

PYTHON PROGRAMMING LAB



Prepared by:

Name of Student: Sparsh Sharma

Roll No: 37

Batch: 2023-27

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Exp. No	List of Experiment
1	1. Write a program to compute Simple Interest.
	2. Write a program to perform arithmetic, Relational operators.
	3. Write a program to find whether a given no is even & odd.
	4. Write a program to print first n natural number & their sum.
	5. Write a program to determine whether the character entered is a Vowel or not
	6. Write a program to find whether given number is an Armstrong Number.
	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>

	<p>iii)</p> <pre> * * * * *</pre>
2	2.1 Write a program that define the list of defines the list of define countries that are in BRICS.
	<p>2.2 Write a program to traverse a list in reverse order.</p> <ol style="list-style-type: none"> 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.
	<p>2.4 Write a program to create a list of tuples from given list having number and add its cube in tuple.</p> <p>i/p: c= [2,3,4,5,6,7,8,9]</p>
	<p>2.5 Write a program to compare two dictionaries in Python?</p> <p>(By using == operator)</p>
	2.6 Write a program that creates dictionary of cube of odd numbers in the range.

	<p>2.7 Write a program for various list slicing operation.</p> <p>a= [10,20,30,40,50,60,70,80,90,100]</p> <ol style="list-style-type: none"> Print Complete list Print 4th element of list Print list from 0th to 4th index. Print list -7th to 3rd element Appending an element to list. Sorting the element of list. Popping an element. Removing Specified element. Entering an element at specified index. Counting the occurrence of a specified element. Extending list. Reversing the list.
3	<p>3.1 Write a program to extend a list in python by using given approach.</p> <ol style="list-style-type: none"> By using + operator. By using Append () By using extend ()
	<p>3.2 Write a program to add two matrices.</p>
	<p>3.3 Write a Python function that takes a list and returns a new list with distinct elements from the first list.</p>
	<p>3.4 Write a program to Check whether a number is perfect or not.</p>
	<p>3.5 Write a Python function that accepts a string and counts the number of upper- and lower-case letters.</p> <p>string_test= 'Today is My Best Day'</p>
4	<p>4.1 Write a program to Create Employee Class & add methods to get employee details & print.</p>

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

Roll Number: 37

Experiment No: 1

Title: Write a program to compute Simple Interest.

Theory: In this I simply use simple interest formula which is. $(p*r*t)/100$

* In this we are asking for p which is principal

* r stands for rate of interest

* t is for time

And at the end print the value after calculating it

Code:

```
P=int(input("Enter The Value Of Principal :- "))
R=int(input("Enter Rate of interest in decimal (per year) :- "))
T=int(input("Enter The Time :- "))
si=(P*R*T)/100
print(si)
```

Output: (screenshot)

```
sparshsharma@Sparshs-MacBook-Air Lab Manual Python % /opt/homebrew/bin/python3 "/Users/sparshsharma/Desktop/Lab Manual Python/Q1s/Q1.1_Simple_Interest.py"
Enter The Value Of Principal :- 1000
Enter Rate of interest in decimal (per year) :- 12
Enter The Time :- 1
120.0
sparshsharma@Sparshs-MacBook-Air Lab Manual Python %
```

Test Case: Any two (screenshot)

```
sparshsharma@Sparshs-MacBook-Air Lab Manual Python % /opt/homebrew/bin/python3 "/Users/sparshsharma/Desktop/Lab Manual Python/Q1s/Q1.1_Simple_Interest.py"
Enter The Value Of Principal :- 5000
Enter Rate of interest in decimal (per year) :- 2
Enter The Time :- 12
1200.0
sparshsharma@Sparshs-MacBook-Air Lab Manual Python %
```

```
sparshsharma@Sparshs-MacBook-Air Lab Manual Python % /opt/homebrew/bin/python3 "/Users/sparshsharma/Desktop/Lab Manual Python/Q1s/Q1.1_Simple_Interest.py"
Enter The Value Of Principal :- 3333
Enter Rate of interest in decimal (per year) :- 23
Enter The Time :- 12
9199.08
sparshsharma@Sparshs-MacBook-Air Lab Manual Python %
```

Conclusion: This the way to calculate the Simple Interest by there taking the value from user and giving ti them

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 1.2

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

Theory: I asked user to input two numbers. Then the code performed and printed all type of arithmetic calculation. Then I compared two numbers with the help of relational operators. And printed the greater number

Code:

```
#Write a program to perform arithmetic, Relational operators.  
  
a = float(input("Enter first number :"))  
b = float(input("Enter Second number :"))  
  
print("Arithmetic operations are :-")  
  
print("Addition of :",a,"+",b,"is",a+b)  
print("Subtraction of :",a,"-",b,"is",a-b)  
print("Multiplication of :",a,"*",b,"is",a*b)  
print("Division of :",a,"/",b,"is",a/b)  
print("Modules of :",a,"%",b,"is",a%b)  
  
print(a,"//",b,"is",a//b)  
  
print(a,"**",b,"is",a**b)
```

```

if a>b:
    print(a,"is greater")
elif b>a:
    print(b,"is greater")
elif a == b:
    print("Both are equal")
else :
    print("Invalid Input ")

```

Output: (screenshot)

```

python/ Q1s/Q1.2_Oprations.py
sparshsharma@Sparshs-MacBook-Air Lab Manual Python % /opt/homebrew/bin/python3 "/Users/sparshsharma/Desktop/Lab
/Q1.2_Oprations.py"
Enter first number :10
Enter Second number :20
Arithmetic operations are :-
Addition of : 10.0 + 20.0 is 30.0
Subtraction of : 10.0 - 20.0 is -10.0
Multiplication of : 10.0 * 20.0 is 200.0
Division of : 10.0 / 20.0 is 0.5
Modules of : 10.0 % 20.0 is 10.0
10.0 // 20.0 is 0.0
10.0 ** 20.0 is 1e+20
20.0 is greater
sparshsharma@Sparshs-MacBook-Air Lab Manual Python %

```

Test Case: Any two (screenshot)

```

sparshsharma@Sparshs-MacBook-Air Lab Manual Python % /opt/homebrew/bin/python3 "
/Users/sparshsharma/Desktop/Lab Manual Python/Q1s/Q1.2_Oprations.py"
Enter first number :2
Enter Second number :4
Arithmetic operations are :-
Addition of : 2.0 + 4.0 is 6.0
Subtraction of : 2.0 - 4.0 is -2.0
Multiplication of : 2.0 * 4.0 is 8.0
Division of : 2.0 / 4.0 is 0.5
Modules of : 2.0 % 4.0 is 2.0
2.0 // 4.0 is 0.0
2.0 ** 4.0 is 16.0
4.0 is greater

```

```

Enter first number :5
Enter Second number :10
Arithmetic operations are :-
Addition of : 5.0 + 10.0 is 15.0
Subtraction of : 5.0 - 10.0 is -5.0
Multiplication of : 5.0 * 10.0 is 50.0
Division of : 5.0 / 10.0 is 0.5
Modules of : 5.0 % 10.0 is 5.0
5.0 // 10.0 is 0.0
5.0 ** 10.0 is 9765625.0
10.0 is greater

```

Conclusion: This is way we have to calculate all arithmetic calculation by taking there value from user and computer will give the value

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 1.3

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

Theory:

In this program we have to checking that is it number is even or odd

If the number is divisible by two then it is give you result as even and if the it number is not divisible by 2 it is odd number

Code:

```
a=int(input("Enter The Value for check Even and Odd :-"))
if a%2==0:
    print(" It Is Even Number ")
else:
    print(" It Is Odd Number ")
```

Output: (screenshot)

```
spارشsharma@Sparshs-MacBook-Air Lab Manual
/Q1.3_Even_Odd.py"
Enter The Value for check Even and Odd :-4
It Is Even Number
```

Test Case: Any two (screenshot)

```
spارشsharma@Sparshs-MacBook-Air Lab Manual
/Q1.3_Even_Odd.py"
Enter The Value for check Even and Odd :-5
It Is Odd Number
```

```
spارشsharma@Sparshs-MacBook-Air Lab Manual
/Q1.3_Even_Odd.py"
Enter The Value for check Even and Odd :-4
It Is Even Number
```

Conclusion:- this the way we calculate even odd number

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 1.4

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

Theory: In this program print the first n natural number &and also print it's their sum by using for loop

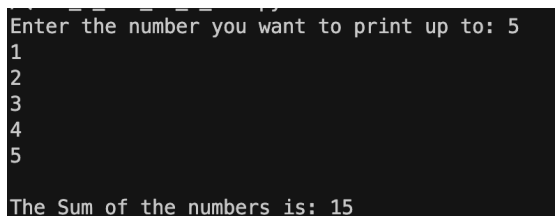
Code:

```
n = int(input("Enter the number you want to print up to: "))
sum_of_numbers = 0

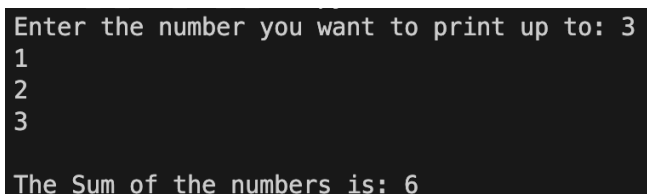
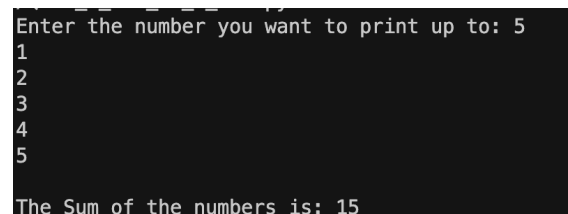
for i in range(1, n + 1):
    sum_of_numbers += i
    print(i)

print("\nThe Sum of the numbers is:", sum_of_numbers)
```

Output: (screenshot)

A screenshot of a terminal window showing the program's output for n=5. The prompt 'Enter the number you want to print up to: 5' is followed by the numbers 1, 2, 3, 4, and 5 printed on separate lines. At the bottom, it says 'The Sum of the numbers is: 15'.

Test Case: Any two (screenshot)

A screenshot of a terminal window showing the program's output for n=3. The prompt 'Enter the number you want to print up to: 3' is followed by the numbers 1, 2, and 3 printed on separate lines. At the bottom, it says 'The Sum of the numbers is: 6'.A screenshot of a terminal window showing the program's output for n=5. The prompt 'Enter the number you want to print up to: 5' is followed by the numbers 1, 2, 3, 4, and 5 printed on separate lines. At the bottom, it says 'The Sum of the numbers is: 15'.

Conclusion:

This is way to calculate the sum of n number and print it .

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 1.5

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

Theory: We have to check the character that is it vowel or not

So that make a list to the Vowel and check it is it Vowel or not

Code:

```
n=str(input(" Enter The value :"))
char=n[0]
Vowel=["A","E","I","O","U","a","e","i","o","u"]
if char in Vowel:
    print("It Is Vowel")
else:
    print("It Is Not Vowel")
```

Output: (screenshot)

```
/Q1.5_Vowel_Or_Not.py"
Enter The value :a
It Is Vowel
```

Test Case: Any two (screenshot)

```
Enter The value :z
It Is Not Vowel
```

```
Enter The value :o
It Is Vowel
```

Conclusion: In this Way we have to check is it Vowel Or Not

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 1.6

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

Theory:

An Armstrong Number is equal to the sum of its own digits each raised to the power of the number of digits.

- * Taking user input to check the number
- * Taking length of a in b
- * Storing the a in c as a is going to change.
- * Initialising sum as 0
- * Using while loop to calculate if the number raise to total number and their addition is same or not
- * If sum and num are same then its Armstrong else not
- * Print It

Code:

```
a=int(input("Enter a number: "))
b=len(str(a))
c=a
sum=0
while c!=0:
    d=c%10
    sum+=d**b
    c//=10
if sum==a:
    print(a,"is an armstrong number")
else:
    print(a,"is not an armstrong number")
```

Output: (screenshot)

```
Enter a number: 4  
4 is an armstrong number
```

Test Case: Any two (screenshot)

```
Enter a number: 10  
10 is not an armstrong number
```

```
Enter a number: 100  
100 is not an armstrong number
```

Conclusion:

This Way we can check the Armstrong Number

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 1.7

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

Theory:

The factorial of a number is the product of all positive integers up to that number.

In This we have to use the for loop to calculate the factorial No. It's mean that loop it continue running till the complication

Code:

```
n=int(input(" Enter The value :"))
if n<0:
    print(" it is not a factorial :")
else:
    fact=1
    for i in range (1,n+1):
        fact =fact*i
print(f" The Factorial Of {n} is : {fact} ")
```

Output: (screenshot)

```
Enter The value :4
The Factorial Of 4 is : 24
```

Test Case: Any two (screenshot)

```
Enter The value :6
The Factorial Of 6 is : 720
```

```
Enter The value :5
The Factorial Of 5 is : 120
```

Conclusion: This Way we can find the the factorial of the number

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 1.8

Title:

1.8 Write a program to print the following pattern

i)

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

ii)

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

iii)

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

Theory: In this we have print different patterns in different ways and with for loop

Code:

```
# I
rows = int(input("Enter number of rows: "))
k = 0
for i in range(1, rows+1):
    for space in range(1, (rows-i)+1):
        print(end=" ")

    while k!=(2*i-1):
        print("* ", end="")
        k += 1

    k = 0
    print()

# II
for i in range(1,6):
    for j in range(i):
        print(i,end='')
    print("\n")

# III

num = int(input("Enter Number:"))

for i in range(0, num):
    for j in range(0, i+1):
        print("*", end=" ")

    print()
```

Output: (screenshot)

```
Enter number of rows: 3
  *
 * * *
* * * * *
1
22
333
4444
55555

Enter Number:4
*
* *
* * *
* * * *
```

Test Case: Any two (screenshot)

```
Enter number of rows: 5
  *
 * * *
* * * * *
* * * * *
* * * * *
1
22
333
4444
55555

Enter Number:6
*
* *
* * *
* * * *
* * * * *
* * * * *
```

```
Enter number of rows: 3
  *
 * * *
* * * * *
1
22
333
4444
55555

Enter Number:4
*
* *
* * *
* * * *
```

Conclusion: By this we can print patterns the different in this way

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 2.1

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

Theory: It is a list of countries that are part of BRICS (Brazil, Russia, India, China, South Africa)

Code:

```
country=["Brazil","Russia","India","China","South Africa."]  
print(country)
```

Output: (screenshot)

```
['Brazil', 'Russia', 'India', 'China', 'South Africa.']
```

Test Case: Any two (screenshot)

```
['Brazil', 'Russia', 'India', 'China', 'South Africa.']
```

Conclusion: By this make a list

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 2.2

Title: Write a program to traverse a list in reverse order.

1.By using Reverse method.

2.By using slicing

Theory: in this program we have to follow first order is :- By using Reverse method. And 2nd Order is .By using slicing

Code:

```
lists = [12, 54, 28, 96, 34, 11, 123, 1]
lists.sort()
print(lists)
```

```
a=lists[1:4]
print(a)
```

Output: (screenshot)

```
[1, 11, 12, 28, 34, 54, 96, 123]
[11, 12, 28]
```

Test Case: Any two (screenshot)

```
[1, 11, 12, 28, 34, 54, 96, 123]
[11, 12, 28]
```

Conclusion: So the Conclusion is reverse method mean reverse the whole the list and using slicing mean from point to another point mean that range

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 2.3

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

Theory: In this code take the Email Id form user the divided into username and domain int the form of tuple for example :-

sparshvishan@gmail.com. So sparshvishan is the : username and

@gmail.com is come in Doomain

Code:n = 1

```
for i in range (0,n):
    Your_mail = input("Enter your gmail : ")
domain=()
users=()
for i in range(0,len(Your_mail)):
    if(Your_mail[i]=='@'):
        domain = Your_mail[i:]
        username = Your_mail[:i]
        i+=1
print("username:-",username)
print("domain:-",domain)
```

Output: (screenshot)

```
Enter your gmail : sparshvishan@gmail.com  
username:- sparshvishan  
domain:- @gmail.com
```

Test Case: Any two (screenshot)

```
Enter your gmail : ruhul@gmail.com  
username:- ruhul  
domain:- @gmail.com
```

```
Enter your gmail : gta5@gmail.com  
username:- gta5  
domain:- @gmail.com
```

Conclusion: So By this we can separate the username and domain the form of tuple

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 2.4

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: in this creating a list which is given in the question and make it into cube in tuple

Code:

```
a=[ (x**3) for x in range (2,10)]  
print(a)
```

Output: (screenshot)

```
Manual Python/Q2s/Q2.5_tuple.py"  
[8, 27, 64, 125, 216, 343, 512, 729]
```

Test Case: Any two (screenshot)

```
Manual Python/Q2s/Q2.5_tuple.py"  
[8, 27, 64, 125, 216, 343, 512, 729]
```

Conclusion: So by this process we can make the cube in the tuple

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 2.5

Title: Write a program to compare two dictionaries in Python?

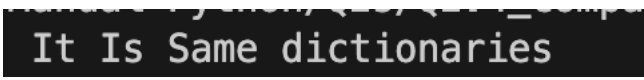
(By using == operator)

Theory: In this question we are comparing the two dictionaries which is `a={1:12,2:24,3:36}` and `b={1:12,2:24,3:36}` is with the help of if else operator we can find that is it same or not

Code:

```
a={1:12,2:24,3:36}
b={1:12,2:24,3:36}
if a==b:
    print(" It Is Same dictionaries ")
else:
    print("It is not ")
```

Output: (screenshot)



It Is Same dictionaries

Conclusion: By this process we can find similarities and difference between two dictionaries

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 2.6

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

Theory: In this code we have to make a dictionary cube of odd number and using an for loop and if condition to printing the value

Code:

```
a={x: (x**3) for x in range(1,11) if x%2==1}  
print(a)
```

Output: (screenshot)

```
Manual Python/Q2s/Q2.6_cube_odd.py"  
{3: 27, 5: 125, 7: 343, 9: 729}
```

Conclusion: The Conclusion is that printing the cube of the odd number which the use of for loop and if condition

Name of Student: Spارش Sharma

Roll Number: 37

Experiment No: 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. Extending list.**
- xi. Reversing the list.**

Theory: In this problem we have to solve multiple statement problem

Code:

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

```
# ii  
z=a[4]  
print(z)
```

```
# iii
```

```
b = a[0:4]
print(b)
```

```
# iv
c=a[-7:3]
print(c)
```

```
# v
d=a.append(110)
print(a)
```

```
# vi
e=sorted(a)
print(e)
```

```
# vii
f=a.pop(10)
print(a)
```

```
# viii
g=a.remove(50)
print(a)
```

```
# ix
h=a[5]
print(h)
```

```
# x. Extending list.
new_element = [110, 120, 130, 140]
a.extend(new_element)
print(a)
```

```
# xi Reversing
a.reverse()
print(a)
```

Output: (screenshot)

```
Handwritten/ques/q25/q25_c15c_50001ng1py
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
50
[10, 20, 30, 40]
[]
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110]
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110]
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
[10, 20, 30, 40, 60, 70, 80, 90, 100]
70
[10, 20, 30, 40, 60, 70, 80, 90, 100, 110, 120, 130, 140]
[140, 130, 120, 110, 100, 90, 80, 70, 60, 40, 30, 20, 10]
```

Test Case: Any two (screenshot)

```
Handwritten/ques/q25/q25_c15c_50001ng1py
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
50
[10, 20, 30, 40]
[]
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110]
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110]
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
[10, 20, 30, 40, 60, 70, 80, 90, 100]
70
[10, 20, 30, 40, 60, 70, 80, 90, 100, 110, 120, 130, 140]
[140, 130, 120, 110, 100, 90, 80, 70, 60, 40, 30, 20, 10]
```

Conclusion: By this Problem we have to solved multiple problem in one form

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 3.1

Title: 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 ()

Theory: This this problem we have solve one problem but into three method

- i. By using + operator. It using by + operator
- ii. By using Append () . It is using a adding a list in the last of the list
- iii. By using extend () . It is using a adding a new list in the last of old list

Code:

```
# I By using + operator.  
c=[22,24,26]  
z=a+c  
print(z)
```

```
# ii  
# By using Append ()  
z=[10,12,14]  
a.append(z)  
print(a)
```

```
# III By using extend ()  
b=[16,18,20]  
a.extend(b)  
print(a)
```

Output: (screenshot)

```
[2, 4, 6, 8]  
[2, 4, 6, 8, 22, 24, 26]  
[2, 4, 6, 8, [10, 12, 14]]  
[2, 4, 6, 8, [10, 12, 14], 16, 18, 20]
```

Test Case: Any two (screenshot)

```
[2, 4, 6, 8]  
[2, 4, 6, 8, 22, 24, 26]  
[2, 4, 6, 8, [10, 12, 14]]  
[2, 4, 6, 8, [10, 12, 14], 16, 18, 20]
```

Conclusion: The Conclusion is we solve one problem into three forms

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 3.2

Title: Write a program to add two matrices.

Theory: In this program there is two matrices . I hardcoded value in the form of x,y code then they will solve the matrices problem and use for loop

Code:

```
x = [[11,22,33],[44,55,66],[77,88,99]]
y = [[10,11,12],[1,3,7],[8,5,4]]

for i in range(0,3):
    for j in range(0,3):
        x[i][j]+=y[i][j]
print("Addition of matires is:-\n",x)
```

Output: (screenshot)

```
Addition of matires is:-
[[21, 33, 45], [45, 58, 73], [85, 93, 103]]
sparshsharma@Sparshs-MacBook-Air:~/lab/Manual$ P
```

Conclusion: Hence , The program accurately adds two matrices, producing the result in a new matrix.

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 3.3

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

Theory:

- * Making a function and making a list in that function
- * Checking the loop if there are distinct elements
- * Making a list a
- * And taking element from user input
- * Function call

Code:

```
def distin(a):  
    b=[]  
    for i in a:  
        if i not in b:  
            b.append(i)  
    print(b)  
a=[]  
while True:  
    c=int(input("Enter a number(0 to exit): "))  
    if c == 0:  
        break  
    else:  
        a.append(c)  
print(a)  
print("Distinct list:")  
distin(a)
```

Output: (screenshot)

```
Enter a number(0 to exit): 0  
[5, 1, 2, 4, 55, 32, 450]  
Distinct list:  
[5, 1, 2, 4, 55, 32, 450]
```

Test Case: Any two (screenshot)

```
[3, 23, 4, 5532, 23, 98]  
Distinct list:  
[3, 23, 4, 5532, 98]
```

```
Enter a number(0 to exit): 0  
[5, 1, 2, 4, 55, 32, 450]  
Distinct list:  
[5, 1, 2, 4, 55, 32, 450]
```

Conclusion:

- * Successfully generates a new list with distinct elements from the original list.
- Utilising set ensures uniqueness, simplifying the removal of duplicates

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 3.4

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

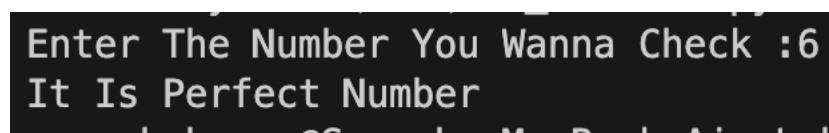
Theory:

A perfect number is one whose sum of proper divisors (excluding itself) equals the number.

Code:

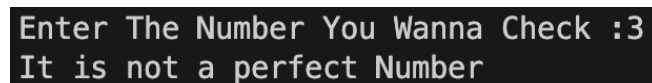
```
a=int(input("Enter The Number You Wanna Check :"))
sum=0
for i in range (1,a):
    if a%i==0:
        sum=sum+i
if sum==a:
    print("It Is Perfect Number")
else:
    print("It is not a perfect Number ")
```

Output: (screenshot)

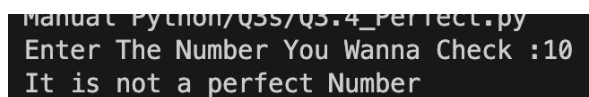


Enter The Number You Wanna Check :6
It Is Perfect Number

Test Case: Any two (screenshot)



Enter The Number You Wanna Check :3
It is not a perfect Number



Manual Python/QSS/Q3.4_Perfect.py
Enter The Number You Wanna Check :10
It is not a perfect Number

Conclusion:

The program effectively checks if a given number is perfect

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 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:

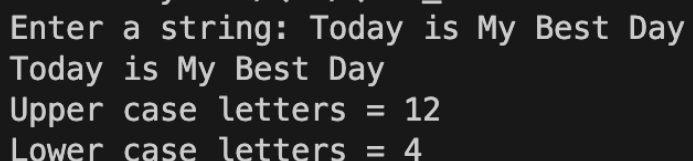
The function counts the number of upper and lower-case letters in a given string by iterating through each character and checking its case

Code:

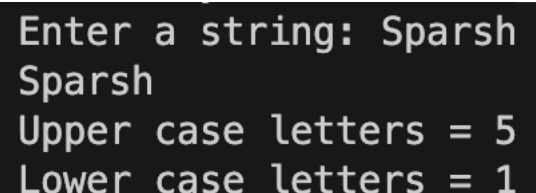
```
x=input("Enter a string: ")
uppercount=0
lowercount=0

for i in x:
    if i.isupper():
        lowercount+=1
    elif i.islower():
        uppercount+=1
print(x)
print('Upper case letters =',uppercount,'\nLower case letters =',lowercount)
```

Output: (screenshot)

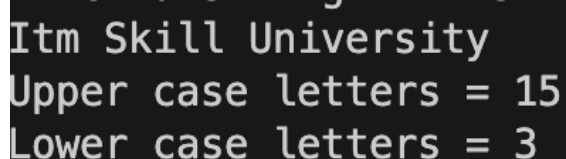


```
Enter a string: Today is My Best Day
Today is My Best Day
Upper case letters = 12
Lower case letters = 4
```



```
Enter a string: Sparsh Sparsh
Sparsh Sparsh
Upper case letters = 5
Lower case letters = 1
```

Test Case: Any two (screenshot)



```
Itm Skill University
Upper case letters = 15
Lower case letters = 3
```

```
Enter a string: Ironman
Ironman
Upper case letters = 6
Lower case letters = 1
```

Conclusion:

*The function accurately counts the number of upper and lower-case letters in the provided string.

- It uses simple character case checks to achieve this.

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 4.1

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

Theory: In this program we have to create a employee class and we have to add some methods to employee details for eg Name, age, number , Emp id ,Dapart

Code:

```
class Employee:

    def __init__(self,name,employee_id,age,number,department,salary):
        self.name = name
        self.employee_id = employee_id
        self.age = age
        self.contact_no = number
        self.dept = department
        self.salary= salary

    def display_details(self):
        print(" Your Name Is :",self.name)
        print(" Your ID is :",self.employee_id)
        print(" You are :",self.age," Years Old")
        print(" Contact Number :",self.contact_no)
        print(" Department : ",self.dept)
        print(" Salary : ",self.salary)

b=Employee("Rahul",1234,23,8923134323," Sales ",1000000)
print("\n\n Details of Employee \n ")
b.display_details()
```

Output: (screenshot)

Details of Employee

Your Name Is : Rahul

Your ID is : 1234

You are : 23 Years Old

Contact Number : 8923134323

Department : Sales

Salary : 1000000

Conclusion: Hence, using functions to hardcoded values and then assigning the attributes of the object these values and printing them to the user.

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 4.2

Title: 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 utilises a function with a combination of positional arguments (*args) and keyword arguments (**kwargs) to receive user input for name, email, and age.

Code:

```
def kbbfunctions(*args, **kwargs):  
    for i in args:  
        print("Name:", i)  
    for i in kwargs:  
        print(i, ":", kwargs[i])  
b=input("Enter your name: ")  
c=int(input("Enter your age: "))  
a=input("Enter your email: ")  
kbbfunctions(b, age=c, email=a)
```

Output: (screenshot)

```
Enter your name: Sparsh Sharma  
Enter your age: 18  
Enter your email: sparshvishan@gmail.com  
Name: Sparsh Sharma  
age : 18  
email : sparshvishan@gmail.com
```


Test Case: Any two (screenshot)

```
Enter your name: Sparsh Sharma  
Enter your age: 18  
Enter your email: sparshvishan@gmail.com  
Name: Sparsh Sharma  
age : 18  
email : sparshvishan@gmail.com
```

Conclusion:

This approach allows flexibility in function calls, enhancing readability and accommodating various input scenarios.

- It simplifies user input handling in the program.

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 4.3

Title:

Write a program to admit the students in the different

Departments(pgdm/btech)and count the students. (Class, Object and Constructor).

Theory: In this code we have to take a class with name of itm and divided into department of Pgdm and B-tech

Code:

```
class ITM:
    def __init__(self):
        self.name = input("Enter your name ")
        self.age = int(input("Enter your age "))
        self.department = int((input("Which department you want
in\n1.Pgdm\n""2.Btech\n ")))

    def display(self):
        if self.department == 1:
            print("*****Btech Department*****")
            print("Name:-",self.name)
            print("Age:-",self.age)

        elif self.department == 2:
            print("*****Pgdm Department*****")
            print("Name:-",self.name)
            print("Age:-",self.age)

list = []
n = int(input("How many students data you are entrying "))
```

```

for i in range(n):
    student = ITM()
    list.append(student)
for student in list:
    student.display()

```

Output: (screenshot)

```

How many students data you are entrying 2
Enter your name Sparsh
Enter your age 19
Which department you want in
1.Pgdm
2.Btech
1
Enter your name Rahul
Enter your age 20
Which department you want in
1.Pgdm
2.Btech
2
****Btech Department****
Name:- Sparsh
Age:- 19
****Pgdm Department****
Name:- Rahul
Age:- 20

```

Test Case: Any two (screenshot) :-

```

2
****Pgdm Department****
Name:- Tony Stark
Age:- 32

```

Conclusion: The program accurately counts and manages student admissions based on the specified departments.

```

How many students data you are entrying 2
Enter your name Sparsh
Enter your age 19
Which department you want in
1.Pgdm
2.Btech
1
Enter your name Rahul
Enter your age 20
Which department you want in
1.Pgdm
2.Btech
2
****Btech Department****
Name:- Sparsh
Age:- 19
****Pgdm Department****
Name:- Rahul
Age:- 20

```

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 4.4

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 program utilises a class named 'Store' to manage product records, display a menu, take user input for quantities, and generate a bill with total amount.

Code:

```
class Store:
    __itemCode = 0
    __price = 0

    def setdata(self):
        self.a = int(input("Product:\n1. Soap\n2. Shampoo\n3. Bread\n4. Milk\n"))

        if self.a == 1:
            self.__itemCode = 1
            self.__price = 10
            self.price = self.__price
            self.n = int(input("How many do you want: "))
        elif self.a == 2:
            self.__itemCode = 2
            self.__price = 100
            self.price = self.__price
            self.n = int(input("How many do you want: "))
        elif self.a == 3:
            self.__itemCode = 3
            self.__price = 40
            self.price = self.__price
            self.n = int(input("How many do you want: "))
        elif self.a == 4:
```

```

        self.__itemCode = 4
        self.__price = 30
        self.price = self.__price
        self.n = int(input("How many do you want: "))
    else:
        print("INVALID INPUT")

```

```

    def getdata(self):
        if self.a == 1:
            print(f"FOR {self.n} Packs of Soap, You Have To Pay: ${self.n * self.price}")
        elif self.a == 2:
            print(f"FOR {self.n} Packs of Shampoo, You Have To Pay: ${self.n * self.price}")
        elif self.a == 3:
            print(f"FOR {self.n} Packs of Bread, You Have To Pay: ${self.n * self.price}")
        elif self.a == 4:
            print(f"FOR {self.n} Packs of Milk, You Have To Pay: ${self.n * self.price}")

```

```

obj = Store()
obj.setdata()
obj.getdata()

```

Output: (screenshot)

```

Product:
1. Soap
2. Shampoo
3. Bread
4. Milk
4
How many do you want: 3
FOR 3 Packs of Milk, You Have To Pay: Rs.90

```

Test Case: Any two (screenshot)

```
Product:
1. Soap
2. Shampoo
3. Bread
4. Milk
1
How many do you want: 3
FOR 3 Packs of Soap, You Have To Pay: Rs.30
```

```
Product:
1. Soap
2. Shampoo
3. Bread
4. Milk
3
How many do you want: 4
FOR 4 Packs of Bread, You Have To Pay: Rs.160
```

Conclusion:

Hence at the last we are printing a bill with total amount at the end.

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 4.5

Title:

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

Theory:

Single inheritance is when there is a single parent class and a single child class which inherits the attributes and methods of the parent class.

*** It help to reduce the code line**

Code:

```
class Addition:
    @staticmethod
    def add(a, b):
        print("Sum:", a + b)

class Numbers(Addition):
    def getNumbers(self):
        c = float(input("Enter a number: "))
        d = float(input("Enter another number: "))
        self.add(c, d)

e = Numbers()
e.getNumbers()
```

Output: (screenshot)

```
Enter a number: 3
Enter another number: 4
Sum: 7.0
```

Test Case: Any two (screenshot)

```
Enter a number: 5  
Enter another number: 10  
Sum: 15.0
```

```
Enter a number: 22  
Enter another number: 11  
Sum: 33.0
```

Conclusion: this is help to reduce the length of code by using single inheritance

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 4.6

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

Theory:

The program utilises multiple inheritance with two base classes, LU and ITM, and one derived class inheriting from both class .

Code:

```
class LU:
    def __init__(self, lu_code):
        self.lu_code = lu_code

    def display_info(self):
        print(f"LU Code: {self.lu_code}")

    def method(self):
        print("LU Work")

class ITM:
    def __init__(self, itm_code):
        self.itm_code = itm_code

    def display_info(self):
        print(f"ITM Code: {self.itm_code}")

    def method(self):
        print("ITM Work")

class DerivedClass(LU, ITM):
    def __init__(self, lu_code, itm_code, derived_code):
        super().__init__(lu_code)
        self.derived_code = derived_code

    def display_info(self):
```

```
super(DerivedClass, self).display_info()  
print(f"Derived Code: {self.derived_code}")
```

```
def method(self):  
    super(DerivedClass, self).method()  
    print("Derived Work")
```

```
derived_object = DerivedClass("LU1010", "ITM420", "DL11")
```

```
derived_object.display_info()  
derived_object.method()
```

Output: (screenshot)

```
LU Code: LU1010  
Derived Code: DL11  
LU Work  
Derived Work
```

Test Case: Any two (screenshot)

```
LU Code: LU1010  
Derived Code: DL11  
LU Work  
Derived Work
```

Conclusion: Hence, using multiple inheritance to show from multiple classes(LU and ITM) and print their details for users

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 4.7

Title: Write a program to implement Multilevel inheritance,

Grandfather→Father→Child to show property inheritance from grandfather to child.

Theory: Multilevel inheritance is when there is a parent class which has a child class which in turn has a child class of its own.

Code:

```
class Grandfather:
    def __init__(self):
        self.name = "ABC"
        self.inherit = 10000
        self.purchase = 1000

class Father(Grandfather):
    def __init__(self):
        super().__init__()
        self.name = " XYZ " + self.name
        self.inherit = self.inherit
        self.purchase = 10000 + self.purchase

class Child(Father):
    def __init__(self, name):
        super().__init__()
        self.name = name + self.name
        self.inherit = self.inherit
        self.purchase = self.purchase
        print("Hello", self.name)
        print("Inherited property: Rs", self.inherit)
        print("Purchased property: Rs", self.purchase)
        print("Total property:", self.inherit + self.purchase)

a = input("Enter your name: ")
```

```
obj = Child(a)
```

Output: (screenshot)

```
Enter your name: Sparsh  
Hello Sparsh XYZ ABC  
Inherited property: Rs 10100  
Purchased property: Rs 12100  
Total property: 22200
```

Test Case: Any two (screenshot)

```
Hello Ironman XYZ ABC  
Inherited property: Rs 10000  
Purchased property: Rs 11000  
Total property: 21000
```

```
Inherited property: Rs 10100  
Purchased property: Rs 12100  
Total property: 22200
```

Conclusion:

So The Conclusion is the using multilevel inheritance to take name from the user and calculate inherited and purchased property from grandfather and father for the child and their full name using methods and attributes from the grandfather and father class.

Name of Student: Sparsh Sharma

Roll Number: 47

Experiment No: 4.8

Title: 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 program utilises inheritance, creating a base class (LibraryItem) and derived classes (Book, DVD, Journal) for items in a library.

Code:

```
class LibraryItem:
    def __init__(self, title, item_id):
        self.title = title
        self.item_id = item_id
        self.checked_out = False

    def display_info(self):
        print(f"Title: {self.title}")
        print(f"Item ID: {self.item_id}")
        print(f"Checked Out: {'Yes' if self.checked_out else 'No'}")

    def check_out(self):
        if not self.checked_out:
            print(f"Checking out {self.title}")
            self.checked_out = True
        else:
            print(f"{self.title} is already checked out.")

    def check_in(self):
        if self.checked_out:
            print(f"Checking in {self.title}")
            self.checked_out = False
        else:
            print(f"{self.title} is not checked out.")
```

```
class Book(LibraryItem):
    def __init__(self, title, item_id, author):
        super().__init__(title, item_id)
        self.author = author
```

```
    def display_info(self):
        super().display_info()
        print(f"Author: {self.author}")
```

```
class DVD(LibraryItem):
    def __init__(self, title, item_id, director, duration):
        super().__init__(title, item_id)
        self.director = director
        self.duration = duration
```

```
    def display_info(self):
        super().display_info()
        print(f"Director: {self.director}")
        print(f"Duration: {self.duration} minutes")
```

```
# Example usage
```

```
book1 = Book("The Catcher in the Rye", "B001", "J.D. Salinger")
dvd1 = DVD("Inception", "D001", "Christopher Nolan", 148)
```

```
book1.display_info()
book1.check_out()
book1.display_info()
book1.check_in()
book1.display_info()
```

```
print("\n")
```

```
dvd1.display_info()
dvd1.check_out()
dvd1.display_info()
dvd1.check_in()
dvd1.display_info()
```

Output: (screenshot)

```
Title: The Catcher in the Rye
Item ID: B001
Checked Out: No
Author: J.D. Salinger
Checking out The Catcher in the Rye
Title: The Catcher in the Rye
Item ID: B001
Checked Out: Yes
Author: J.D. Salinger
Checking in The Catcher in the Rye
Title: The Catcher in the Rye
Item ID: B001
Checked Out: No
Author: J.D. Salinger
```

Test Case: Any two (screenshot)

```
Title: Inception
Item ID: D001
Checked Out: No
Director: Christopher Nolan
Duration: 148 minutes
Checking out Inception
Title: Inception
Item ID: D001
Checked Out: Yes
Director: Christopher Nolan
Duration: 148 minutes
Checking in Inception
Title: Inception
Item ID: D001
Checked Out: No
Director: Christopher Nolan
Duration: 148 minutes
```

```
Title: The Catcher in the Rye
Item ID: B001
Checked Out: No
Author: J.D. Salinger
Checking out The Catcher in the Rye
Title: The Catcher in the Rye
Item ID: B001
Checked Out: Yes
Author: J.D. Salinger
Checking in The Catcher in the Rye
Title: The Catcher in the Rye
Item ID: B001
Checked Out: No
Author: J.D. Salinger
```

Conclusion:

Inheritance provides a hierarchical structure, allowing common functionality in the base class and unique features in derived classes

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 5.1

Title:

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

Theory:

Modules in Python encapsulate code, promoting code organization and reusability. They can be imported into other scripts.

Code:

```
class Addition:
    def add(self, x, y):
        print("Sum:", x + y)

import my_module

a = my_module.Addition()

b = float(input("Enter First number: "))
c = float(input("Enter Second Number: "))

a.add(b, c)
```

Output: (screenshot)

```
Enter the first number: 5  
Enter the second number: 7  
The sum of 5.0 and 7.0 is: 12.0
```

Conclusion:

So the conclusion is creating a module for addition of two values given by the user and printing it by importing the module in main program.

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 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:

Module is a collection of functions. Its functions can be used in another program by importing the module

Code:

```
import time
import random

def get_account_number():
    while True:
        try:
            account_no = int(input("\nEnter your card number: \n"))
            if 10000000 <= account_no <= 99999999:
                return account_no
            else:
                print("Account number should be of 8 numbers.\n")
        except ValueError:
            print("Invalid input. Please enter a valid number.\n")

def withdraw_money(balance):
    while True:
        try:
            money = float(input("\nEnter the money you want to withdraw ₹"))
            if money > balance:
                print("\nYour balance is lower than the amount you want to withdraw")
```

```

        elif money < 100:
            print("Minimal amount should be ₹100")
        else:
            return money
    except ValueError:
        print("Invalid input. Please enter a valid number.\n")

```

```

def deposit_money(balance):
    while True:
        try:
            money = float(input("\nEnter the amount you want to deposit ₹"))
            return balance + money
        except ValueError:
            print("Invalid input. Please enter a valid number.\n")

```

```

def transfer_money(balance, account_no):
    while True:
        try:
            money = float(input("\nEnter the amount you want to transfer: ₹"))
            ac = int(input("\nEnter the account you want to transfer to: "))
            if ac == account_no:
                print("\nCan't send money to yourself, can you?\n")
            elif not (10000000 <= ac < 99999999):
                print("\nAccount no. should be of 8 digits\n")
            elif money > balance or money < 100:
                if money > balance:
                    print("Not enough money in your account\n")
                else:
                    print("Minimal transfer amount is ₹100\n")
            else:
                time.sleep(2)
                balance -= money
                print(f"\nTransferred amount ₹{money:.3f} to Account with account no: {ac}\n")
                print(f"Your bank now has ₹{balance:.3f}")
                return balance
        except ValueError:
            print("Invalid input. Please enter a valid number.\n")

```

```

def main():
    c = random.randint(1000, 10000)
    account_no = get_account_number()
    print("\nChecking your card status, please wait :)\n")
    time.sleep(2)
    print("\nWELCOME TO ATM\n")

```

```

while True:
    print("\nWhat would you like to do?\n1. Withdrawal\n2.
Check balance\n3. Deposit money\n4. Transfer money\n5. Cancel\n")
    n = int(input())

    if n == 1:
        c -= withdraw_money(c)
        time.sleep(2)
    elif n == 2:
        print(f"\nYour account has ₹{c:.3f}\n")
    elif n == 3:
        c = deposit_money(c)
        print("Successfully deposited\n")
    elif n == 4:
        c = transfer_money(c, account_no)
    elif n == 5:
        print("\nTHANK YOU FOR CHOOSING US :)\n\n")
        return
    else:
        print("\nInvalid option :)\n")

if __name__ == "__main__":
    main()

```

Output: (screenshot)

```

What would you like to do?
1. Withdrawal
2. Check balance
3. Deposit money
4. Transfer money
5. Cancel

1

Enter the money you want to withdraw ₹12
Minimal amount should be ₹100

Enter the money you want to withdraw ₹100

```

Test Case: Any two (screenshot)

```
Enter the amount you want to transfer: ₹123
Enter the account you want to transfer to: 12312345
Transferred amount ₹123.000 to Account with account no: 12312345
Your bank now has ₹7399.000
```

```
What would you like to do?
1. Withdrawal
2. Check balance
3. Deposit money
4. Transfer money
5. Cancel

1

Enter the money you want to withdraw ₹12
Minimal amount should be ₹100

Enter the money you want to withdraw ₹100
```

Conclusion:

- **The Bank Module provides a structured approach for basic banking operations.**

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 5.3

Title: 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:

In Python, a package is a way of organising related modules

Modules within a package can be accessed using the dot notation.

Code: Module :-

```
class Nissan:
    def __init__(self, model):
        self.model = model
    def start_engine(self):
        print(f"Nissan {self.model} engine started.")
    def drive(self):
        print(f"Driving the Nissan {self.model}.")
```

Module 2 :-

```
class BMW:
    def __init__(self, model):
        self.model = model
    def start_engine(self):
```

```
        print(f"BMW {self.model} engine started.")
    def drive(self):
        print(f"Driving the BMW {self.model}.")
```

Module 3 :-

```
class Audi:
    def __init__(self, model):
        self.model = model
    def start_engine(self):
        print(f"Audi {self.model} engine started.")
    def drive(self):
        print(f"Driving the Audi {self.model}.")
```

```
from cars.bmw import BMW
from cars.audi import Audi
from cars.nissan import Nissan
```

```
bmw_car = BMW(model="X5")
bmw_car.start_engine()
bmw_car.drive()
print()
audi_car = Audi(model="A4")
audi_car.start_engine()
audi_car.drive()
print()
nissan_car = Nissan(model="Altima")
nissan_car.start_engine()
nissan_car.drive()
```

Output: (screenshot)

```
BMW X5 engine started.
Driving the BMW X5.

Audi A4 engine started.
Driving the Audi A4.

Nissan Altima engine started.
Driving the Nissan Altima.
```


Test Case: Any two (screenshot)

```
BMW X5 engine started.  
Driving the BMW X5.  
  
Audi A4 engine started.  
Driving the Audi A4.  
  
Nissan Altima engine started.  
Driving the Nissan Altima.
```

```
BMW X5 engine started.  
Driving the BMW X5.
```

Conclusion:

So the Conclusion is to creating a package containing modules for different car models, with each module containing methods for each car model.

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 6.1

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

Theory:

Multithreading is a concurrent execution mechanism where multiple threads operate independently within the same process, sharing resources like memory space.

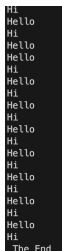
Code:

```
from threading import Thread
from time import sleep
def hi():
    for i in range(10):
        print("Hi")
        sleep(0.5)
def hello():
    for i in range(10):
        print("Hello")
        sleep(0.5)
t1=Thread(target=hi)
t2=Thread(target=hello)
t1.start()
t2.start()
t1.join()
t2.join()
print(" The End ")
```

Output: (screenshot)

```
Hi
Hello
Hi
Hello
Hello
Hi
Hello
Hi
Hello
Hi
Hello
Hi
Hello
Hi
Hello
Hi
Hello
Hi
Hello
Hi
The End
```

Test Case: Any two (screenshot)



```
Hi
Hello
Hi
Hello
Hello
Hi
Hello
Hi
Hello
Hi
Hello
Hi
Hello
Hi
Hello
Hi
Hello
Hi
Hello
Hi
The End
```

Conclusion:

- The program exemplifies multithreading by printing "Hello" and "Hi" concurrently.
- Multithreading enhances program efficiency by allowing simultaneous execution of tasks, improving responsiveness and performance.

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 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:

APIs (Application Programming Interfaces) allow different software systems to communicate with each other. Weather APIs provide access to weather-related data.

Code:

```
api_key=""
import requests
import datetime
city=input("Enter city: ")
response=requests.get(f"https://api.openweathermap.org/data/2.5/
weather?q={city}&APPID={api_key}&units=Metric")
a=response.json()
if 'message' in a:
    print("City not Found!")
else:
    print("\nCity:",city)
    print("Temperature:",a['main']['temp'], "C")
    print("Humidity:",a['main']['humidity'])
    print("Sunrise(IST):",datetime.datetime.fromtimestamp(a['sys']
['sunrise']))
    print("Sunset(IST):",datetime.datetime.fromtimestamp(a['sys']
['sunset']))
```

Output: (screenshot)

```
City: mumbai  
Temperature: 30.99 C  
Humidity: 48  
Sunrise(IST): 2023-12-28 07:10:01  
Sunset(IST): 2023-12-28 18:09:27
```

Test Case: Any two (screenshot)

```
City: mumbai  
Temperature: 30.99 C  
Humidity: 48  
Sunrise(IST): 2023-12-28 07:10:01  
Sunset(IST): 2023-12-28 18:09:27
```

```
City: patna  
Temperature: 23.96 C  
Humidity: 78  
Sunrise(IST): 2023-12-28 06:34:12  
Sunset(IST): 2023-12-28 17:07:05
```

Conclusion:

The program utilises the OpenWeatherMap API to fetch and display temperature, sunrise, sunset, and humidity for a given city.

Using APIs enables developers to integrate external data seamlessly into their applications, enhancing functionality and user experience.

Name of Student: Sparsh Sharma

Roll Number: 37

Experiment No: 7.2

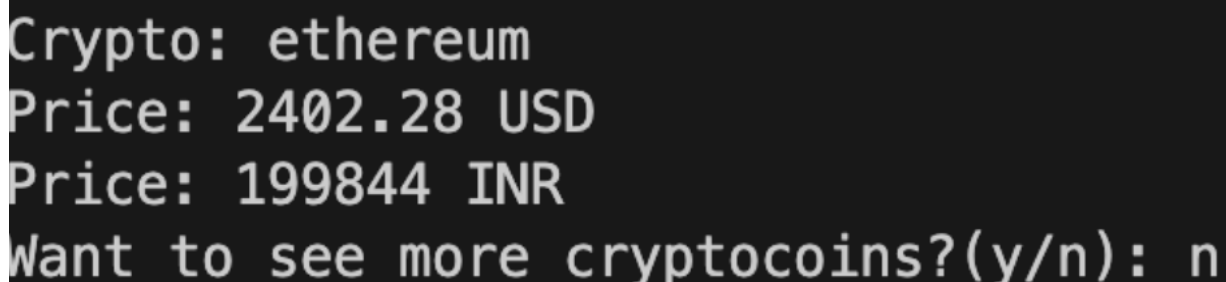
Title: Write a program to use the 'API' of crypto currency.

Theory: Cryptocurrency APIs provide a way for developers to access real-time data about cryptocurrencies, including prices, market data, and other relevant information.

Code:

```
import requests
api_id=""
while True:
    coin=input("Enter cryptocoin: ")
    response = requests.get(f"https://api.coingecko.com/api/v3/
simple/price?ids={coin}
&vs_currencies=usd,inr&x_cg_demo_api_key={api_id}")
    a=response.json()
    if coin in a:
        print("\nCrypto:",coin)
        print("Price:",a[coin]['usd'], "USD")
        print("Price:",a[coin]['inr'], "INR")
    else:
        print("Invalid cryptocoin!")
    b=input("Want to see more cryptocurrencies?(y/n): ")
    if b.lower() == "n":
        break
```

Output: (screenshot)



```
Crypto: ethereum
Price: 2402.28 USD
Price: 199844 INR
Want to see more cryptocurrencies?(y/n): n
```

Test Case: Any two (screenshot)

```
Crypto: ethereum  
Price: 2402.28 USD  
Price: 199844 INR  
Want to see more cryptocurrencies?(y/n): n
```

Conclusion:

- * The program demonstrates how to use a cryptocurrency API (CoinGecko) to retrieve and display the current price of a specified cryptocurrency.**
- APIs offer a powerful means for developers to incorporate dynamic data into applications, enhancing functionality and keeping information up-to-date.**

