S.No	Date	Title	marks	sign
1	23/08/25	Introduction to python comands, script	15/15	
2.	30/07/25	implement conditional and looping statement	15	
3	06/08/25	importing and creating python mobile and Package in python program	15	
4.	13/08/25	use various datatypes list tuples and dictionary in Python programming	15	
5.	20/08/25	implement various searching and sorting operation in python	15	
6	03/08/25	utilizing functions concept's in python programming	15	
7.	10/09/25	Implement python various txt/csv file operation	15	
8	17/09/25	Implement Python generator and decorator	15	
9.	24/09/25	Implement Exception and Exceptional handling	15	
10	8/10/25	use matplotlib module for plotting in python.	15	
11	15/10/25	use Tkinter module for GUI design	15	
12.	15/10/25	simulate gaming connects using pygame.	15	

Completed

06/08/25

Task-3: Importing and creating python modules and packages in python program

Aim: To implement and demonstrate the process of importing built-in modules, creating user defined modules and organizing code into packages in python, thereby promoting code reusability modularity, and maintainability.

3.1 1) perform common math and random operations
2) work with the operating system (create / change directories, list contents) and read the python version
3) compute basic statistics (mean, median, mode, standard deviation)

Algorithm:

1) Import require modules: math, random, os, sys, statistics pathlib.
2) math & random:
- compute sqrt(5) radians(30), a random float in [0.0, 1.0]; factorial(5), gcd(5, 15), abs(-10), pow(3, 5), log base 3 of 2, log10(a) for a=100 and check: NaN / Infinity:
3) OS & sys:
- create C:\Python lab if not present and print current working directory.
- create C:\Python slot 24 if not present and change the current working directory to it.
- Print Python interpreter version
4) statistics
- on lists: [5, 6, 8, 10] and [2, 5, 3, 28, 3, 9, 4, 25, 6] compute mean, median, mode, stdev.
5) Print neatly formatted results.

(3) Python program:

```
# program to calculate total expenses of kaaran
# step1: Assign expenses
    books = 150
    groceries = 220
    transport = 90
# step2: calculate total
    total_expense = books + groceries + transport
# step3: display the result
    print("Total expenses incurred by kaaran = ₹ total_expense)
```

(4) sample input:

(values are already assigned in the programm. no manual input required)

Books = ₹150

Groceries = ₹220

Transport = ₹90

(5) sample output.

Total expenses incurred by kaaran = ₹460

Task-I Running python script and various expression in an interactive interpreter key terms covered:

Introduction to python, commands, scripts, Tags - easy [CO1, 5]

1-1 Karan spent ₹150 on books, ₹220 on groceries and ₹90 on transport, Help him calculate the total expenses.

Aim:- To write a python program that calculates the total amount spent by Karan on books groceries and transport.

1. start the program
2. Accept the amount spent on books, groceries and transport
3. calculate the total expenses by summing all three amounts
4. Display the total amount spent
5. End the programm.

Result: hence the python program for finding the area of traangle with all siter using heron's formula. is done executed successfully. books, grceries transport has been done sucessfully using python.

#
Python program:

```
#BMI calculator
#step1: Get input from the user.
weight = float(input("Enter your weight in kilograms:"))
height = float(input("Enter your height in meters:"))
#step2 : calculate BMI
bmi = weight / (height**2)
#step3: Display result
print("your Body mass Index (BMI) is:", round(bmi,2))
```

Input:

Enter your weight in kilo grams : 70

Enter your height in meters = 1.75

output:

your Body mass Index (BMI) is :22.86

1.2 write a BMI calculator Ask the user for weight (kg) and height (m) then calculate and display their BMI

① Aim:-

To write a python program that calculates and displays the body mass index (BMI) of a person using their weight (in kilograms) and heigh in meters)

1. start the program
2. prompt the user to input their weight in kilograms
3. Prompt the user to input their height in meters
4. calculate the BMI using the formula

$$BMI = \frac{weight}{height}$$

5. Display the calculated BMI
6. End the program.

Result: Hence the python program for calculating the Body main index (BMI) has been done executed successfully.

Python program:

```
import math
# step 1: Assign side lengths
a = 8
b = 6
c = 4
# step 2: calculate semi- perimeter
s = (a+b+c) / 2
# step 3: Apply Heron's formula
area = math.sqrt(s*(s-a)*(s-b)*(s-c))
# step 4: display result
print("The area of the triangle is:", round(area,
    "square cm").
```

Input:

side a = 8cm
side b = 6cm
side c = 4cm

output

The area of the triangle is 11.62 square cm

1.3

Maya wants to calculate the area of scalene triangle with sides of length 8cm, 6cm, and 4cm Help her write a python program that computes the area using Heron's formula.

AIM:-

To write a python program to find the area of a triangle when the lengths of all three sides are given using Heron's formula.

ALGORITHM:-

1) Start the program
2) Accept (or) assign the length of three sides:
   a, b and c
3) calculate the semi- perimeter:
   $s = \frac{a+b+c}{2}$
4) use Heron's formula to calculate the area:
   Area $= \sqrt{s(s-a)(s-b)(s-c)}$
5) display the area of the triangle
6) End the program:

EX NO.	1
PERFORMANCE (5)	5
RESULT AND ANALYSIS (3)	3
VIVA VOCE (3)	3
RECORD (4)	4
TOTAL (15)	15
SIGN WITH DATE	18/8/18

Result: Hence the python program for finding the area of triangle with our siter using heron's formula is done executed successfully

PROGRAM:

```
score = int(input("Enter the score"))
if score >=90:
    print("The Grade is A")
elif(score <=89 and score >=80):
    print("The Grade is B")
elif(score <=79 and score>=70):
    Printf("The Grade is C")
elif(score <=69 and score >=60):
    printf("The Grade is D")
else:
    printf("The Grade is F")
```

output:

```
Enter the score: 60
The Grade is D.
```

EX NO.	
1	PERFORMANCE (5)
2	RESULT AND ANALYSIS (3)
3	VIVA VOCE (3)
4	RECORD (4)
15	TOTAL (15)
16	SIGN WITH DATE

30/1/25

Task 2: Implement conditional, control and looping statement.

To implement conditional, control, and looping statements using Python.

2.1 you are developing a simple grade management system. for a school. The system needs to determine the grade of a student based on their score in a test. The grading system follow these rules:

If the score is 90 or above, the grade is "A".
If the score is between 80 and 89, the grade is "B"
If the score is between 70 and 79, the grade is "C"
If the score is between 60 and 69, the grade is "D"
If the score is below 60 and the grade is "F".

Algorithm:

1. start.
2. Get the input mark from the user.
3. with the use of an if-elif-else statement do
   - If the mark >=90 print grade "A"
   - If the mark is between 80 and 89 print grade "B"
   - If the mark is between 70 and 79 print grad "C"
   - If the mark is between 60 and 69 print grade "D"
   - If the mark is below is 60, print grade "F".
4. stop.

Python Program:

```
# Battery Health checker
Percentage = int(input("Enter battery percentage :"))
if percentage >= 90:
  print("Excellent Battery Health")
elif percentage >= 70:
  Print("Good Battery Health")
elif percetage >=40:
  Print("Average Battery Health")
else:
Print("Poor Battery Health")
```

Input

Battery charge percentage (integer)

Sample output:

Enter battery percentage: 85

Good Battery Health

2.2 The electronics maintanence team a data center needs a tool to asses the health status of ups back batteries based on their current change percentage. you are asked to develop a python input and categories the battery heath using the following condition

- If the percentage is greater than or equal to 90, display:
- Excellent Battery Health"
- If the percentage is between 70 and 89, display:
- "Good Battery Health"
- If the percentage is between 40 and 69, display:
- "Average Battery Health"
- If the percentage is below 40, display:
"poor battery Health"

Task:
write a python that wer ladderized if-elif-else statement

Algoridhm:
1. Accept battery percentage from the user
2. use ladderized if-elif-else to determine the health category
   - if percentage ≥ 90 → "Excellent Battery Health"
   - if 70 ≤ percentage ≥ 90 → "Good battery heath"
   - If 40 ≤ percentage < 70 → "Average Battery Health"
   - If percentage < 40 → "poor Battery Health"

Program.

```
for i in range(1,6):
height = int(input(f"Enter height of visitor {i} in cm:"))
if height >= 120:
  print("Allowed to ride.")
else:
print("Not allowed to ride.")
```

Sample input

Enter height of visitor 1 in cm: 130
Enter height of visitor 2 in cm: 110
Enter height of visitor 3 in cm: 150
Enter height of visitor 4 in cm: 90
Enter height of visitor 5 in cm: 125

Sample output:

Allowed
Not allowed
Allowed
Not allowed
Allowed

2. If you're coding a system of an amusement park that checks the height of each visitor.

- if the height 120cm or more, print "Allowed".
- otherwise, print "not allowed"

Repeat this for 5 visitors.

Algorithm:

1. start the program
2. set the total number of visitors to 5.
3. Loop from visitor 1 to vistor 5:
   - Accept the height of the visitor as input (in cm).
   - If height is greater than or equal to 120cm, print "Allowed".
   - Else, Print "not allowed."
4. End the loop after 5 visitors have been checked.
5. stop the program.

VEL TECH	
EX NO.	2
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	
SIGN WITH DATE	15

Result: Thus, the python program was successfully implemented using conditional statement (if -else) control flow and looping statements

```
Printf("current working directory", os.getcwd())
target_dir = path(r"C:\python slot 52.4")
target_dir.mkdir(parents=True, exist_ok=True)
os.chdir(target_dir)
Printf("changed into : {target_dir}")
Printf("Directory contents", os.listdir())
Print f("Pythonversion", sys.version)
Print("IN- STATISTICS")
data 1 = [5, 6, 8, 10]
data 2 = [2, 5, 3, 2, 8, 3, 9, 4, 2, 5, 6]
print (f"mean[data]) = " stats.mean(data 1)")
print[f"median [data 1]] = ", stats. median (dat 1))
print (f"mode (dat 2 ]) :", stats.mode (data 2))
Printf (f"stdev [data 2]] =", stats.stdev (data 2))
```

Expected sample output:

```
-- MATH &RANDOM --
sqrt(5) = 2.23606799774949
radians(30) = 0.52 35987755982 8298
random() in [0,1) = 0.3944887175646646 - will vary
radiant (2, 6) = 6
pi = 3.141592653589793
ceil (2.3) = 3  factorial (5) = 120  gcd (5, 15) = 5. abs (-10) = 10
pow (3,5) = 243,  log base 10 of 2 = 0.6309297535714574
log (0 (100) = 2.0
isinf(20) = True, isnan (naw) = True.
-- OS & SYS --
created/ ensured: c:\python slot
current working directory : C:\... (your current path)
created (ensured & changed into : c:\python slot 52.4
directory contents of c:\python slot 52.4 : []
```

Python version : 3xx (details)

--- STASTICS) .

mean [5,6,8,10] = 7.25

median [5,6,8,10] = 7.0

mode([2,5,3,2,8,3,9,4,2,5,6]) = 2

Stdev([2,5,3,2,8,3,9,4,2,5,6]) = 2.027156333833201093

Program:

cardfun

```
import random
def func():
cards = []
for i in range (1,53):
cards.append(i)
shuffled cards = random.sample(cards, k=52)
Print ("\n\n", shuffle- cards, "\n\n")
```

mymod.py

```
import cardfun
Card fun.func()
```

output:

Restart:

C:\Users\student.MAT-2VC683\AppData\Local\Programs\Python\Python311\Lib\site-packages\cardspace\mymod.py

[5, 24, 13, 22, 20, 41, 38, 51, 4, 7, 34, 49, 14, 50, 37, 40, 15, 35, 17, 18, 33, 39, 36, 42, 12, 6, 16, 19, 44, 48, 29, 29, 2, 27, 11, 31, 46, 28, 21, 32, 8, 25, 30, 22, 26, 10, 43, 47, 3, 44, 52, 1, 45, 9]

7.2

create a python package named cardpack. containing a module cardfun that imports the random module. Assign a range of cards, call a function from the module and display a random sample of cards.

Algorithm:

step1 :- start

step2 :- To create a package cardpack.

step3 :- To create a module cardfun and import random function

step4 :- Assign a cards range

step5 :- call a module function

step6 :- display the random sample cards.

step7 :- stop

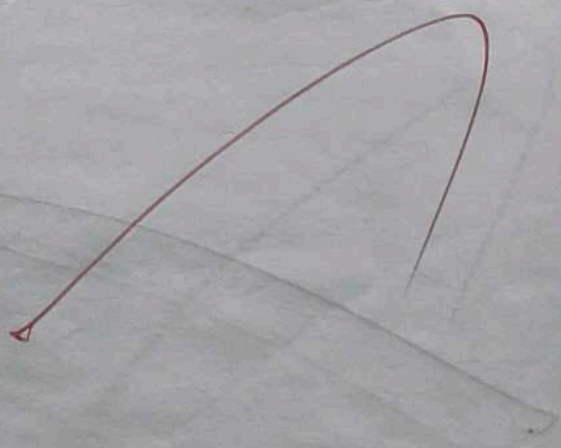
Program 1 : (mymath)

```
def add (a,b):
return a+b
def substration (a,b):
return a-b:
def multiply (a,b):
return a*b
def divide (a,b):
if b==0:
raise value Error ("cannot divide by zero")
return a/b
```

```
import mymath
a=10
b=5
print("Addition:", mymath.add (a,b))
print("subtraction:" mymath.subtract (a,b))
print ("multiplication:", mymath.multiply (a,b))
print (" division", mymath divide (a,b))
```

output :

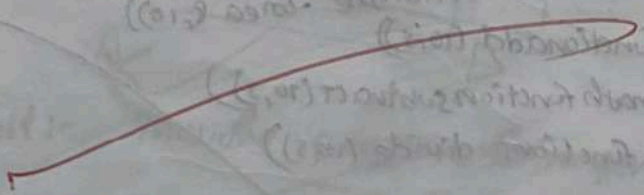
```
Addition : 15
subtraction: 5
multiplication: 50
Division : 2.0
```



7.3 You are tasked with developing a modular calculator application in Python. The calculator should support basic arithmetic operations, subtraction, addition, multiplication, and division. Each operation should be implemented in a seperate module. Additionally module and display results

Algorithm:

1. Define functions for addition, subtraction, multiplication and division
2. Handle division by zero by raising an error if the divisor is zero
3. Import the module. (mymath) containing these functions
4. Initialize two numbers (a=10, b=5)
5. call each function using mymath. <function-name> (a, b)
6. Print the results of all operations



4.2 output:

All lab slots : (9, 11, 14, 16, 14)

Is 14:00 present ?. True.

14:00 occurs 2 time(s)

First occurance position (1-based): 3

mornins slots : (9, 11)

Afternoon slots : (14, 16, 14)

```
Print ("Total:", tota)
print ("Average:", round (avg,2))
Print ("Best day:", best-day, "with", max-val)
print ("worst day:", worst-day "with", min-val)
```

4.2 Tuple- lab Time table

your department has a fixed daily Lab schedule represented by a. tuple of starting it appears using count(), find its first position using index(), and display morning and afternoon slots using slicing.

Aim: To manage and query an immutable daily lab slot schedule using a tuple, demonstrating membership checks, count(), index, and slicing.

Algorithm:

1. start
2. Define slots as a fixed tuple of integers
3. Read query hour.
4. check existence with query in slots.
5. use count(): if positive, use index() to find the first position.
6. slice into morning and afternoon.
7. Print results.
8. stop

Python program

```
# Tuple scenario
slots = (9, 11, 14, 16, 14) # immutable daily schedule. query = 14
exist = (query in slots)
freq = slots.count (query) # tuple.count()
first_pos = slots.index (query) if exists else "N/A" # tuple.index()
morning = slots [:2]
afternoon = slots [2:]
Print ("All lab slots:", slots)
Print (f"Is {query} : present?", exists)
Print (f"{query} : occurs ", freq, "times(s)")
print ("first occurance position (1-based):", first_pos)
Print ("morning slots :", morning)
print ("Afternoon slots:", afternoon)
```

4.3 output

Enter number of items in price list : 3
Enter item name : box
Enter price of box : 15
Enter item name : pen
Enter the price of pen : 10
Enter item name : pencil
Enter the price of pencil : 5
Enter item to update price (or press Enter to skip) : box
Enter new price for box : 20
Enter an item to remove from price list (or press Enter to skip) : pen
Available items : ['box', 'pencil']
prices : [20.0, 5.0]
Costliest item : box at 20.0
Remove 'pen' price (if existed) : 10.0

2025/2025

Task 5 - Implement various searching and sorting operations in python programming.

Aim :- To implement various searching and sorting operations in python programming.

5.1 A company stores employees records in a list of dictionaries, where each dictionary contains id, name and department.

Algorithm :

1. Input defination.
2. Define the function find_employee_by_id that takes two parameters.
a. list of dictionaries (employees) where each dictionary represents an employee. records with keys id, name and department.
b. An integeor (targed id), representing the employee ID to be searched
3. Iterate Thorough the list.
use a loop for to iterate through each dictionary in the employee list.
4. check for matching ID.
5. Return matching Record :
If a match is found, return the current dictionary
6. Hand no match :
If the loop completes without finding a match, return None.

Program 5.1

```
def find_employee_by_id (employees, target_id):
    for employee in employee :
    if employee ['id'] : target_id :
        return employee.
    return none.
# Test the function
    employees = [
{'id' : 1, 'name' : 'Alice', 'department' : 'HR'};
{'id' : 2, 'name' : 'Bob' , 'department' : 'Engineering'},
{'id' : 3, 'name' : 'Charlie', 'department' : 'sales' },
]
```

output:

{id : 2 'name': 'Bob' 'department': 'Engineering'}

Print (find_employee_by_id (employee, 2)) # output: {'id': 2, 'name':
'Bob', 'department': 'Engineering'}

5.2. you are developing a grade management system for a school. The system maintains a list of the student records.

Algorithm:

1. Initialization:
- Get the length of the students list and store it in n.
2. outer loop:
- Iterate from i = 0 to n-1 (inclusive). This loop represent the number of passes through the list.
3. Track swaps.
Initialize a boolean variable swapped to false.
4. Inner loop.
5. compare and swap:
- For each pair of adjacents elements (i.e student[j] and students [j+1]):
6. Early Termination
7. completion.

Program 5.2

```
def bubble_sort_scores (students):
    n = len(students)
    for i in range (n):
        # Track if any swap is made in this pass swapped = False.
        for j in range (0, n-i-1):
            if students [j] ['score'] > students [j+1] ['score']:
                # swap if the score of the current students is greater than the nex
                student[j], students [j+1] = students [j+1], students [j]
                swapped = True.
        # if no two elements were swapped, the list is already sorted
        if not swapped:
            break.
```

output:

Beforesorting:

{'name': 'Alice', 'score': 88}

{'name': 'Bob', 'score': 98}

{'name': 'Charlie', 'score': 75}

{'name': 'Diana', 'score': 85}

Aftersorting:

{'name': 'Charlie', 'score': 75}

{'name': 'Diana', 'score': 85}

{'name': 'Alice', 'score': 88}

{'name': 'Bob', 'score': 95}

```
# Example usage
students = [
 {'name': 'Alice', 'score': 88},
 {'name': 'Bob', 'score': 95},
 {'name': 'Charlie', 'score': 75},
 {'name': 'Diana', 'score': 85}
 ]
 print ("Before sorting:")
 for student in students
 print (student)
bubble_sort_scores(students)
 printf ("\n After sorting:")
 for student in students
  print (student)
```

VEL TECH	
EX NO.	
PERFORMANCE (5)	
RESULT AND ANALYSIS (5)	
VIVA VOCE (5)	
RECORD (5)	
(20)	
DATE	

Result: Thus, the program for various searching and sorting operations is executed and verified successfully.

output:

```
Welcome to the student grades analyzer.
Number of students: 4.
Type of student_names list: <class 'list'>.
Type of student_grades list: <class 'list'>.
Highest grade: 92
Lowest grade: 78
Sorted grades: [78, 85, 90, 92]
Reversed grades: [92, 90, 85, 78]
Grade indices from 1 to number_of_students: [1, 2, 3, 4]
```

Task.No: 6 utilizing "Functions" concepts in Python Programing

Date: 08/08/25

Aim:- To write the Python program using 'Functions' concepts in Python programming.

6.) you are developing a small python script to analyze and manipulate a list of student grades for a class project.

Algorithm:

1. start the program.
2. print a welcome message, outputs a simple greeting.
3. Determine and print the number of students:
4. Print the types of lists: use s type[] to show the type of student-name
5. Find and print highest and lowest grades: uses max[] and min[]
6. Printe sorted list of grades: uses sorted[] to sort the grades.
7. print reversed list of grades: uses reversed[] to reverse the sorted lists and converts it to a list.
8. Generate and print a range of grade indices:
9. stop.

Program:

```
def analyze_student_grades[]:
 #sample data.
 student_names ["Alice","Rob","Charlie","Diana"]
 student_grades [85, 92,78,90]
# 1. print a welcome message
 Print ("welcome to the student Grades Analyzer!\n")
# 2. Determine and print the number of students
 num_students: len[student_names]
 print ("number of students:" num_students]
# print the type of student names list and the grade list.
  print ("Type of student_grades list:", type (student_grades)]
```

Find and print the highest and lowest grade ~~highest grade~~

highest_grade = max (student_grades)

lowest_grade = min (student_grades)

Print ("In. Highest grades:", highest_grade)

Print ("lowest grades", lowest_grades)

5. print the list of grades sorted in ascending order.

sorted_grades = sorted (student_grades)

Print("In sorted grades:", sorted_grades)

6. print the list of grades in reverse order

reversed_grades = list (reversed (sorted_grades))

Print ("Reversed grades:", reversed_grades))

7. Generate and print a range of grades indices from 1 to the number of students

grade_indices: list (range (1, num_students +1))

~~Print ("Reversed grades:", reversed_grades)~~

Print ("In Grade indices from 1 to number of students:", grade_indices)

Run the analysis

analyze_student_grades()

2. you are tasked with creating a small calculator application to help us perform basic arithmetic operations

Algorithm:

1. start the program
2. user input for numbers: The program prompts the user to enter two numbers.
3. user input for operation: The program prompts the user to choose an arithmetic open. (addition, subtraction, multiplication, division).

5. Display Result: The programs displays the results of the operation.

6. Stop.

7.2 program:

```
def add(a,b):
    """Return the sum of two numbers."""
    return a+b
def subtract(a,b):
    """Return the difference between two numbers."""
    return a-b
def multiply(a,b):
    """Return the quotient of two numbers. Handles division
    by zero."""
    if b!=0:
        return a/b
    else:
        return "Error: Division by zero"
def greet(name):
    """Return a greeting message for the user."""
    return f"Hello, {name}! welcome to the program."
def main():
    # Demonstrating the use of user-defined functions
    # Arithmetic operations
    num1 = 10
    num2 = 5
    print("Arithmetic operations:")
    print(f"Sum of {num1} and {num2}:", add(num1, num2))
    print(f"Difference between {num1} and {num2}:", subtract
        (num1, num2))
    print(f"Product of {num1} and {num2}:", multiply(num1, num2))
    print(f"Quotient of {num1} and {num2}:", divide(num1, num2)
# Greeting the user.
```

Output:

Arithmetic operations:

sum of 10 and 5: 15

Difference between 10 and 5: 5

Product of 10 and 5: 50

Quotient of 10 and 5: 2.0

Greeting:

Hello, Alice! welcome to the program

~~#Greeting~~

```
# Greeting the user.
user_name = "Alice"
Print ("In Greeting :")
Print (great (user_name))
# Run the main function
if __name__ == "__main__":
  main()
```

VEL TECH	
EX NO.	6
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	
SIGNATURE WITH DATE	15

Result:- Thus, the python program using 'function' concepts was successfully executed and the output was verified.

Task 7) Implement various txt / csv file operations

Date: 10/09/25

Aim:- To write a python program for creating and updating student registration details using txt file operation.

Algorithm:

step1 : start

step2 : using open() method, create and write text file 'myfile.txt' with student details.

step3 : update the new registered student details using append operation in it

step4 : open the file in read mode and using read() method print the student details

step5 : using seek tell print the particular student record.

step6 : using tell method print the current position of the file.

step7 : close the file

step8 : stop.

Program :

```
file = open("student1.txt","w")
input1 = input("Enter column name\n")
file.write(input1)
file.writ("\n")
n=int(input("Enter the no.of students"))
for in range(0,n):
  input2 = input("Enter student details with for new")
  file.write(input2)
  file.write("\n")
file=open("student1.txt","a")
input3 = input("Enter updated student details\n")
file.write(input3)
file = open("student1.txt","a")
input 3= input("Enter updated students details\n")
file.write(input3)
```

output:

Student details using ~~fixed~~ Read function is

VTU NO	NAME	AGE
2305	RAM	21
1920	SHIVA	21
2305	RAM	20
1920	SHIVA	21

output of Read line function is

2305 RAM 20

"find the current position of the file pointer:

29

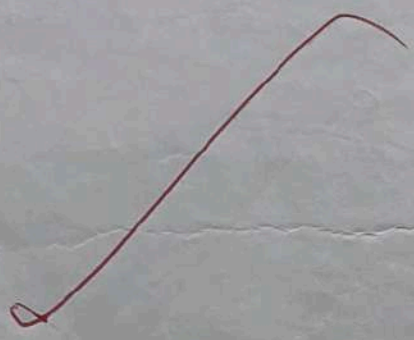
```
file = open("student 1.txt","r")
print("student details using Read function is:")
print(file.read())
print("\n")
file.seek(0)
print("The length of first line is:")
line = file.readline()
len = len(line)
print(len)
file.seek(len+1)
print("output of Readline (first student record) function is:")
print("file.readline()")
print("\n find the current position of file pointer:")
f=file.tell()
print(f)
file.close()
```

Result: Thus, the python program for creating, and updating student registration details using txt file operations was executed successfully.

output:

Filename: merge.txt

output: 5, 4, 8, 4.



Task-2 counting cases.

counting a python program whose file name is "merge.txt". To illustrate the below context inside of the file.

Python is a high level language developed by Guido van Rossum in 1991.

count the total number of uppercase, lower case, and digits used in the text file "merge.txt".

Input:

File name: merge.txt

output: 5, 48, 4.

```
#program to count uppercase, lowercase, and digits in a file (merge.txt)
#step1: create and write content to the file.
with open("merge.txt", "w") as f:
    f.write("Python is a High level language, developed by Guido van
            Rossum in 1991")
#step2: open the file for reading
with open("merge.txt", "r") as f:
    text = f.read()
#step3: initialize counters
    upper_count = 0
    lower_count = 0
    digit_count = 0
#step4: count uppercase, lowercase and digits
    for char in text:
    if char.isupper():
        upper_count += 1
    elif char.islower():
        lowercount += 1
    elif char.isdigit():
        digit_count += 1
#step5: print the result.
print("uppercase letters:", upper_count)
print("lowercase letters:", lower_count)
print("Digits:", digit_count)
# compact output as required
print(f"{upper_count}, {lower_count}, {digit_count}")
```

output:

Saurav - 7169.0

Abinav - 7178.0

Hasvad - 7 1520.0

Ravi - 7188.0

construct a python program to read the above table of student grades for each student print out the result as students average grade using another text file (result.txt)

program:

```
#program to read student's grades from a file, calculate averages, save results.
#step1: Read input data from grades.txt
with open("grades.txt","r") as f:
  lines = f.readlines()
#step2: Extract number of students
n = int(lines[0].strip())
#step3: Extract weights
weights = lines[1].strip().split()
weights = [float(w) for w in weights]
#step4: process each student is data
students = []
for i in range(2,2+n):
  parts = lines[i].strip().split()
  names = parts[0]
  marks = [int(m) for m in parts[1:]]
#calculate weighted average.
total = 0
for j in range(4):
  total = total + marks[j] * weights[j]
  student.append((name, round(total,2)))
#step5: write results into results.txt
with open("results.txt","w") as f:
  for name, avg in students:
  f.write(name+":"-7" +str(avg)+"\n")
print("Average grades have been written to results.txt")
```

VEL TECH	
EX NO.	7
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	
SIGN WITH DATE	15

Result: Thus the python program to simple read various text file operations was successfully executed and output was verified

output:

Enter the starting number : 1
Enter the ending number : 50
Enter the stepvalue : 5

1
6
11
16
21
26
31
36
41
46

Date: Week 5 Task 8: Implement Python generator and decorator.

Aim: write a python program to implement python generator and decorators.

8.1: write a python program that include a generator function to produce a sequence of numbers The generator should be called.

a. produce a sequence of number. when provided start, end, and step values

~~b. produce~~ a.

Algorithm:

1. Define generator function
   - Define to function number - sequence (start, end, step=1)
2. Initialize current value:
   - set to the value start
3. Create, generator object:
   create generator. object by calling. number_sequence (start, end, step) without providing values
4. print generated sequence:
   - Iterate over the values produced by the generator object.
   - print each value.

8.1 program:

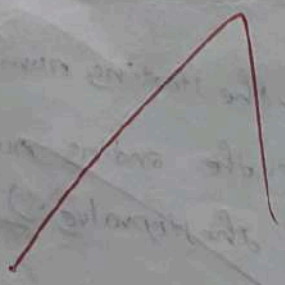
```
def number_sequence (start, end, step=1):
    current = start
    while current <= end:
        yield current
        current += step
start = int(input("Enter the starting number:"))
end = int(input("Enter the ending number:"))
step = int(input("Enter the step value:"))
# create the generator
sequence_generator = number_sequence(start, end, step)
# print the generated sequence of numbers
for number in sequence_generator:
    print(number)
```

output:

0

1

2



Produce a default sequence of numbers starting from 0, ending at 10 with step 1, if no values are provided

Algorithm:

1. Start function
   - Define the function my_generator
2. Initialize counter:
   set value to 0.
3. Generate value,
   while value is less than n,
4. create Generator object.
5. Iterate and print values

8.1 (b) Program.

```
def my_generator(n):
#intidize courtier
 value=0
#loop until the counter is less than n.
while value<n:
#produce the current value of the counter yield value
#increment the counter
   value=1
#iterate over the generator object produced by my_generator.
 for value in mygenerator():
    #print each value produced by generator. print(value)
```

8.2 imagine you are working on a messaging application that needs to format messages differently based on the users preferences. We can choose to have their messages

Algorith:

1. create decorators:
   . define uppercase_decorator to convert the result of a function.
2. Define functions:
   . define shout function to retrieve input.
3. Define Greet function:
   . define greet function what
4. Execute the program.
   . call greet (shout) to print the greeting in uppercase.

output:

HI, IAM CREATED BY A FUNCTION PASSED AS AN ARGUMENT

hi, iam created by a function passed as an argument

Program:

```
def uppercase_decorator(func):
    def wrapper(text):
        return func(text).upper()
    return wrapper
def lowercase_decorator(func):
    def wrapper(text):
        return func(text).lower()
    return wrapper
@uppercase_decorator
def shout(text):
    return text
@lowercase_decorator
def whisper(text):
    return text
def greet(func):
    greeting = func("Hi, I am created by a function passed as an argument")
    print(greeting)
greet(shout)
greet(whisper)
```

VEL TECH	
EX NO.	8
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	15
SIGN WITH DATE	

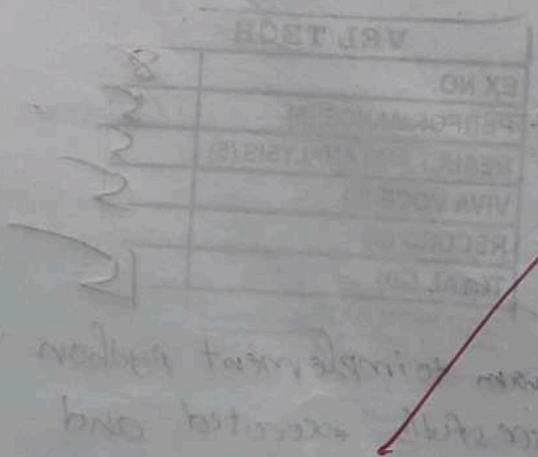
Result: Thus the python program to implement python generator and decorators was succesfully executed and the output was verified.

output:

Gradeslist: [85, 90, 98, 92, 88]

Enter the index of the grade you want to view: 10

Invalid index. Please enter a valid index.



Date: 24/09/25

Task 9: Implement Exception and Exceptional handling in python.

Aim: To implement Exceptions and Exceptional handling in python

Algorithm:

1. Start the program
2. Initializes a list of grades (eg [85,90,78,92,88]).
3. Prompts the user to enter the index of the grade.
4. Attempts to display the grade at the specified index.
5. If the index is out of range, catch the index error.

Programm:

```
# initialize the list of grades
grades = [85,90,78,92,88]
# display the grades list
print ("Grades list:", grades)
# prompt the user to enter the index of the grade they want to view
try:
    index = int(input("Enter the index of the grade you want to view:"))
    # Attempt to display the grade at specific index.
# Handle the case where the index is out of range.
# print ("Invalid index. please enter a valid range")
    Print ("Invalid index. please enter a valid index.")
# Handle the case where the inputs is not an integer
    Print ("Invalid input. please enter a numerical index.")
```

output:

Enter the numerator : 10

Enter the denominator : 0

ERROR!

Error: division by zero is not allowed

Task 9.2 you are developing python calculator program, that performs basic arithmetic operations.

Algorithm:

1. start the program
2. prompts the user to enter two numbers.
3. Attempts to divide the numerator by the denominator.
4. If the denominator is zero, catches the zero division error.

Program:

```
# function to perform division
def divide_numbers():
    try:
        # prompt the user. to enter the numerator.
        numerator = float(input("Enter the numerator:"))
        # prompt the user to enter denominator.
        denominator = float(input("Enter the denominator:"))
        # Attempt to perform division
        result = numerator / denominator
        print("Result: {result}")
    except zero division Error:
        # Handle division by zero error.
        print("Error: division by zero ; not allowed")
        # Handle invalid input that is not a number
        print("Error: please enter valid numbers.")
# call the function to execute the division operation
divide_numbers()
```

output:

Enter a number : 15

Exception occured: invalidage

Task 9.3

You are building a python application to determine if a person is eligible to vote based on their age. According to the rules, only individuals who are 18 years.

Algorithm:

1. Define the custom exception
2. prompt the user for input.
3. check if age is below 18.
4. Raise an exception if the condition is met.
5. Handle the exception with a custom error message.

Program:

```
#define python user-defined exceptions
class invalid Age Exception (Exception):
  "raised where input value is less than 18" pass
# you need to guess this number.
try:
input_num = int (input("Enter a number:"))
  if input_num < numerator:
    raise invalid Age Exception
  else:
  print ("Eligible to vote")
  except Invalid Age Exception:
    print ("Exception occured : Invalid Age")
```

VEL TECH	
EX NO.	9
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	15

Result: Thus the program for implement exceptions and Exceptional handlings is executed and verified successfully.

Date: 8/10/25 Task 10: use mata plotlib module for plotting in python

Aim: To use matplotlib module for plotting in python

Problem 10:1: write a python programming to display a bar chart of popularity of programming languages

sample data:

Programming languages: Java, python, PHP, Javascript, c#, c++

sample output:

Algorithm:

1. Define two lists of programming languages and their popularity respectively
2. Find the maximum popularity value in the list.
2. Define a scaling factor to scale the bar height within certain limits
4. For each language and popularity pair.

Program:

```
#pip install matplotlib
import matplotlib.pyplot as plt
languages = ['Java', 'Python', 'PHP', 'Javascript', 'C#', 'C++']
Popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]
plt.bar = [languages, popularity, color = 'b']
plt.title = ('popularity of programming Languages')
plt.ylabel ('popularity')
plt.show()
```

10.2: Write a python programming to create a pie chart of the popularity of programming languages

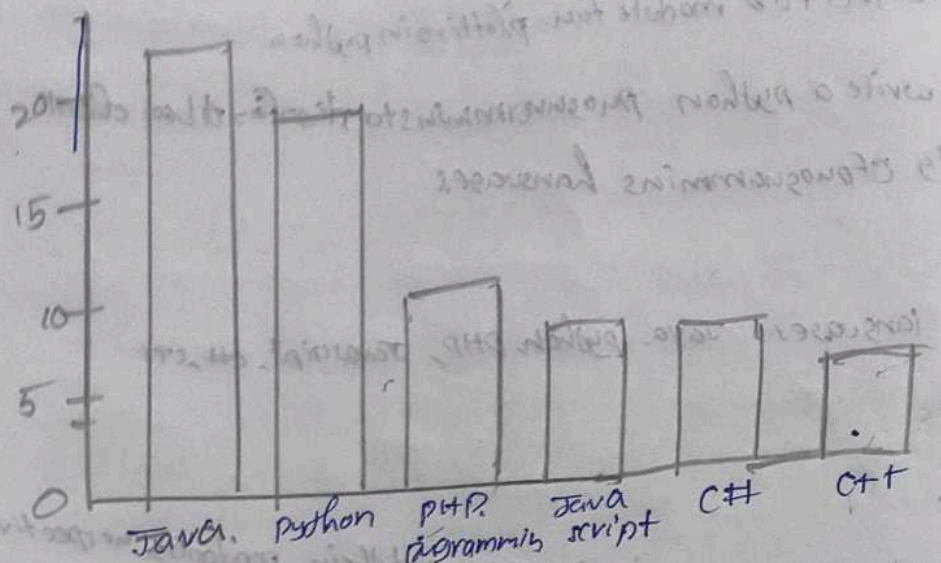
sample data:

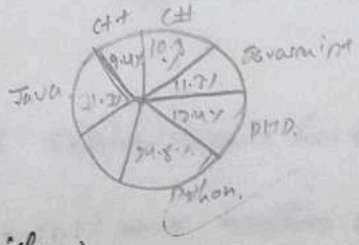
programming languages: Java, python, PHP, javascript, C#, C++.

popularity: 22.2, 17.6, 8.8, 8, 7.7, 6.7.

sample output:

output:





Algorithm:

1. create a list of programming languages and popularity.
2. create a pie chart using the matplotlib library.
3. set the title and legend for the pie chart.
4. show the pie chart

Program:

```
Import matplotlib.pyplot as plt
#step1
languages = ['Java', 'python', 'PHP', 'Javascript', 'C#', 'C++']
popularity = [22.2, 17.6, 8.8, 7.7, 6.7]
#step2
plt.pie (popularity, labels = languages, autopct = '%1.1f%%']
#step3
plt.title (popularity of programming languages)
plt. legend (languages, loc 'best')
#step4
plt.show()
```

VEL TECH	
EX NO.	10
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
T. (20)	15
SIGNATURE	

Result: Thus, the python program use matplotlib module for plotting is executed and verified successfull.

output :

Hello, world!

Date: 15/10/25 Task 11. Use Tkinter module for UI design.

Aim:

To use Tkinter module for UI design.

Problem 11.1: Write a python GUI program to create a label and change the label font style using tkinter module.

Hello

Algorithm:

1. Import tkinter module
2. Create a main window
3. Create a label with desired text
4. Add the label to the main window using pack() method
5. Add the button to the main window using pack()
6. Start the main loop

Program:

```
import tkinter as tk
#Function to change font style.
def change_font():
    label.config(font=("Arial", 18, "bold"))
#Create main window
root = tk.Tk()
#Create label with desired text
label = tk.Label(root, text="Hello, world!", font=("Helvetica", 14))
# Add Label to main window
label.pack()
#Create button to change font style.
button = tk.Button(root, text="change font", command=change_font)
# Add button to main window
button.pack()
#Start the main loop
root.mainloop()
```

Task11.2 : write a python GUI program to create three single text-box to accept a value from the user using tkinter module.

Name
DevID
password

Algorithm:

1. Import the tkinter module.
2. Create the main window.
3. Add labels and text boxes to the main window
4. create a button to submit
5. Get the values
6. Close the main window when the button is closed.

Program:

```
import tkinter as tk.
#create the main window
root = tk.Tk()
root.title("Text Box Input")
#create labels and text-boxes
label1 = tk.label(root, text = "Enter value 1:")
entry1 = tk.Entry(root)
label2 = tk.label(root, text = "Enter value 2:")
entry2 = tk.Entry(root)
label3 = tk.label(root, text = "Enter value 3:")
entry3 = tk.entry(root)
#set the size of the text boxes
entry1.config(width = 20)
entry2.config(width = 20)
entry3.config(width = 20)
#create a function to set the values entered in the text-boxes
def get_values():
    val1 = entry1.get()
    val2 = entry2.get()
    val3 = entry3.get()
    print("value1:", val1)
    print("value2:", val2)
    print("value3:", val3)
```

output:

Enter value 1:

Enter value 2:

Enter value 3:

submit

```
#create a button to submit the values entered into text boxes
submit_button = tk.Button(root, text="Submit", command=get_values)
#Add the labels, text boxes, and button to the main window.
label1.pack()
entry1.pack()
label2.pack()
entry2.pack()
label3.pack()
entry3.pack()
submit_button.pack()
#Run the main event loop.
root.mainloop()
```

VEL TECH	
EX NO.	11
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	15
SIGN WITH DATE	

Result: Thus the program using Tkinter module for UI design was executed and verified successfully.

Date: 15/10/25 Task 12 - simulate Gaming concepts using pygame

Aim: To simulate Gaming concepts using Pygame

Snakegame:

Problem: write a python program to create a snake game using pygame packages.

conditions:

1. set the window size
2. creat a snake
3. make the snake
4. make the snake to move in the directions when left right down and UP keys is passed
5. If the snake hits the window. Game over.

Algorithm:

1. Import pygame package and initialize it.
2. Define window size and title.
3. create a snake which initializes the snake position and keyboard movement.
4. Create a function to check if the snake collides with the window and end of the game.
5. End of the game if user quits or snake collides with the window.

Program:

#importing libraries

```
import pygame
import time
import random
snake_speed = 15
# window size
window_x = 720
window_y = 480
# defining colors
black = pygame.Color(0, 0, 0)
white = pygame.Color(255, 255, 255)
red = pygame.Color(255, 0, 0)
green = pygame.Color(0, 255, 0)
blue = pygame.Color(0, 0, 255)
# initialising pygame
pygame.init()
# initialize game window
pygame.display.set_caption('Snakes for Snakes')
game_window = pygame.display.set_mode((window_x, window_y))
```

```
#FPS (frames per second) controller fps = pygame.time.Clock()
#defining snake default position
snake_position = [100,50]
#defining first 4 blocks of snake body
snake_body = [[100,50],
              [90,50],
              [80,50],
              [70,50]]
#fruit position
fruit_position = [random.randrange(1,(window_x//10))*10, random.randrange
    (1,(window_y//10))*10]
fruit_spawn = True
#setting default snake direction towards
#right
direction = 'right'
change_to = direction
#initial score
score = 0
#displaying score function
def show_score(choice, color, font, size):
#creating font object score_font
# score_surface
#displaying text
# game over function
def game_over():
#creating font object my_font
my_font = pygame.font.SysFont('times new roman', 50)
#creating a text surface on which text
#will be drawn
game_over_surface = my_font.render('Your Score is : '+str(score), True, red)
#create a rectangular object for the text
#surface object
game_over_rect = game_over_surface.get_rect()
#blit will draw the text on screen
game_window.blit(game_over_surface, game_over_rect)
pygame.display.flip()
# after 2 seconds we will quit the program
time.sleep(2)
#deactivating pygame libraries
#quit the program quit()
```

```
#main function
while True:
#handling key events
for event in pygame.event.get()
 if event.type = pygame.KEYDOWN:
  if event.key = pygame.K_UP:
   change_to = 'UP'
  if event.key = 'pygame.K_DOWN:
   change_to = 'DOWN'
  if event.key = pygame.K_RIGHT:
   change_to = 'RIGHT'

# If two keys pressed simultaneously
# We don't want snake to move into two
# directions simultaneously
 if change_to = 'UP' and direction != 'DOWN':
   direction = 'UP'
 if change_to = 'DOWN' and direction != 'UP':
   direction = 'LEFT'

# moving the snake
 if direction = 'UP':
   snake_position[1] -= 10
 if direction = 'DOWN':
   snake_position[1] += 10
 if direction = 'LEFT'
   snake_position[0] -= 10
 if direction = 'RIGHT':
   snake_position[0] += 10

# snake body growing mechanism
# if fruits and snakes collide then scores
# will be incremented by 10
 snake_body.insert(0, list(snake_position))
 if snake_position[0] == fruit_position and snake_position[1] == fruit_position[1]
   score += 10
   fruit_spawn = False
 else:
  snake_body.pop()
 if not fruit_spawn:
```

output:

score : 0

```
fruit - position [random. randrange (1, (window_x // 10) * 10,
    random. randrange (1, (window-y // 10)) * 10]

fruit-spawn = True.
game_window.fill (black)
pygame. draw. rect (game_window, white, pygame. Rect (fruit_position[1], 10, 10))

# Game over conditions
if snake_position [0] < 0 or snake_position [0] > window - x - 10:
    game_over()
if snake_position [1] < 0 or snake_position [1] > window - y - 10:
    game_over()
# Touching the snake body
for block in snak_body. [1 : ]:
    if snake_position [0] = block [0] and snake_position [1] = = block[1] :
    game over()
# display score continuously
show_score (1, white 'times new roman', 20)
# Refresh. game screen
pygame. display. update ()
# Frame per second / Refresh Rate
fps. tick. (snake. speed)
```

problem 2: write a python program to develop. achess board using pygame.

Algorithm:

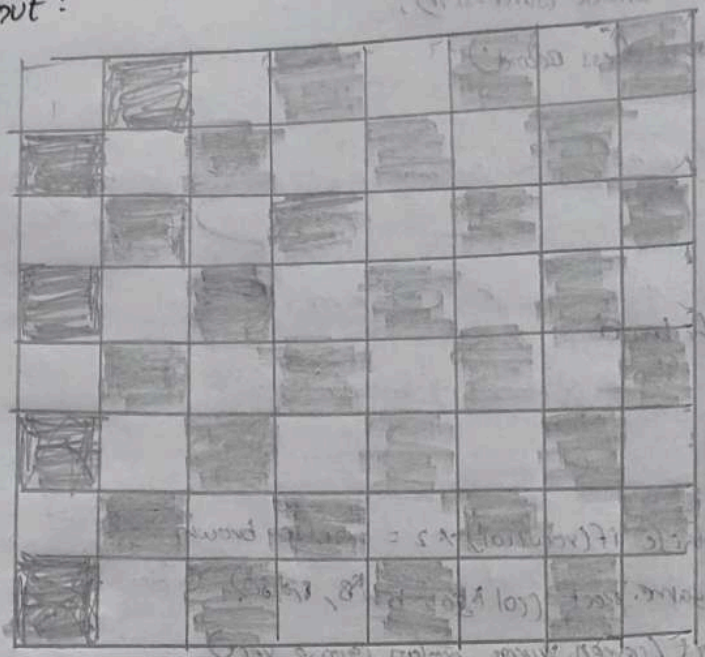
1. Import pygame and initialize it.
2. set screen. size and title
3. Define colors for the board and pieces
4. Define a function to draw the board by loading image.
5. Define the intial. state
6. Draw the board and pieces on the screen
7. start the game loop.

Program:

```
import pygame
# intialize pygame
Pygame. init()
# set screen size and title
```

```
screen_size = (640, 640)
screen = pygame.display.set_mode(screen_size)
pygame.display.set_caption('chess board')
#define colours
black = (0,0,0)
white = (255,255,255)
brown = (153, 76, 0)
#define function to draw the board
def draw_board():
    for row in range(8):
        for col in range(8):
            square_colour = white if (row+col) % 2 == 0 else brown
            square_rect = pygame.Rect(col*80, row*80, 80, 80)
            pygame.draw.rect(screen, square_colour, square_rect)
#define function to draw the pieces.
def draw_board():
    piece_images = {
        'r': pygame.image.load('images/rook.png'),
        'n': pygame.image.load('image1/knight.png')
        'b': pygame.image.load('image1/kins..png')
    }
    for row in range(8):
        for col in range(8):
            piece = board[row][col]
            if piece != '.':
                piece_image = piece_images[piece]
                piece_rect = pygame.Rect(col*80, row*80, 80, 80)
                screen.blit(piece_image, piece_rect)
#define intial state of the board.
board = [['r', 'n', 'b', 'q', 'k', 'b', 'n', 'r'],
         ['p', 'p', 'p', 'p', 'p', 'p', 'p', 'p'],
         ['.', '.', '.', '.', '.', '.', '.', '.'],
         ['.', '.', '.', '.', '.', '.', '.', '.'],
         ['.', '.', '.', '.', '.', '.', '.', '.'],
         ['.', '.', '.', '.', '.', '.', '.', '.'],
```

output :



```
['P', 'P', 'P', 'P', 'P', 'P', 'P', 'P']
['R', 'N', 'B', 'Q', 'K', 'B', 'N', 'R']
]
# draw the board and pieces
draw_board()
draw_pieces(board)
# start the game
while True:
    for event in pygame.get():
    if event.type == pygame.QUIT:
        pygame.quit()
    pygame.display.update()
```

VEL TECH	
EX NO.	12
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	
SIGN WITH DATE	15

Result: Thus the program for pygame is executed and verified successfully.

completed