

SCHOOL OF ENGINEERING & TECHNOLOGY  
BACHELOR OF TECHNOLOGY  
OBJECT ORIENTED PROGRAMMING USING PYTHON  
3<sup>RD</sup> SEMESTER  
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

**Lab: 1 – Write Program based on Functions**

**Practical: 1** Write a program in C to find the square of any number using the function.

**Source Code:**

```
n=int(input("Enter a number:"))
print("The divisors of the number are:")
for i in range(1,n+1):
    if(n%i==0):
        print(i)
```

**Output:**

```
Enter a number:6
The divisors of the number are:
1
2
3
6
```

**Practical: 2** Take two lists a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89] b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13] and write a program that returns a list that contains only the elements that are common between the lists (without duplicates). Make sure your program works on two lists of different sizes. Write this using at least one list comprehension.

**Source Code:**

```
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
c=[]
for i in range(len(a)):
    if a[i] in b:
        c.append(a[i])
c=set(c)
print(c)
```

**Output:**

```
{1, 2, 3, 5, 8, 13}
```

**Practical: 3** Let's say I give you a list saved in a variable: a = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]. Write one line of Python that takes this list and makes a new list that has only the even elements of this list in it.

**Source Code:**

```
a = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
b=[num for num in a if num%2==0]
print(b)
```

**Output:**

```
[4, 16, 36, 64, 100]
```

**Practical: 4** Rock – paper – scissor game.

**Source Code:**

```
player1 = input("Player 1: ")
player2 = input("Player 2: ")
if player1 == "rock" and player2 == "paper":
    print("Player 2 is the winner!!")
elif player1 == "rock" and player2 == "scissor":
    print("Player 1 is the winner!!")
elif player1 == "paper" and player2 == "scissor":
    print("Player 2 is the winner!!")
elif player1 == player2:
    print("It's a tie!!")
```

**Output:**

```
Player 1: paper
Player 2: scissor
Player 2 is the winner!!
```

**Practical: 5** Write a program to input roll no, student name, marks of physics, chemistry, and math out of 100. (0-100). Calculate total, percentage, calculate STATUS (pass, fail) if students' scores above 40 in all the 3 subjects the STATUS should be pass otherwise fail. Calculate GRADE if STATUS is pass.

**Source Code:**

```
roll_no = int(input("Roll No.: "))
student_name = input("Name: ")
marks_physics = int(input("Enter your marks in 'Physics': "))
marks_chemistry = int(input("Enter your marks in 'Chemistry': "))
marks_maths = int(input("Enter your marks in 'Maths': "))
print("\n-----Your Result-----\n")
print("Roll No.: ", roll_no)
```

```
print("Name: ", student_name)
total_marks = marks_physics + marks_chemistry + marks_maths
print("Total marks obtained out of 300: ", total_marks)
percentage = (total_marks / 300) * 100
print("Percentage: ", percentage)
if marks_physics > 40 and marks_chemistry > 40 and marks_maths > 40:
    status = 'PASS'
    print("Status:", status)
else:
    status = 'FAIL'
    print("Status:", status)
    print("\n-----\n")
if status == 'PASS':
    if percentage > 70:
        print("Grade: DISTINCTION")
        print("\n-----\n")
    elif percentage > 60:
        print("Grade: FIRST CLASS")
        print("\n-----\n")
    elif percentage > 50:
        print("Grade: SECOND CLASS")
        print("\n-----\n")
    else:
        print("Grade: PASS CLASS")
        print("\n-----\n")
```

**Output:**

```
Roll No.: 21124008
Name: Deep Mehta
Enter your marks in 'Physics': 86
Enter your marks in 'Chemistry': 82
Enter your marks in 'Maths': 94

-----Your Result-----

Roll No.: 21124008
Name: Deep Mehta
Total marks obtained out of 300: 262
Percentage: 87.33333333333333
Status: PASS
Grade: DISTINCTION
```

**Lab: 2**

**Practical: 1** Write a program which inputs a number. Display that number in word format.

**Source Code:**

```
def word(i):
    if i == '0':
        print("Zero", end=" ")
    elif i == '1':
        print("One", end=" ")
    elif i == '2':
        print("Two", end=" ")
    elif i == '3':
        print("Three", end=" ")
    elif i == '4':
        print("Four", end=" ")
    elif i == '5':
        print("Five", end=" ")
    elif i == '6':
        print("Six", end=" ")
    elif i == '7':
        print("Seven", end=" ")
    elif i == '8':
        print("Eight", end=" ")
    elif i == '9':
        print("Nine", end=" ")
```

```
num = input("Enter a number: ")
print("Entered number:", end=" ")
for i in num:
    word(i)
```

**Output:**

```
Enter a number: 420
Entered number: Four Two Zero
```

**Practical: 2** Program to print binary form any number using 16-bit representation. (without library function) (You can use list for 16-bit representation)

**Source Code:**

```
n = int(input("Enter a number: "))
binary = ""
for i in range(16):
    binary = str(n%2) + binary
    n = n//2
print(binary)
```

**Output:**

```
Enter a number: 51
00000000000110011
```

**Practical: 3** Write a Python program to create a sequence where the first four members of the sequence are equal to one, and each successive term of the sequence is equal to the sum of the four previous ones. Find the Nth member of the sequence.

**Source Code:**

```
num = int(input("Enter the number: "))
lst = [1, 1, 1, 1]
for i in range(num):
    if i < 4:
        print(lst[i], end=" ")
    else:
        nxtelement = lst[i - 4] + lst[i - 3] + lst[i - 2] + lst[i - 1]
        lst.append(nxtelement)
        print(lst[i], end=" ")
print("\nThe Sequence upto", num, "is:", lst)
```

**Output:**

```
Enter the number: 12
1, 1, 1, 1, 4, 7, 13, 25, 49, 94, 181, 349,
The Sequence upto 12 is: [1, 1, 1, 1, 4, 7, 13, 25, 49, 94, 181, 349]
```

**Practical 4:** Write a Python program to find the number of divisors of a given integer is even or odd.

**Source Code:**

```
n = int(input("Enter a number: "))
if n < 1:
    print("Enter number greater than 0")
elif n == 1:
    divisors = 1
else:
    divisors = 2
for i in range(2, n//2 + 1):
    if n % i == 0:
        divisors += 1
if divisors % 2 == 0:
    print("Number of divisors is even")
else:
    print("Number of divisors is odd")
```

**Output:**

```
Enter a number: 8
Number of divisors is even
```

**Practical 5:** Write a Python program to check whether three given lengths (integers) of three sides form a right triangle. Print "Yes" if the given sides form a right triangle otherwise print "No".

**Source Code:**

```
a = int(input("Enter side 1: "))
b = int(input("Enter side 2: "))
c = int(input("Enter side 3: "))
h = max(a, max(b, c))
RightAngled = False
if a == h:
    RightAngled = (b**2) + (c**2) == (a**2)
elif b == h:
    RightAngled = (a**2) + (c**2) == (b**2)
elif c == h:
```

```

RightAngled = (a**2) + (b**2) == (c**2)
if RightAngled:
    print("Yes")
else:
    print("No")

```

**Output:**

```

Enter side 1: 5
Enter side 2: 4
Enter side 3: 3
Yes

```

**Practical: 6****Source Code:**

```

print("Input x1,y1,x2,y2,x3,y3,xp,yp:")
x1, y1, x2, y2, x3, y3, x4, y4 = map(float, input().split())
print('PQ and RS are parallel.' if abs((x2 - x1)*(y4 - y3) - (x4 - x3)*(y2 - y1)) < 1e-10 else 'PQ and RS
are not parallel')

```

**Output:**

```

Input x1,y1,x2,y2,x3,y3,xp,yp:
12 24 14 28 6 12 4 8
PQ and RS are parallel.

```

**Lab: 3**

**Practical: 1** Write a program to display set of prime numbers between the given input range from user.

**Source Code:**

```

start = int(input("Enter start number: "))
end = int(input("Enter end number: "))
def isPrime(n):
    if n <= 1:
        return False
    for i in range(2, n//2 + 1):
        if n % i == 0:
            return False
    return True
for i in range(start, end):
    if isPrime(i):
        print(i, end = " ")

```

**Output:**

```
Enter start number: 1
Enter end number: 40
2 3 5 7 11 13 17 19 23 29 31 37
```

**Practical: 2** Write a program to check whether inputted string is palindrome or not.

**Source Code:**

```
text = input("Enter a string: ")
if text == text[::-1]:
    print(text, "is a palindrome")
else:
    print(text, "is not a palindrome")
```

**Output:**

```
Enter a string: abba
abba is a palindrome
```

**Practical: 3** Input data from keyboard in first 5 element. The last five elements must be

a[5]=count of odd nos  
a[6]=count of even nos  
a[7]=sum of even nos  
a[8]=sum of odd nos  
a[9]=sum of first five  
Display the whole list

**Source Code:**

```
a = []
for i in range(5):
    a.append(int(input("Enter a number: ")))
odd_count = 0
even_count = 0
sum_even = 0
sum_odd = 0
sum_all = 0
for n in a:
    if n % 2 == 0:
        even_count += 1
        sum_even += n
    else:
        odd_count += 1
```



```

    sum_odd += n
    sum_all += n
a.extend([odd_count, even_count, sum_even, sum_odd, sum_all])
print(a)

```

**Output:**

```

Enter a number: 2
Enter a number: 3
Enter a number: 4
Enter a number: 5
Enter a number: 6
[2, 3, 4, 5, 6, 2, 3, 12, 8, 20]

```

**Practical :4** An equation of the form is known as the quadratic equation. The values of x that satisfy the equation are known as the roots of the equation. A quadratic equation has two roots which are given by the following two formula

$$root1 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

$$root2 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

Write a program that requests the user to input the values of a, b, and c and outputs root1 and root2.

**Source Code:**

```

from math import *
a = int(input("Enter 'a': "))
b = int(input("Enter 'b': "))
c = int(input("Enter 'c': "))
d = (b*b) - (4*a*c)
d = sqrt(d)
b = -b
root1 = (b + d) / 2*a
root2 = (b - d) / 2*a
print("\nRoot 1: ", root1, "& Root2:", root2)

```

**Output:**

```

Enter 'a': 4
Enter 'b': 12
Enter 'c': 2
Root 1: -2.8339895114832743 & Root2: -45.166010488516726

```

**Practical: 5**

Write a program for lottery simulation. Generate 6-digit random number.

Allow user to input only 6-digit number.

If all the 6 digits matches then user wins 100000,

if 5 digits match in sequence then user wins 85000,  
if 4 digits match in sequence then user wins 50000,  
if 3 digits matches in sequence then user wins 20000,  
if 2 digit matches user wins 2000

**Source Code:**

```
import random
cpu_generated_number = str()
for i in range(0, 6):
    cpu_generated_number += str(random.randint(0, 9))
# print(cpu_generated_number)
count_match = 0
user_input = input("Enter your 6-digit number:")
if len(user_input) > 6:
    print("Please Enter a valid 6-digit Number.")
else:
    for i in user_input:
        if i in cpu_generated_number:
            count_match = count_match + 1
    if count_match == 6:
        print("Congratulations you win: Rs. 1,00,000!")
    elif count_match == 5:
        print("Congratulations you win: Rs. 85,000")
    elif count_match == 4:
        print("Congratulations you win: Rs. 50,000")
    elif count_match == 3:
        print("Congratulations you win: Rs. 20,000")
    elif count_match == 2:
        print("Congratulations you win: Rs. 2,000")
    else:
        print("Better Luck Next Time :).1")
```

**Output:**

```
Enter your 6-digit number:951753
Congratulations you win: Rs. 50,000
```

**Lab: 4**

**Practical: 1** Write a program to print following pattern. (hint: nested for loop)

Input number required no. of lines and pattern character from user.

```

@@@@@@
@@@@@
@@@@@
@@@@
@@@
@@
@

```

**Source Code:**

```

n = int(input("Enter number of lines: "))
ch = input("Enter pattern character: ")[0]
for i in range(n, 0, -1):
    for j in range(i):
        print(ch, end = " ")
    print("")

```

**Output:**

```

Enter number of lines: 6
Enter pattern character: @
@@@@@@
@@@@@
@@@@@
@@@@
@@@
@@
@

```

**Practical: 2** Write a OOP program to print following pattern. (hint: nested for loop)

Input number required no. of lines and pattern character from user.

```

      @
    @ @ @
  @ @ @ @ @
@ @ @ @ @ @ @
@ @ @ @ @ @ @ @ @
@ @ @ @ @ @ @ @ @ @ @

```

**Source Code:**

```

n = int(input("Enter number of lines: "))
for i in range(1, n+1):
    for j in range(n-i):
        print(" ", end = " ")
    for k in range(2*i - 1):

```

```
print("@", end = "")
print("")
```

**Output:**

```
Enter number of lines: 6
@
@@@
@@@@
@@@@@
@@@@@
@@@@@
@@@@@
```

**Practical: 3** Write a program to print following pattern. (hint: nested for loop)

Input number required no. of lines.

```
1
121
12321
1234321
123454321
12345654321
```

**Source Code:**

```
n = int(input("Enter number of lines: "))
for i in range(1, n+1):
    for j in range(n-i):
        print(" ", end = "")
    for k in range(1, i+1):
        print(k, end = "")
    for k in range(i-1, 0, -1):
        print(k, end = "")
    print("")
```

**Output:**

```
Enter number of lines: 6
1
121
12321
1234321
123454321
12345654321
```

**Practical: 4** Write a Python program to construct the following pattern, using a nested for loop.

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

```

* * * *
* * * * *
* * * *
* * *
* *
*

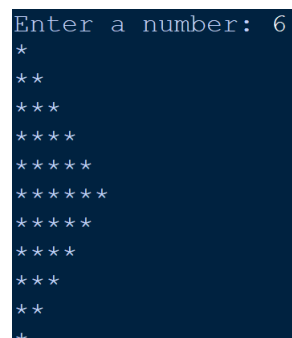
```

**Source Code:**

```

n = int(input("Enter a number: "))
count = 1
for i in range(1, 2*n):
    for j in range(count):
        print("*", end = "")
    if i < n:
        count = count + 1
    else:
        count = count - 1
    print("")

```

**Output:**


```

Enter a number: 6
*
**
***
****
*****
*****
*****
****
***
**
*

```

**Practical: 5** We are making n stone piles! The first pile has n stones. If n is even, then all piles have an even number of stones. If n is odd, all piles have an odd number of stones. Each pile must more stones than the previous pile but as few as possible. Write a Python program to find the number of stones in each pile.

**Source Code:**

```

a=int(input("input:"))
lst=[]
for i in range(a,3*a,2):
    lst.append(i)
print(lst)

```

**Output:**

```
input:2
[2, 4]

===== RESTART: C:/Users/DELL/AppData/Local/Programs/Python/Python31
input:10
[10, 12, 14, 16, 18, 20, 22, 24, 26, 28]

===== RESTART: C:/Users/DELL/AppData/Local/Programs/Python/Python31
input:17
[17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49]
```

**Practical: 6** Write a Python program find a list of integers with exactly two occurrences of nineteen and at least three occurrences of five

**Source Code:**

```
a=[]
n=int(input("enter size: "))
for i in range(n):
    a.append(int(input("enter number:")))
count=0
print("input:" ,a)
for i in range(n):
    if (a[i]==19):
        count=count+1
if(count==2 or count>=3):
    print("true")
else:
    print("false")
```

**Output:**

```
enter size: 5
enter number:19
enter number:4
enter number:6
enter number:19
enter number:2
input: [19, 4, 6, 19, 2]
true
```

**Lab: 5- Class and Object-I**

**Practical: 1** Write a OOP in python to input empid, name, basic salary, no. of experience in years. Calculate hra(35% of basic), da (58% of basic) and pf (9.5% of basic).

Also calculate bonus based on experience in years.

If experience in years is >= 30, bonus must be 59% of basic,

If experience in years is  $\geq 23$ , bonus must be 51% of basic,

If experience in years is  $\geq 15$ , bonus must be 45% of basic,

If experience in years is  $\geq 7$ , bonus must be 33% of basic,

If experience in years is  $< 7$ , bonus must be 16% of basic

Calculate netsalary as  $\text{basic} + \text{da} + \text{hra} - \text{pf} + \text{bonus}$ .

**Source Code:**

class Employee:

```
def __init__(self, empid, name, basic_salary, years):
    self.empid = empid
    self.name = name
    self.basic_salary = basic_salary
    self.years = years

def hra(self):
    return 0.35 * self.basic_salary

def da(self):
    return 0.58 * self.basic_salary

def pf(self):
    return 0.095 * self.basic_salary

def bonus(self):
    if self.years >= 30:
        bonus = 0.59 * self.basic_salary
    elif self.years >= 23:
        bonus = 0.51 * self.basic_salary
    elif self.years >= 15:
        bonus = 0.45 * self.basic_salary
    elif self.years >= 7:
        bonus = 0.33 * self.basic_salary
    else:
        bonus = 0.16 * self.basic_salary
    return bonus

def net_salary(self):
    n = self.basic_salary + self.da() + self.hra() + self.pf() + self.bonus()
    return n

e1 = Employee(2, "Deep", 100000, 4)
print("Net salary:", e1.net_salary())
```

**Output:**

```
Net salary: 218500.0
```

**Practical: 2** Write a OOP program to input Customer id, Customer name, electricity unit charges used.

Calculate electricity bill according to the given condition:

For first 50 units Rs. 0.50/unit

For next 100 units Rs. 0.75/unit

For next 100 units Rs. 1.20/unit

For unit above 250 Rs. 1.50/unit

An additional surcharge of 20% is added to the bill

**Source Code:**

```
class Customer:
```

```
    def __init__(self, cid, cname, units):
```

```
        self.cid = cid
```

```
        self.cname = cname
```

```
        self.units = units
```

```
    def calculate(self):
```

```
        u = self.units
```

```
        bill = 0
```

```
        if u <= 50:
```

```
            bill = u * 0.50
```

```
        elif u <= 150:
```

```
            bill = 25 + ((u-50) * 0.75)
```

```
        elif u <= 250:
```

```
            bill = 25 + 75 + ((u-150) * 1.20)
```

```
        else:
```

```
            bill = 25 + 75 + 120 + ((u-250) * 1.50)
```

```
        bill += 0.20 * bill
```

```
        return bill
```

```
cid1 = int(input("Enter customer id: "))
```

```
cname1 = input("Enter customer name: ")
```

```
units1 = int(input("Enter number of electricity units consumed: "))
```

```
c1 = Customer(cid1, cname1, units1)
```

```
print(c1.calculate())
```

**Output:**



```
Enter customer id: 21124008
Enter customer name: Deep
Enter number of electricity units consumed: 19
11.4
```

**Practical: 3** Write an OOP to calculate exponent from inputted base and power value.

E.g. Enter a base value : 3

Enter a power value : 4

For base 3 and power 4, the answer is 81

Driver code:

Base=MyNumber(3)

Power=MyNumber(4)

Exponent=Base^PowerExponent.display()

**Source Code:**

class Exponent:

def \_\_init\_\_(self, base, power):

self.base = base

self.power = power

def display(self):

print("For base {0} and power {1}, the answer is {2}".format(self.base, self.power, self.base \*\* self.power))

b = int(input("Enter a base value: "))

p = int(input("Enter a power value: "))

e1 = Exponent(b, p)

e1.display()

**Output:**

```
Enter a base value: 5
Enter a power value: 6
For base 5 and power 6, the answer is 15625
```

**Practical: 4** Write an OOP program to with list of 5 elements. Find the max from inputted numbers without using any library function.

**Source Code:**

class ListOp:

def \_\_init\_\_(self, l):

self.l = l

def max(self):

maxno = self.l[0]

for ele in self.l:

if ele > maxno:

```
        maxno = ele
    return maxno

l1 = [int(x) for x in input("Enter 5 space separated numbers: ").split()]
print("Max number:", ListOp(l1).max())
```

**Output:**

```
Enter 5 space separated numbers: 19 5 25 48 99
Max number: 99
```

**Practical: 5** Write an OOP program with list of 5 elements. Add a function in class to sort the elements in ascending / descending order.

**Source Code:**

```
class ListOp:
    def __init__(self, l):
        self.l = l

    def sort(self):
        #insertion sort - ascending order
        l = self.l
        n = len(l)
        for i in range(1, n):
            hole = i
            value = l[i]
            while hole > 0 and l[hole-1] > value:
                l[hole] = l[hole-1]
                hole -= 1
            l[hole] = value
        print("Sorted list:", l)

l1 = [int(x) for x in input("Enter 5 space separated numbers: ").split()]
ListOp(l1).sort()
```

**Output:**

```
Enter 5 space separated numbers: 99 25 30 70 55
Sorted list: [25, 30, 55, 70, 99]
```

**Practical: 6** Write a Python class to convert an integer to a roman numeral.

**Source Code:**

```
class Converter:
    def __init__(self):
        self.num = int(input("Enter a number: "))
        if self.num >= 2000:
```

```
        return "Please Enter numbers between 1 to 1999"
    self.string = []
    def selector(self, value):
        if self.value == 1:
            self.string += ['I']
        elif self.value == 2:
            self.string += ['II']
        elif self.value == 3:
            self.string += ['III']
        elif self.value == 4:
            self.string += ['IV']
        elif self.value == 5:
            self.string += ['V']
        elif self.value == 6:
            self.string += ['VI']
        elif self.value == 7:
            self.string += ['VII']
        elif self.value == 8:
            self.string += ['VIII']
        elif self.value == 9:
            self.string += ['IX']
        elif self.value == 10:
            self.string += ['X']
        elif self.value == 20:
            self.string += ['XX']
        elif self.value == 30:
            self.string += ['XXX']
        elif self.value == 40:
            self.string += ['XL']
        elif self.value == 50:
            self.string += ['L']
        elif self.value == 60:
            self.string += ['LX']
        elif self.value == 70:
            self.string += ['LXX']
```

```
elif self.value == 80:
    self.string += ['LXXX']
elif self.value == 90:
    self.string += ['XC']
elif self.value == 100:
    self.string += ['C']
elif self.value == 200:
    self.string += ['CC']
elif self.value == 300:
    self.string += ['CCC']
elif self.value == 400:
    self.string += ['CD']
elif self.value == 500:
    self.string += ['D']
elif self.value == 600:
    self.string += ['DC']
elif self.value == 700:
    self.string += ['DCC']
elif self.value == 800:
    self.string += ['DCCC']
elif self.value == 900:
    self.string += ['CM']
elif self.value == 1000:
    self.string += ['M']
def valuedigit(self):
    quotient = self.num
    count = 0
    while quotient > 0:
        remainder = quotient % 10
        if count == 0:
            self.value = remainder
            self.selector(self.value)
            count += 1
        elif count == 1:
            self.value = remainder * 10
```

```

        self.selector(self.value)
        count += 1
    elif count == 2:
        self.value = remainder * 100
        self.selector(self.value)
        count += 1
    else:
        self.value = remainder * 1000
        self.selector(self.value)
        quotient = quotient // 10
    return self.string[::-1]
num1 = Converter()
lst = num1.valuedigit()
print("The Roman Numeral is: ",end="")
for i in lst:
    print(i, end="")

```

**Output:**

```

Enter a number: 16
The Roman Numeral is: XVI

```

**Practical: 7** Write OOP program to calculate the area of triangle given its three sides. The formula or algorithm used is:  $\text{Area} = \sqrt{s(s-a)(s-b)(s-c)}$ , where  $s = (a + b + c) / 2$  or perimeter / 2 and a, b & c are the sides of triangle.

**Source Code:**

```

from math import sqrt
class Triangle:
    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c
    def area(self):
        s = (self.a + self.b + self.c) / 2
        return sqrt(s * (s-self.a) * (s-self.b) * (s-self.c))
a, b, c = [int(x) for x in input("Enter 3 sides of triangle: ").split()]
t1 = Triangle(a, b, c)
print("Area =", t1.area())

```

**Output:**

```
Enter 3 sides of triangle: 6 8 9  
Area = 23.525252389719434
```

**Lab: 6- Class and Object-I**

**Practical: 1** Create a class Numerics in Python which contains

- a. Function to calculate factorial of a number.
- b. Function to display multiplication table of given number.

NOTE: Do not use any library function.

**Source Code:**

```
from math import factorial

class Numerics:

    def factorial(self, n):

        if n == 0:

            return 1

        return n * factorial(n-1)

    def mulTable(self, n):

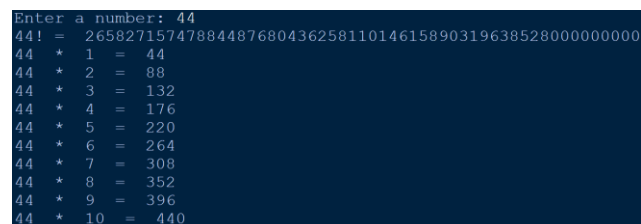
        for i in range(1, 11):

            print(n, " * ", i, " = ", n*i)

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

print(str(n) + "! = ", Numerics().factorial(n))

Numerics().mulTable(n)
```

**Output:**

```
Enter a number: 44
44! = 2658271574788448768043625811014615890319638528000000000
44 * 1 = 44
44 * 2 = 88
44 * 3 = 132
44 * 4 = 176
44 * 5 = 220
44 * 6 = 264
44 * 7 = 308
44 * 8 = 352
44 * 9 = 396
44 * 10 = 440
```

**Practical: 2** Create a class in Python which contains

- a. Function to reverse a number.
- b. Function to check whether the number is palindrome or not.

**Source Code:**

```
class NumOp:

    def reverse(self, n):

        rev = 0

        while n != 0:

            rev = (rev * 10) + (n % 10)

            n //= 10

        return rev
```

```
def isPalin(self, n):  
    return n == self.reverse(n)  
n = int(input("Enter a number: "))  
numop1 = NumOp()  
print("Entered number:", n)  
print("Reverse of number:", numop1.reverse(n))  
print("Is it palindrome?", numop1.isPalin(n))
```

**Output:**

```
Enter a number: 121  
Entered number: 121  
Reverse of number: 121  
Is it palindrome? True
```

**Practical: 3** Write a program which accepts the data of 5 cricket players(batsman)as follows.

- a. Name of the player
- b. Number of ODI's played/batted
- c. Total Runs scored by player in ODI Calculate the batting averages of all the players and display them. Also display the details of the players having maximum batting average and minimum batting average.

**Source Code:**

```
class Player:  
    def __init__(self, name, num_odi, runs):  
        self.name = name  
        self.num_odi = num_odi  
        self.runs = runs  
    def battingAvg(self):  
        return self.runs/self.num_odi  
print("Enter name, number of odis played and total runs scored of 5 players: ")  
x, y, z = input().split()  
player1 = Player(x, int(y), int(z))  
x, y, z = input().split()  
player2 = Player(x, int(y), int(z))  
x, y, z = input().split()  
player3 = Player(x, int(y), int(z))  
x, y, z = input().split()  
player4 = Player(x, int(y), int(z))  
x, y, z = input().split()
```



```

player5 = Player(x, int(y), int(z))

batting_avgs = [player1.battingAvg(), player2.battingAvg(), player3.battingAvg(), player4.battingAvg(),
player5.battingAvg()]

print("Batting average of player 1:", batting_avgs[0])
print("Batting average of player 2:", batting_avgs[1])
print("Batting average of player 3:", batting_avgs[2])
print("Batting average of player 4:", batting_avgs[3])
print("Batting average of player 5:", batting_avgs[4])
print("Minimum batting average:", min(batting_avgs))
print("Maximum batting average:", max(batting_avgs))

```

**Output:**

```

Deep 121 10000
Medhansh 100 6800
Poojan 111 9800
Yaksh 92 7500
Yash 80 6000
Batting average of player 1: 82.64462809917356
Batting average of player 2: 68.0
Batting average of player 3: 88.28828828828829
Batting average of player 4: 81.52173913043478
Batting average of player 5: 75.0
Minimum batting average: 68.0
Maximum batting average: 88.28828828828829

```

**Practical: 4** Design a voting application which imitates the current EVM's used in elections. In an EVM, add at least 4 candidates with necessary details like id, name of candidate, name of the party, no. of votes, etc. The voters can cast their votes in queue one by one. At the end provide facility to display results from EVM. Design appropriate classes for the required system.

**Source Code:**

```
class Candidate:
```

```
    num_of_candidates = 0
```

```
    def __init__(self, id, name_candidate, name_party):
```

```
        self.id = id
```

```
        self.name_candidate = name_candidate
```

```
        self.name_party = name_party
```

```
        self.votes = 0
```

```
        Candidate.num_of_candidates += 1
```

```
    def castVote(self):
```

```
        self.votes += 1
```

```
    @classmethod
```

```
    def displayResults(cls):
```

```
        candidates.sort(key = lambda x : x.votes, reverse = True)
```

```
        for can in candidates:
```

```
            print(can.name_candidate, "-", can.votes)
```

```
candidates = []
```

```
candidates.append(Candidate(1, "Winston Churchill", "Conservative Party"))
candidates.append(Candidate(2, "Narendra Modi", "BJP"))
candidates.append(Candidate(3, "Vladimir Putin", "Independent Party"))
candidates.append(Candidate(4, "JF Kennedy", "Democratic Party"))

choice = -1
while True:
    print("Type 1 to vote for", candidates[0].name_candidate)
    print("Type 2 to vote for", candidates[1].name_candidate)
    print("Type 3 to vote for", candidates[2].name_candidate)
    print("Type 4 to vote for", candidates[3].name_candidate)
    print("Type 5 to conclude elections")
    choice = int(input())
    if choice in [1, 2, 3, 4]:
        candidates[choice-1].castVote()
    elif choice == 5:
        break
    else:
        print("Incorrect choice")
Candidate.displayResults()
```

**Output:**

```
Type 1 to vote for Winston Churchill
Type 2 to vote for Narendra Modi
Type 3 to vote for Vladimir Putin
Type 4 to vote for JF Kennedy
Type 5 to conclude elections
2
Type 1 to vote for Winston Churchill
Type 2 to vote for Narendra Modi
Type 3 to vote for Vladimir Putin
Type 4 to vote for JF Kennedy
Type 5 to conclude elections
1
Type 1 to vote for Winston Churchill
Type 2 to vote for Narendra Modi
Type 3 to vote for Vladimir Putin
Type 4 to vote for JF Kennedy
Type 5 to conclude elections
3
Type 1 to vote for Winston Churchill
Type 2 to vote for Narendra Modi
Type 3 to vote for Vladimir Putin
Type 4 to vote for JF Kennedy
Type 5 to conclude elections
```

```
Type 1 to vote for Winston Churchill
Type 2 to vote for Narendra Modi
Type 3 to vote for Vladamir Putin
Type 4 to vote for JF Kennedy
Type 5 to conclude elections
5
Winston Churchill - 1
Narendra Modi - 1
Vladamir Putin - 1
JF Kennedy - 0
```

**Practical: 5** Write a Python class to find a pair of elements (indices of the two numbers) from a given array whose sum equals a specific target number. Note: There will be one solution for each input and do not use the same element twice. Input: numbers= [10,20,10,40,50,60,70], target=50 Output: 3, 4

**Source Code:**

```
class Solution:
```

```
    def twoSum(self, nums, target):
        lookup = {}
        for i, num in enumerate(nums):
            if target - num in lookup:
                return (lookup[target - num] + 1, i + 1)
            lookup[num] = i
```

```
print("index1=%d, index2=%d" % Solution().twoSum([10,20,10,40,50,60,70],50))
```

**Output:**

```
index1=3, index2=4
```

**Lab: 7 – Abstract Class & Inheritance**

**Practical: 1** Write a program, to create a child class Teacher (name, age) that will inherit the properties of Parent class Staff (role, department, salary)

**Source Code:**

```
class Staff:
```

```
    def __init__(self):
        self.role=input("enter your role: ")
        self.department=input("enter your department: ")
        self.salary=int(input("enter your salary: "))
```

```
class Teacher(Staff):
```

```
    def __init__(self):
        super().__init__()
        self.name=input("enter your name: ")
        self.age=int(input("enter your age: "))
```

```
def display(self):
    print("role: ",self.role)
    print("department: ",self.department)
    print("salary: ",self.salary)
    print("name: ",self.name)
    print("age: ",self.age)

t1=Teacher()
t1.display()
```

**Output:**

```
enter your role: teacher
enter your department: CSE
enter your salary: 500000
enter your name: Deep
enter your age: 19
role: teacher
department: CSE
salary: 500000
name: Deep
age: 19
```

**Practical: 2** Define a Base Class Vegetable having data member Color and member function getdata() which takes color as an input and putdata() which print the color as an output. Vegetable Class has one subclass named Tomato having data members weight and size and member function gtdata() which takes weight and size as an input and ptdata() which prints weight and size as output. Write a OOP Program which inherits the data of Vegetable class in Tomato class using Single Inheritance.

**Source Code:**

```
class Vegetable:
    def __init__(self):
        self.color=0

    def getdata(self):
        self.color=input("enter color: ")

    def putdata(self):
        print("color: ",self.color)

class Banana(Vegetable):
    def __init__(self):
        super().__init__()
        self.weight=0
        self.size=0

    def gtdata(self):
        self.weight=int(input("enter the weight: "))
        self.size=int(input("enter the size: "))
```

```

def ptdata(self):
    print("weight: ",self.weight)
    print("size: ",self.size)
t1=Banana()
t1.getdata()
t1.gtdata()
t1.putdata()
t1.ptdata()

```

**Output:**

```

enter color: yellow
enter the weight: 550
enter the size: 5
color: yellow
weight: 550
size: 5

```

**Practical: 3** Write a program to create a class Medicine which stores type of medicine, name of company, date of manufacturing. Class Tablet is inherited from Medicine. Tablet class has name of tablet, quantity per pack, price of one tablet as members. Class Syrup is also inherited from Medicine and it has quantity per bottle, dosage unit as members. Both the classes contain necessary member functions for input and output data. Write an OOP program that enters data for tablet and syrup, also display the data. Use the concepts of Multiple Inheritance.

**Source Code:**

```

class Medicine:
    def __init__(self):
        self.typeofmed=input("enter type of medicine: ")
        self.companyname=input("enter company's name: ")
        self.manufacuringdate=input("enter manufacuring date: ")
class Tablet(Medicine):
    def __init__(self):
        super().__init__()
        self.nameoftablet=input("enter name of tablet: ")
        self.quantity=int(input("enter quantity: "))
        self.price=int(input("enter price: "))
    def display1(self):
        print("type of med: ",self.typeofmed)
        print("company name: ",self.companyname)
        print("manufacuring date: ",self.manufacuringdate)
        print("nameoftablet: ",self.nameoftablet)

```

```
        print("quantity: ",self.quantity)
        print("price: ",self.price)
class Syrup(Medicine):
    def __init__(self):
        super().__init__()
        self.quantity=int(input("enter quantity: "))
        self.dosage=int(input("enter dosage: "))
    def display2(self):
        print("type of med: ",self.typeofmed)
        print("company name: ",self.companynname)
        print("manufacuring date: ",self.manufacuringdate)
        print("quantity: ",self.quantity)
        print("dosage: ",self.dosage)
t1=Tablet()
s1=Syrup()
print("Information about tablet: ")
t1.display1()
print("Information about syrup: ")
s1.display2()
```

**Output:**

```

enter type of medicine: tablet
enter company's name: xyz
enter manufacuring date: 10oct
enter name of tablet: ajhf
enter quantity: 12
enter price: 100
enter type of medicine: cold
enter company's name: kjh
enter manufacuring date: 12nov
enter quantity: 52
enter dosage: 1
Information about tablet:
type of med:  tablet
company name:  xyz
manufacuring date:  10oct
nameoftablet:  ajhf
quantity:  12
price:  100
Information about syrup:
type of med:  cold
company name:  kjh
manufacuring date:  12nov
quantity:  52
dosage:  1

```

**Practical: 4** Write a program to create a class Bank Account, Inherit Bank Account to Savings Account and Fixed Deposit Account. The savings account should allow customer to deposit and withdraw. The fixed-deposit account must calculate the due amount by inputting principal, rate of interest and no. of years value.

**Source Code:**

```

class Bank:
    def __init__(self):
        self.acno=int(input("enter account number: "))
        self.acname=input("enter account holder name: ")
        self.choice=int(input("choose your service(savings account, fixed deposit): "))
    def disp(self):
        return (self.choice)
class Saving(Bank):
    def __init__(self):
        self.balance=int(input("enter balance: "))
        self.trans=int(input("choose your transaction type(withdrawal,deposit): "))
    def disp1(self):
        if self.trans==1:
            wa=int(input("enter withdrawal amount"))

```

```

        print("your balance is ",self.balance-wa)
    if self.trans==2:
        da=int(input("enter deposit amount"))
        print("your balance is ",self.balance+da)
class Fixed(Bank):
    def __init__(self):
        self.p=int(input("enter principal amount: "))
        self.year=int(input("enternumber of years: "))
    def disp2(self):
        print("the bank currently 6.7%, you will recieve",self.p*(1+6.7/100)**self.year,"after",self.year,"years")
b1=Bank()
res=b1.disp()
if res==1:
    b2=Saving()
    b2.disp1()
else:
    b3=Fixed()
    b3.disp2()

```

**Output:**

```

enter account number: 20000
enter account holder name: Deep
choose your service(savings account, fixed deposit): 1
enter balance: 5000
choose your transaction type(withdrawal,deposit): 2
enter deposit amount: 1
your balance is 5001

```

**Practical: 5** Create an Abstract class vehicle having average as data and function getdata() and putdata(). Derive class car and truck from class vehicle having data members: fuel type (petrol, diesel, CNG) and no of wheels, respectively. Write an OOP program that enters the data of two cars and a truck and display the details of them.

**Source Code:**

```

from abc import ABC, abstractmethod
class Vehicle(ABC):
    @abstractmethod
    def __init__(self):
        pass
    @abstractmethod
    def getdata(self):
        pass

```



```
@abstractmethod
def putdata(self):
    pass
class Car(Vehicle):
    def __init__(self):
        self.fueltype=0
        self.numofwheels=0
    def getdata(self):
        self.fueltype=input("enter fuel type: ")
        self.numofwheels=int(input("enter number of wheels: "))
    def putdata(self):
        print("CARS")
        print("fuel type: ",self.fueltype)
        print("number of wheels: ",self.numofwheels)
class Truck(Vehicle):
    def __init__(self):
        self.fueltype=0
        self.numofwheels=0
    def getdata(self):
        self.fueltype=input("enter fuel type: ")
        self.numofwheels=int(input("enter number of wheels: "))
    def putdata(self):
        print("TRUCKS")
        print("fuel type: ",self.fueltype)
        print("number of wheels: ",self.numofwheels)
c1=Car()
c1.getdata()
c1.putdata()
c2=Car()
c2.getdata()
c2.putdata()
t1=Truck()
t1.getdata()
t1.putdata()
```

**Output:**

```

enter fuel type: petrol
enter number of wheels: 4
CARS
fuel type: petrol
number of wheels: 4
enter fuel type: diesel
enter number of wheels: 4
CARS
fuel type: diesel
number of wheels: 4
enter fuel type: petrol
enter number of wheels: 6
TRUCKS
fuel type: petrol
number of wheels: 6

```

**Practical: 6** Create abstract superclass Figure having abstract method def area (). Create concrete subclasses Rectangle and Circle of Figure which provide specific implementation of area () method. Rectangle has length and breadth data-members while circle has radius data member. Define parameterized constructors in both classes. In the driver part create objects of Rectangle and Circle classes with suitable length, breadth, and radius.

**Source Code:**

```
from abc import ABC, abstractmethod
```

```
class Shape(ABC):
```

```
    @abstractmethod
```

```
    def __init__(self):
```

```
        pass
```

```
    @abstractmethod
```

```
    def getdata(self):
```

```
        pass
```

```
    @abstractmethod
```

```
    def area(self):
```

```
        pass
```

```
class Circle(Shape):
```

```
    def __init__(self):
```

```
        self.r=0
```

```
    def getdata(self):
```

```
        self.r=int(input("enter radius: "))
```

```
    def area(self):
```

```
        self.area=3.14*self.r*self.r
```

```
        print("Area of circle: ",self.area)
```

```
class Rectangle(Shape):
```

```
    def __init__(self):
```

```
        self.l=0
        self.b=0
    def getdata(self):
        self.l=int(input("enter length: "))
        self.b=int(input("enter breath: "))
    def area(self):
        self.area=2*self.l*self.b
        print("Area of rectangle: ",self.area)
c1=Circle()
c1.getdata()
c1.area()
c2=Rectangle()
c2.getdata()
c2.area()
```

**Output:**

```
enter radius: 6
Area of circle:  113.03999999999999
enter length: 7
enter breath: 8
Area of rectangle:  112
```

**Practical: 7** Create Following class Vehicle having data members name, maximum speed, mileage, seating capacity. Create a Bus and Car child class that inherits from the Vehicle class. The default fare charge of any vehicle is seating capacity \* 100. If Vehicle is Bus instance, we need to add an extra 10% on full fare as a maintenance charge. So total fare for bus instance will become the final amount = total fare + 10% of the total fare.

**Source Code:**

```
from abc import ABC, abstractmethod
class Vehicle(ABC):
    @abstractmethod
    def __init__(self):
        pass
    @abstractmethod
    def fare(self):
        pass
class Bus(Vehicle):
    def __init__(self):
        self.name=input("enter name: ")
        self.speed=int(input("enter maximum speed: "))
```

```
self.mileage=int(input("enter mileage: "))
self.capacity=int(input("enter seating capacity: "))
def fare(self):
    self.amount=1
    self.tfare=self.capacity*100
    self.tfare=self.tfare*0.1+self.tfare
    print("total fare: ",self.tfare)
class Car(Vehicle):
    def __init__(self):
        self.name=input("enter name: ")
        self.speed=int(input("enter maximum speed: "))
        self.mileage=int(input("enter mileage: "))
        self.capacity=int(input("enter seating capacity: "))
    def fare(self):
        self.tfare=self.capacity*100
        print("total fare: ",self.tfare)
c1=Bus()
c1.fare()
c2=Car()
c2.fare()
```

**Output:**

```
enter name: bus
enter maximum speed: 120
enter mileage: 55
enter seating capacity: 40
total fare: 4400.0
enter name: car
enter maximum speed: 140
enter mileage: 35
enter seating capacity: 5
total fare: 500
```

**Lab: 8- Function and Operator Overloading**

**Practical: 1** Write a program to overload +, ==, !=, -, \*, >=, <= to perform operation between objects of MyNumber class.

**Source Code:**

```
class Mynumber:
    def __init__(self):
        self.num=34
```

```

def disp(self):
    a=55
    print(self.num+a)
    print(self.num==True)
    print(self.num!=True)
    print(self.num-a)
    print(self.num*a)
    print(self.num>=a)
    print(self.num<+a)

n1=Mynumber()
n1.disp()

```

**Output:**

```

89
False
True
-21
1870
False
True

```

**Practical: 2** Write an OOP program to accept two numbers and one mathematical operator. Calculate and display appropriate answer. Eg output Enter first number : 45 Enter mathematical operator : + Enter second number : 60 45+60 = 105 Note: Make use of Operator overloading concept in python Operator overloading is accomplished by following operator mapped functions +: `__add__` , \*: `__mul__` , /: `__truediv__` , -: `__sub__` //: `__floordiv__` , %: `__mod__` , \*\*: `__pow__`

**Source Code:**

```

class Mynumber:
    def __init__(self):
        self.a=int(input("Enter first number: "))
        self.op=input("Enter mathematical operator: ")
        self.b=int(input("Enter second number: "))
    def disp(self):
        if self.op=="+":
            self.res=self.a.__add__(self.b)
        elif self.op=="-":
            self.res=self.a.__sub__(self.b)
        elif self.op=="*":
            self.res=self.a.__mul__(self.b)
        elif self.op=="/":

```

```

        self.res=self.a.__truediv__(self.b)
    elif self.op=="//":
        self.res=self.a.__floordiv__(self.b)
    elif self.op=="%":
        self.res=self.a.__mod__(self.b)
    elif self.op=="**":
        self.res=self.a.__pow__(self.b)
    else:
        print("invalid operation")
    print(self.res)
n1=Mynumber()
n1.disp()

```

**Output:**

```

Enter first number: 12
Enter mathematical operator: *
Enter second number: 2
24

```

**Practical: 3** Write a OOP program to find Euclidean Distance. Also overload minus (operator) to find the same.

**Source Code:**

```

from math import sqrt, pow
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def __sub__(self, p2):
        return sqrt(pow(self.y - p2.y, 2) + pow(self.x - p2.x, 2))
p1 = Point(*[int(x) for x in input("Enter 1st point: ").split()])
p2 = Point(*[int(x) for x in input("Enter 2nd point: ").split()])
print("Distance between 2 points:", p1-p2)

```

**Output:**

```

Enter 1st point: 1 2
Enter 2nd point: 4 5
Distance between 2 points: 4.242640687119285

```

**Lab: 9 - INPUT-OUTPUT, MODULES & PACKAGES Files**

**Practical-1:** Write an Object Oriented Program which reads texts from a file. It must display file statistics a below.

- No. of sentences.
- No. of words.
- No. of total characters (Does not include whitespace)
- No. of whitespaces
- Total no. of digits, uppercase and lowercase letters

**Source Code :**

```
class statistics():
    def __init__(self):
        self.snts = 0 #sentences
        self.wrds = 0 #words
        self.char = 0 #characters without whitespaces
        self.space = 0 #whitespaces
        self.digits = 0 #digits
        self.upper = 0 #upper case characters
        self.lower = 0 #lower case characters
        self.file = None
    def calc(self, file = 'file.txt'):
        self.snts = 0
        self.wrds = 1
        self.char = 0
        self.space = 0
        self.digits = 0
        self.upper = 0
        self.lower = 0
        self.file = file
        try:
            fhand = open(file, "r")
        except:
            print("File Not Found")
            return
        else:
            for ele in fhand.readlines():
                for ch in ele:
                    if ch == ".":
                        self.snts += 1
                    if ch == " ":
                        self.wrds += 1
                        self.space += 1
                    elif ch == "\n":
                        self.wrds += 1
                    else:
                        self.char += 1
                        if ch.isdigit() is True:
                            self.digits += 1
                        if ch.islower() is True:
                            self.lower += 1
                        if ch.isupper() is True:
                            self.upper += 1
            fhand.close()
    def print(self):
        print(f"Number of Sentences: {self.snts}")
        print(f"Number of Words: {self.wrds}")
        print(f"Number of Characters: {self.char}")
        print(f"Number of Whitespaces: {self.space}")
        print(f"Number of Digits: {self.digits}")
        print(f"Number of Uppercase Characters: {self.upper}")
        print(f"Number of Lowercase Characters: {self.lower}")
def main():
    stat = statistics()
    stat.calc()
    stat.print()
main()
```

**Output :**

```
· Number of Sentences: 4
  Number of Words: 26
  Number of Characters: 99
  Number of Whitespaces: 25
  Number of Digits: 0
  Number of Uppercase Characters: 7
  Number of Lowercase Characters: 88
·
```

**Practical 2 :** Write an Object Oriented Program which copies the content of one file to another file in uppercase.

**Source Code :**

```
class Content():
    def __init__(self, src = "file.txt", destn = 'temp.txt'):
        self.src = src
        self.destn = destn
    def del_temp(self):
        try:
            ohand = open(self.destn, 'w')
        except:
            print("Destination File Doesn't exist")
            return
        ohand.write("")
        ohand.close()
    def copy(self):
        try:
            ihand = open(self.src, 'r')
        except:
            print("Source File Doesn't exist")
            return
        try:
            ohand = open(self.destn, 'w')
        except:
            print("Destination File Doesn't exist")
            return
        for ch in ihand:
            ohand.write(ch)
        ihand.close()
        ohand.close()
    def printfile(self):
        try:
            ihand = open(self.src, 'r')
        except:
```



```

        print("Source File Doesn't exist")
        return
    try:
        ohand = open(self.destn, 'r')
    except:
        print("Destination File Doesn't exist")
        return
    print("Source File: ")
    for ele in ihand.readlines():
        for ch in ele:
            print(ch,end="")
    print()
    ihand.close()
    print("Destination File: ")
    for ele in ohand.readlines():
        for ch in ele:
            print(ch,end="")
    print()
    ohand.close()
def main():
    content = Content()
    content.del_temp()
    print("Before Copying")
    content.printfile()
    content.copy()
    print("After Copying")
    content.printfile()
main()

```

**Output :**

```

Before Copying
Source File:
The birds are flying . The boy is walking . The Ganges are a great river system . The Narmada River flows from rift valley .
Destination File:

After Copying
Source File:
The birds are flying . The boy is walking . The Ganges are a great river system . The Narmada River flows from rift valley .
Destination File:
The birds are flying . The boy is walking . The Ganges are a great river system . The Narmada River flows from rift valley .

```

**Practical 3:** Write an Object-Oriented Program which creates vocabulary of words and counts each word in a document.

Eg. Content The birds are flying. The boy is walking. The Ganges are a great river system. The Narmada River flows from rift valley. output:

[(The,3), (birds,1), (are,1), (birds,1), (are,2), (flying,1), (boy,1), (river,2)]

**Source Code :**

```

class Vocab():
    def __init__(self, src = "file.txt"):
        self.src = src

```

```

        self.dict = dict()
    def calc(self):
        try:
            ihand = open(self.src, 'r')
        except:
            print("Source File Doesn't exist")
            return
        for line in ihand.readlines():
            for word in line.split():
                self.dict[word] = self.dict.get(word, 0) + 1
        ihand.close()
    def print_file(self):
        try:
            ihand = open(self.src, 'r')
        except:
            print("Source File Doesn't exist")
            return
        print("The file is:")
        for line in ihand.readlines():
            print(line)
    def print_dict(self):
        print("Dictionary of frequency of all words in file:")
        for k, v in self.dict.items():
            print(k, v)
def main():
    vocab1 = Vocab()
    vocab1.print_file()
    vocab1.calc()
    vocab1.print_dict()
main()

```

**Output :**

```

The file is:
The birds are flying . The boy is walking . The Ganges are a great river system . The Narmada River flows from rift valley .
Dictionary of frequency of all words in file:
The 4
birds 1
are 2
flying 1
. 4
boy 1
is 1
walking 1
Ganges 1
a 1
great 1
river 1
system 1
Narmada 1
River 1
flows 1
from 1
rift 1
valley 1

```

**Practical 4:** Write an Object Oriented Program which imitates Banking Transaction for a Saving account on an ATM machine. Store Bank Customer details permanently in a file. The Software is operated by a Bank

Customer through an ATM. After entering username and password (instead of card and pin), customer is allowed to view his details, Change his pin number, withdraw, deposit into his account. He must be able to transfer amount from his account to other also. (Use JSON data to handle a file)

### Source Code :

```
import json
pin=input("Enter Your Pin")
while True
    print("\t\t\twelcome To The SBI")
    with open('data.json','r+') as f:
        d=json.load(f)
    if pin in d:
        print(f"welcome {d[pin]['name']}")
        print("(1)Check Balance\n(2)Withdraw\n(3)deposit\n(4)Exit")
        choice=input("Enter your Choice")
        if choice == '1':
            print(f"The Remaining amount in Your Account is:- {d[pin]['amt']}")
            print("Thank You")
        elif choice == '2':
            width = int(input("enter the amount you want to widthdraw"))
            if width >= int(d[pin]['amt']):
                print("insufficient balance")
                print("Thank You")
            else:
                remain=int(d[pin]['amt'])- width
                d[pin]['amt'] = str(remain)
                print(f"widthdrawl successful\nRemaining Amount is:- {d[pin]['amt']}")
                print("Thank You")
            with open('data.json','w') as k:
                json.dump(d,k)
        elif choice == '3':
            depo = int(input("enter the amount you want to deposit"))
            drem=depo + int(d[pin]['amt'])
            [pin]['amt'] = str(drem)
            print(f"Deposit successful\nUpdated Amount is:- {d[pin]['amt']}")
            print("Thank You")
            with open('data.json','w') as k:
                json.dump(d,k)
        elif choice == '4':
            exit()
        else:
            print("Wrong Input Given Please Choose Correct Option")
```

### Output :

```

Enter Your Pin1111
welcome To The SBI
welcome samuel
1)Check Balance
2)Withdraw
3)deposit
4)Exit
Enter your Choice1
The Remaining amount in Your Account is:- 52000
Thank You

welcome To The SBI
welcome samuel
1)Check Balance
2)Withdraw
3)deposit
4)Exit
Enter your Choice

```

```

welcome To The SBI
welcome samuel
1)Check Balance
2)Withdraw
3)deposit
4)Exit
Enter your Choice2
enter the amount you want to widthdraw65800
insufficient balance
Thank You

```

**Practical 5:** Write a python program mathfinder.py which finds sum, average, maximum, minimum value entered through command line arguments.

**Source Code :**

```

import sys
lst = (1,2,3,4,5)
for ele in sys.argv[1:]:
    try:
        ele = int(ele)
    except ValueError:
        pass
    else:
        lst.append(ele)
print(f"List is: {lst}")
print(f"Sum of all values is {float(sum(lst))}")
print(f"Average of all values is {sum(lst)/len(lst)}")
print(f"Maximum of all values is {float(max(lst))}")
print(f"Minimum of all values is {float(min(lst))}")

```

**Output :**

```
List is: (1, 2, 3, 4, 5)
Sum of all values is 15.0
Average of all values is 3.0
Maximum of all values is 5.0
Minimum of all values is 1.0
```

**Lab: 11- EXTENDING AND EMBEDDING PYTHON**

**Practical 1:** Write a python program which invokes C function which does summation of all the values stored in 4X4 matrix.

**Source Code :**

```
# importing required modules
from ctypes import *
# connecting to shared file
so_file = "./my_functions.so"
my_functions = CDLL(so_file)
# initialising and printing the list
lst = [
    [1,2,3,4],
    [5,6,7,8],
    [9,10,11,12],
    [13,14,15,16]
]
print(f"List is: {lst}")

# converting list to array
arr = (c_int * 4 * 4)(* (tuple(i) for i in lst))

# calling c function from shared file
print(f"Sum of all list elements is: {my_functions.sum_matrix(arr)}")
```

**Output :**

```
List is: [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]]
Sum of all list elements is: 136
```