

        self.title = title

        self.author = author

        self.item_id = item_id

        self.copies_sold = copies_sold

        self.availability = True

    def display_info(self):

        print(f"{self.item_id}. {self.title} by {self.author} ({'Available' if self.availability else 'Not Available'})")

class Book(LibraryItem):

    def __init__(self, title, author, item_id, genre, copies_sold):

        super().__init__(title, author, item_id, copies_sold)

        self.genre = genre

    def display_info(self):

        super().display_info()

        print(f"    Genre: {self.genre}")

class DVD(LibraryItem):
```

```

def __init__(self, title, director, item_id, duration, copies_sold):

    super().__init__(title, director, item_id, copies_sold)

    self.director = director

    self.duration = duration

def display_info(self):

    super().display_info()

    print(f"    Director: {self.director}, Duration: {self.duration} minutes")

class Journal(LibraryItem):

    def __init__(self, title, author, item_id, volume, copies_sold):

        super().__init__(title, author, item_id, copies_sold)

        self.volume = volume

    def display_info(self):

        super().display_info()

        print(f"    Volume: {self.volume}")

def display_catalog(items):

    print("\nLibrary Catalogue:")

    for item in items:

        item.display_info()

books = [

    Book("The Catcher in the Rye", "J.D. Salinger", "B001", "Fiction", 100),

    Book("To Kill a Mockingbird", "Harper Lee", "B002", "Classic", 150),

    Book("The Hobbit", "J.R.R. Tolkien", "B003", "Fantasy", 120),

]

dvds = [

    DVD("Inception", "Christopher Nolan", "D001", 148, 200),

    DVD("The Shawshank Redemption", "Frank Darabont", "D002", 142, 180),

    DVD("The Dark Knight", "Christopher Nolan", "D003", 152, 220),

]

journals = [

    Journal("Nature", "Various", "J001", "Vol. 587", 50),

    Journal("Science", "Various", "J002", "Vol. 374", 60),

]

while True:

```

```

print("\nChoose an option:")

print("1. View Books")
print("2. View DVDs")
print("3. View Journals")
print("4. Exit")

choice = input("Enter your choice (1-4): ")

if choice == "1":
    display_catalog(books)
elif choice == "2":
    display_catalog(dvds)
elif choice == "3":
    display_catalog(journals)
elif choice == "4":
    print("Exiting program. Thank you!")
    break
else:
    print("Invalid choice. Please enter a number between 1 and 4.")

item_id = input("Enter the item ID you want to borrow (or '0' to go back): ")

if item_id == "0":
    continue

quantity = int(input("Enter the quantity you want to borrow: "))

selected_item = None
catalog = books + dvds + journals
for item in catalog:
    if item.item_id == item_id:
        selected_item = item
        break

    if selected_item and selected_item.availability and quantity <=
selected_item.copies_sold:
        selected_item.availability = False

    print(f"\n{quantity} {selected_item.title}(s) successfully borrowed.")

```

```
else:

    print("\nInvalid selection or not enough copies available. Please try again.")
```

OUTPUT:

```
python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog4.8.py"
aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog4.8.py"

Choose an option:
1. View Books
2. View DVDs
3. View Journals
4. Exit
Enter your choice (1-4): █
```

TESTCASE:

```
python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog4.8.py"
aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog4.8.py"

Choose an option:
1. View Books
2. View DVDs
3. View Journals
4. Exit
Enter your choice (1-4): 1

Library Catalogue:
B001. The Catcher in the Rye by J.D. Salinger (Available)
    Genre: Fiction
B002. To Kill a Mockingbird by Harper Lee (Available)
    Genre: Classic
B003. The Hobbit by J.R.R. Tolkien (Available)
    Genre: Fantasy
Enter the item ID you want to borrow (or '0' to go back): 0

Choose an option:
1. View Books
2. View DVDs
3. View Journals
4. Exit
Enter your choice (1-4): 2

Library Catalogue:
D001. Inception by Christopher Nolan (Available)
    Director: Christopher Nolan, Duration: 148 minutes
D002. The Shawshank Redemption by Frank Darabont (Available)
    Director: Frank Darabont, Duration: 142 minutes
D003. The Dark Knight by Christopher Nolan (Available)
    Director: Christopher Nolan, Duration: 152 minutes
Enter the item ID you want to borrow (or '0' to go back): █
```

CONCLUSION:

The program provides a user interface to view library items by category and borrow them if available. It utilizes inheritance to represent different types of library items and maintains availability status. The program offers a practical example of object-oriented programming, encapsulation, and user interaction in a library context.

Experiment 5.1

TITLE:

5.1 Write a program to create my_module for addition of two numbers and import it in main script.

THEORY:

CODE:

MODULE:

```
def add_numbers(a, b):  
    return a + b
```

MAIN CODE:

```
import my_module  
  
num1 = int(input("Enter first number: "))  
num2 = int(input("Enter second number: "))  
  
result = my_module.add_numbers(num1, num2)  
  
print(f"The sum of {num1} and {num2} is: {result}")
```

OUTPUT:

```
python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog5.1mod.py"
● aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoun
kar/ashpython/labmanauaipython/prog5.1mod.py"
Enter first number: 5
Enter second number: 5
The sum of 5 and 5 is: 10
○ aayushchoukar@Aayushs-MacBook-Air labmanauaipython %
```

CONCLUSION:

Upon execution, the main script interacts with the custom module to add two numbers entered by the user, demonstrating the modular organization of code for better reusability and maintainability in Python.

Experiment 5.2

TITLE:

Write a program to create the Bank Module to perform the operations such as Check the Balance, withdraw and deposit the money in bank account and import the module in main file.

THEORY:

The Python program includes a module named `bank_module` with a `BankAccount` class for performing basic banking operations. The main script imports the module, creates a bank account, and interacts with it to check balance, withdraw, and deposit money, demonstrating modularity and encapsulation in programming.

CODE:

```
bank_module.py
class BankAccount:
    def __init__(self, account_holder, initial_balance=0):
        self.account_holder = account_holder
        self.balance = initial_balance
    def check_balance(self):
        return self.balance
    def deposit(self, amount):
        self.balance += amount
        return f"Deposited ₹{amount}. New balance: ₹{self.balance}"
    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
            return f"Withdrew ₹{amount}. New balance: ₹{self.balance}"
        else:
            return "Insufficient funds. Withdrawal denied."

main_script.py
from bank_module import BankAccount
account_holder_name = input("Enter account holder's name: ")
initial_balance = float(input("Enter initial balance: "))
account = BankAccount(account_holder_name, initial_balance)
print(f"\nAccount Holder: {account.account_holder}")
print("Initial Balance:", account.check_balance())
withdraw_amount = float(input("Enter the withdrawal amount: "))
print(account.withdraw(withdraw_amount))
deposit_amount = float(input("Enter the deposit amount: "))
print(account.deposit(deposit_amount))
print("Updated Balance:", account.check_balance())
```


OUTPUT:

```
Initial Balance: 5000.0
Enter the withdrawal amount: 1500
Withdrew ₹1500.0. New balance: ₹3500.0
Enter the deposit amount: 300
Deposited ₹300.0. New balance: ₹3800.0
Updated Balance: 3800.0
```

CONCLUSION:

The code exhibits a modular approach to banking operations, encapsulating functionality within a class. It showcases the use of custom modules for better code organization and reusability, providing a structured way to manage bank account-related tasks in Python.

Experiment 5.3

TITLE:

5.3 Write a program to create a package with name cars and add different modules (such as BMW, AUDI, NISSAN) having classes and functionality and import them in main file cars.

THEORY:

The code demonstrates the creation of a Python package named cars with modules (BMW, AUDI, NISSAN), each defining a class representing different car models. The main script imports these modules, creates instances of the car classes, and displays information about each car.

CODE:

```
BMW.py
class BMW:
    def __init__(self, model, color):
        self.model = model
        self.color = color
    def display_info(self):
        print(f"BMW {self.model} in {self.color}")
AUDI.py
class AUDI:
    def __init__(self, model, color):
        self.model = model
        self.color = color
    def display_info(self):
        print(f"AUDI {self.model} in {self.color}")
NISSAN.py
class NISSAN:
    def __init__(self, model, color):
        self.model = model
        self.color = color
    def display_info(self):
        print(f"NISSAN {self.model} in {self.color}")
script_main.py
from cars import BMW, AUDI, NISSAN
bmw_car = BMW.BMW(model="X5", color="Black")
audi_car = AUDI.AUDI(model="A4", color="Silver")
nissan_car = NISSAN.NISSAN(model="Altima", color="Blue")
bmw_car.display_info()
audi_car.display_info()
nissan_car.display_info()
```

OUTPUT:

```
BMW X5 in Black  
AUDI A4 in Silver  
NISSAN Altima in Blue
```

CONCLUSION:

The program highlights the benefits of organizing code into packages and modules, enhancing readability and maintainability. The separation of concerns allows for a structured representation of different car classes and their functionalities, providing a clear and modular design.

Experiment 6.1

TITLE:

6.1 Write a program to implement Multithreading. Printing “Hello” with one thread & printing “Hi” with another thread.

THEORY:

The program demonstrates multithreading with two threads, each printing "Hello" and "Hi" in a loop concurrently. The threading module is used to create and manage threads, showcasing parallel execution of tasks.

CODE:

```
# Write a program to implement Multithreading. Printing "Hello" with one thread & printing
"Hi" with another thread.

import threading

def print_hello():
    for _ in range(5):
        print("Hello")

def print_hi():
    for _ in range(5):
        print("Hi")

thread_hello = threading.Thread(target=print_hello)
thread_hi = threading.Thread(target=print_hi)

thread_hello.start()
thread_hi.start()

thread_hello.join()
```

```
thread_hi.join()
```

OUTPUT:

```
python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog6.1.py"
aayushchoukar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoukar/ashpython/labmanauaipython/prog6.1.py"
Hello
Hello
HiHello
Hello

HiHello

Hi
Hi
Hi
```

CONCLUSION:

The code illustrates the simultaneous execution of "Hello" and "Hi" messages in an interleaved manner, highlighting the concurrent nature of multithreading. This example exemplifies the use of threads for parallel execution, enhancing program efficiency by utilizing multiple execution paths.

Experiment 7.1

Title: Write a program to use 'whether API' and print temperature of any city, also print the sunrise and sunset times for the same humidity of that area.

Theory:

This program utilizes the OpenWeather API to display current weather details for a given city. It uses the requests library to make an API request, fetches temperature information, converts temperatures from Kelvin to Celsius and Fahrenheit, and provides details like humidity, general weather description, sunrise, and sunset times

CODE:

```
import datetime as dt
import requests

baseurl="http://api.openweathermap.org/data/2.5/weather?"
api_key="2ae13b8340fd07ea38e13aa7c176532d"
city=input('Enter City: ')

def kel_to_cel_fahren(kelvin):
    celsius=kelvin-273
    fahrenheit=celsius*(9/5)+32
    return celsius,fahrenheit

url= baseurl + 'appid=' + api_key + '&q=' + city
response=requests.get(url).json()
# print(response)

temp_kelvin=response['main']['temp']
temp_celsius, temp_fahrenheit=kel_to_cel_fahren(temp_kelvin)
max_temp=response['main']['temp_max']
tempc,tempf=kel_to_cel_fahren(max_temp)
min_temp=response['main']['temp_min']
temc,temf=kel_to_cel_fahren(min_temp)
```

```

humidity=response['main']['humidity']

description=response['weather'][0]['description']

sunrise=dt.datetime.utcfromtimestamp(response['sys']['sunrise']+response['timezone'])
sunset=dt.datetime.utcfromtimestamp(response['sys']['sunset']+response['timezone'])


print(f"Tempertature in {city}: {temp_celsius:.2f}'C or {temp_fahrenheit}'F")
print(f"Maximum Tempertature in {city}: {tempc:.2f}'C or {tempf}'F")
print(f"Minimum Tempertature in {city}: {temc:.2f}'C or {temf}'F")
print(f"Humidity in {city}: {humidity}%")
print(f"General Weather in {city}: {description}")
print(f"Sunrises in {city} at {sunrise}.")
print(f"Sunsets in {city} at {sunset}.")

```

OUTPUT:

```

python3 -u "/Users/aayushchoukhar/ashpython/labmanauaipython/weatherAPI.py"
● aayushchoukhar@Aayushs-MacBook-Air labmanauaipython % python3 -u "/Users/aayushchoukhar/ashpython/labmanauaipython/weatherAPI.py"
Enter City: KARACHI
/Users/aayushchoukhar/ashpython/labmanauaipython/weatherAPI.py:25: DeprecationWarning: datetime.datetime.utcfromtimestamp() is deprecated and scheduled for removal in a future version. Use timezone-aware objects to represent datetimes in UTC: datetime.datetime.fromtimestamp(timestamp, datetime.UTC).
    sunrise=dt.datetime.utcfromtimestamp(response['sys']['sunrise']+response['timezone'])
/Users/aayushchoukhar/ashpython/labmanauaipython/weatherAPI.py:26: DeprecationWarning: datetime.datetime.utcfromtimestamp() is deprecated and scheduled for removal in a future version. Use timezone-aware objects to represent datetimes in UTC: datetime.datetime.fromtimestamp(timestamp, datetime.UTC).
    sunset=dt.datetime.utcfromtimestamp(response['sys']['sunset']+response['timezone'])
)
Tempertature in KARACHI: 19.05'C or 66.29000000000002'F
Maximum Tempertature in KARACHI: 19.05'C or 66.29000000000002'F
Minimum Tempertature in KARACHI: 19.05'C or 66.29000000000002'F
Humidity in KARACHI: 42%
General Weather in KARACHI: haze
Sunrises in KARACHI at 2024-01-02 07:16:31.
Sunsets in KARACHI at 2024-01-02 17:53:46.
○ aayushchoukhar@Aayushs-MacBook-Air labmanauaipython % █

```

CONCLUSION:

The program uses api key and says perfect temperature,weather conditions,sun rise and set time ,program is working accurately .

Experiment 7.2

TITLE:

Write a program to use the 'API' of crypto currency.

THEORY:

The program uses cryptp currency api and with the help of gui of tkinter it gives price of bitcoin .

CODE:

```
import requests
import tkinter as tk
from datetime import datetime

def trackbitcoin():
    url = "https://min-api.cryptocompare.com/data/price?fsym=BTC&tsyms=USD,JPY,EUR"
    response = requests.get(url).json()
    price = response["USD"]
    time = datetime.now().strftime("%H:%M:%S")

    labelPrice.config(text=str(price) + " $")
    labelTime.config(text="Updated at: " + time)

    canvas.after(1000, trackbitcoin)

canvas = tk.Tk()
```



```
canvas.geometry("400x500") # Corrected the typo here

canvas.title("BITCOIN TRACKER")


f1 = ("poppins", 24, "bold")
f2 = ("poppins", 22, "bold")
f3 = ("poppins", 18, "normal")


label = tk.Label(canvas, text="Bitcoins Price", font=f1)
label.pack(pady=20)


labelPrice = tk.Label(canvas, font=f2)
labelPrice.pack(pady=20)


labelTime = tk.Label(canvas, font=f3)
labelTime.pack(pady=20)


trackbitcoin() # Added this line to start tracking immediately


canvas.mainloop()
```

OUTPUT:

Bitcoins Price

45396.27 \$

Updated at: 22:21:24

CONCLUSION:

The program accurately perform the task and gives the price of bitcoin using GUI program is working accurately.