INDEX

Navdurga

Name : R. Ramesh Std. : Sec. :

Roll No. : 55 Sub :

S.No.	Date	Title	Page No.	Teacher's Sign / Remarks
1	05/08/25	INTRODUCTION TO PYTHON COMMANDS SCRIPT	01	
2	30/08/25	Implement conditional and looping statements	15	
3	06/08/25	Importing and creating python modules	15	
4	13/08/25	Use various datatypes like tuple and dictionary in python programming.	15	
5.	20/08/25	Implement and various learning and sorting operations in python	15	
6.	03/08/25	Utilizing Functions concepts in python programming.	15	
7.	10/09/25	Implement various text file operations.	15	
8	17/09/25	Implementation of python generators AND decorators.	15	
9.	24/09/25	you are developing a python program that parses a list of grades of students	15	
10	8/10/25	Use matplotlib module for plotting in python	15	
11.	15/10/25	Use Tkinter module for UI design	15	Completed
12	15/10/25	write a python program to develop a Chessboard using Pro grame.	15	

* Python Program

```
# Program to calculate total expenses of Karan
# step 1: Assign expenses
        books = 150
        groceries = 220
        transport = 90

# step 2: Calculate total
    total_expense = books + groceries + transport
# step 3: Display the result
    Print ("total expenses incurred by Karan:
                                ₹" total expense)
```

Sample Input

Values are already assigned in the program
no manual input required.

Books = ₹150

groceries = ₹ 220

Transport = ₹ 90

sample output:-

Total expenses incurred by Karan = ₹ 460

05/8/2019

Task-1 running python script and Moodovc expressions in an interactive interpreter key terms covered: Introduction to python commands, scripts, Task - daily costs: Karan spent ₹150 on books ₹220 on groceries and ₹70 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

Algorithm :-

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 program.

Result :- Thus the amount spent by karan on books groceries and transport are proved.

Python program

```
# BMI calculator
# Step-1: Get input from the user
weight = float (input ("enter your weight in kilograms:"))
height = float (input ("enter your height in meters:"))
# Step 2: Calculate BMI
    bmi = weight / (height**2)
# Step3 : Display result
    print f("your body mass Index (BMI) is: {bmi:.2f}")
```

round (bmi, 2)

Sample Input:-

Enter your weight in kilograms : 70
Enter your height in meters : 1.75

Your body mass Index (BMI) is : 22.86

11. Write a BMI calculation for the user their weight (kg) and height (m) then calculate and display their BMI

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

Algorithm :-

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 BMI using the formula

$$BMI = \frac{weight}{height^2}$$

5. Display the calculated BMI
6. End the program

Result:- Thus the body mass index of a person using their weight (kg) and height (m) are found

* Python program:-

```
Import math

# step-1 : Assign side length

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 (area2) "square cm")
```

Sample Input:-

values are already assigned.

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

Sample output:

The area of the triangle is : 11.62 square cm

1.3 Jaya wants to calculate the area of a 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 length of all three sides are given using Heron formula.

Algorithm:-

1. Start the program
2. Accept or assign the length of the three sides a, b and c.
3. Calculate the semi perimeter

$$s = \frac{a+b+c}{2}$$

4. Using Heron's formula calculate area

$$\text{Area} = \sqrt{s(s-a)(s-b)(s-c)}$$

5. Display the area of the triangle
6. End the program

VEL TECH - CSE	
NO	1
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	3
VIVA VOCE (5)	3
RECORD (5)	4
TOTAL (15)	15
SIGN WITH DATE	

Result Thus the area of triangle when the length of all three sides are proved by Heron's formula

Program

```
Score = Input(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):
    print("The grade is C")
elif (score <=69 and score. >= 60):
    print ("The grade is D")
else:
    print ("The Grade is F")
```

OUTPUT:-

Enter the scores

The grade is D

30/07/25

TASK 2. IMPLEMENT CONDITIONAL AND LOOPING STATEMENTS

Aim:- 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 follows 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, 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 marks >=90 print grade "A"
   - If the mark is between 80 and 89 print grade "B"
   - If the mark is between 70 and 79 print grade "C"
   - If the mark is between 60 and 69 print grade "D"
   - If the mark is below 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 percentage >= 40:
    print("Average Battery health")
else:
    print("Poor Battery health")
```

Input :-

charge
Battery_percentage (Integer)

Output :-

Enter the battery percentage: 85
good battery health

2.2 The electronics maintenance team at a data center needs a tool to alert the health status of UPS backup batteries based on current their charge percentage python program accepts the battery charge percentage as input and categorizes the battery health using the following conditions:

- If the percentage is greater than or equal to 90, display
  > "Excellent Battery Health"
- If the percentage is greater than 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 program that uses laddered if-elif-else statements.

Algorithm:-

1. Accept battery percentage from the user.
2. Use laddered if-else to determine the health category.
   - If percentage ≥ 90 → "Excellent Battery Health"
   - If 70 ≤ percentage < 90 → "Good battery Health"
   - If 40 ≤ percentage < 70 → "Average battery health"
   - If percentage < 40 → "Poor Battery Health"

Program:

```
for p in range (1,6):
    height = int (input ("enter height of visitor"))
    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.

2.3 You're coding a system at an amusement park that checks the height of each visitor.

- If the height is 120 cm 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 visitor 5:
   - Accept the height of the visitor as input (from)
   - If height is greater than or equal to 120, 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)	5
	15

RESULT:- Thus the python program was successfully implemented using conditional statements (if-else), CONTROL FLOW, and LOOPING STATEMENTS.

PROGRAM :-

```
import math
import random
import os
import sys
import statistics as stats
from pathlib import Path.

print ("\n--- MATH & RANDOM ---")
print ("sqrt(5) =", Math.sqrt(5))
print ("radians(30) =", math.radians(30))
print ("random() in [0,1) =", random.random())
print ("randint(2,6) =", random.radiant(2,6))
print ("pi =", math.pi)
print ("ceil(2.3) =" math.ceil(2.3))
print ("floor(2.3) =", math.floor(2.3))
print ("factorial(5) =" math.factorial(5))
print ("gcd(5,15) =" math.gcd(5,15))
print ("abs(-10) =", abs(-10))
print ("pow(3,5) =", pow(3,5))
print ("log base 3 of 2 =", math.log(2,3))
a_val = 100
print (f"log10({a_val}) =" math.log10(a_val))
int_val = float('int')
nan_val = float('nan')
print(f"isinf(inf) = {math.isinf(int_val)}, isnan(NAN) = {math.isnan(nan_val)}")
print("\n--- os & sys ---")
Path_pythonlab = Path(r"C:\Pythonlab")
Path_Pythonlab = Path(r"
Path = pythonlab.mkdir(parents=True, exist_ok=
print(f"created/ensured: {Path_pythonlab}")
print("current working directory:", os.getcwd())
target_dir = Path(r"C:\Python lot\RL4")
target_dir.mkdir(parents=True, exist_ok=True)
os.chdir(target_dir)
print(f"changed into: {target_dir}")
print("Directory contents:")
print("Python version:", os.listdir())
```

06/08/25

TASK 3: IMPORTING AND CREATING PYTHON MODULES AND PACKAGE IN PHYTHON PROGRAM.

Aim:-

To implement and demonitiate 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) compare basic stattics (mean, median, mode standard deviation.)

Algorithm:-

1) Import required modules: math, random, os, sys, statistics, pathlib
2) Math & Random.
3) compute: sqrt(5) radians(30), a random float in[0.0,10) a random integer in [2,16] (inclusive) ti, ceil(2.3), floor(2.3), factorial(5), gcd(5,15) abs(-10), pow(3,5), log base 3 of 2

3) OS & sys:
   - create c:\python lab if not present and print the current working directory
   - create c:\python lab 2.4 if not present and change the current working directly to it
   - list all files [directories in the new current directory

data 1 = [5, 6, 8, 10]

data 2 = [2, 5, 3, 2, 8, 3, 9, 4, 2, 5, 6]

Print (f"mean ({data 1}) =", stats. mean (data
print (f"median ({data 2}) =", stats. median (data
print (f"mode ({data 2}) =", stat. mode (data
print (f"stdev ({data 2}) =", stat. stdev (data

EXPECTED SAMPLE OUT PUT

--- MATH & RANDOM ---

sqrt (5) 2.23606797749979.

radians (90) 0.5 23598775598 2988.

random () in [0, 1) = 0.37444887175646646 <-

vary <- Intuitive: will vary

Pi = 3.141592653589793.

ceil (2.3) = 3

floor (2.3) = 2

factorial (5) = 120

gcd (5, 15) = 5

abs (-10) = 10

pow (3, 5) = 243

log base 3 of 2 = 0.630929753571457 4

log 10 (100) = 2.0

Pinf (inf) = True, Pinan (NAN) = True

--- OS & SYS ---

Created / ensured: C:\Python lab

Current working directory: C:\... (your current

Created / ensured & changed into C:\Python

Directory contents of C:\Python lab 52L4:0

Python version: 3.x.x (... details ---)

--- STATISTICS ---

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]) = 2.27 563595

Print python interpreter version.

4) statistics

- on lists: [5,6,8,10] and [2,5,3,2,8,3,4,2,5,6],

Compute mean, median, mode, stdev.

5) Print neatly formatted results.

PROGRAM :-

```
card Fun
Pmport random
dlt - fun():
card1 = []
for i in range (1,53):
cardi. append (i)
shuttled _ cards = random . sample (card1, k=52)
PrPNt ("\n\n", shuttled _ cardi, "\n\n")
```

My mod. Py

```
Pmport CardFun
CardFun. func()
```

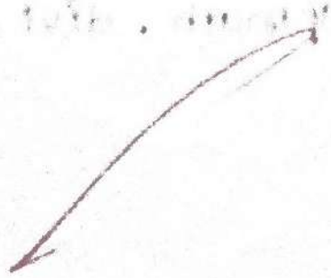
OUTPUT :-

```
RESTART:
c:\Users\student.MATRVC6833\AppData\Local\Progra
  Python\Python312\LPblSPte - Packages\ card pack\My
[5,24,13,22,20,41,38,51,4,7,34,49,14, 50,37,
  35,17,18,33,39,36,42,12,6, 16,19,48,29,2,27,
  46,28,21,32,8,25,30,23,26,10,45,47,3,44,52
  45,9]
```

3.2 create a python package name card pack 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:-

Step - 1: Start
Step 2: To create a package cardpack
Step 3: TO create a module cardfun and import random function
Step 4: Assign a cards range
Step 5: call a module function
Step 6: Display the random sample cards
Step 7: Stop.



```
mymath : (my) math
def add (a,b):
    return a+b
def subtract (a,b):
    return a-b
def multiply (a,b):
    return a*b
def divide (a,b):
    if b==0:
        raise ValueError("Cannot divide by zero")
    return a/b
import mymath
a=10
b=5
print("Addition:", mymath (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
```

3.3 You are tasked with developing a modular calculator application in python. The calculator should support basic arithmetic operations: addition, subtraction, multiplication, and division. Each operation should be implemented in a seperate module

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 (my math) containing these function
4. Initialize two numbers (a=10, b=5)
5. Call each function using 'my math < function_name => (a,b)'
6. Print the result of all operations.

PROGRAM:

1. create the math-function.py module

```
def add (a,b):
 return a+b
def subtract(a,b):
 return a-b
def multiply (a,b):
 return a*b
def divide (a,b):
 if b == 0
return "ERROR! Division by zero."
 return a/b
```

2. create the area-function.py module

```
import math
def circle_area (radius):
return math.pi * radius * radius
def rectangle_area (length, width):
return length * width
def triangle_area (base, height):
 return 0.5 * base * height
```

3. Create the main.py file

```
import mathfunctions
import area function
# Using math functions
Print ("Addition:" mathfunctions.add (10,5))
print ("subtraction:" mathfunctions.subtract(10,5))
print ("multiplication:" mathfunctions.divide (10,5))
# Using area-functions
Print("Circle Area (radius = 7):", area function.
Print ("Rectangle Area (5x10):", area function.
Print (" Triangle Area (base=6, height=8):", area
function.triangle_area (
```

3.4 You are working on a python that requires you to perform various mathematical operations & geometric area calculations. To organize your code better, you decide to create a package named my package which includes sub packages pack1 and pack2 with two modules: math functions and area functions. Demonstrate the use of the functions by performing a few calculations and printing the results.

Algorithm:-

1. Create math functions.py module:
2. Create area functions.py module:
3. Create main.py:
4. Print the output as expected.

OUTPUT:

Addition : 15
subtraction : 5
multiplication : 50
Division : 2.0
Circle Area (radius = 7): 153.93804002589985
Rectangle Area (5x10) : 50
Triangle Area (base=6, Height=8): 24.0

RESULT:-

Thus, the program for importing python modules and packages was successfully executed and the output was verified.

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

18/08/25

TASK 4:- Use various data types, List Tuples and Dictionary in python programming key terms covered: Data types, List Tuple, set, Dict Tags - Easy 10

Aim:- Record a cafeteria the sales for 7 days using a list, compute total and average sales, find the best / worst day and count how many days crossed a target

ALGORITHM:-

1. Start
2. Create an empty list sales = []
3. For 7 days, append integer sales to the list using append()
4. compute total = sum(sales) and avg = total/7
5. Find max_val = max(sales) min_val = min(sales)
6. Find corresponding days with index()
7. Count days above a target using count() and a boolean re-map or with a loop.
8. Stop.

Program (Use) append(), index(), count();

```
# List scenario
days = 7
Sales = []
target = 500 # target sales for the day.
for s in range(3):
    sample_entries = int(input("enter the seven days sales count"))
    Sales.append(sample_entries) # list.append()
total = sum(Sales)
avg = total/days
```

Output :-

enter the seven days sales coow 100
enter the seven days sales coow 450
enter the sevendays sales coow 1250
enter the seven days sales coow 78
enter the seven days sales coow 900
enter the seven days sales coow 239
enter the seven days sales coow

sales (mon - sun) : [100, 450, 1250, 78, 900, 239]
Total : 3974
Average 567.71
Best days 3 with 1250
Worst day 5 with 78

Output:-

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

11 14:00 present? True

14:00 occurs 2 times

First occurence position(1-based): 3

morning slots: (9, 11)

afternoon slots (14, 16, 14)

4.2. TUPLE - LAP TIME TABLE

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
exists = (query in slots)      # tuple.count()
freq = slots.count(query)
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}:00 present?", exists)
print("First occurrence position (1-based):", first_pos)
print("Morning slots:", morning)
print("Afternoon slots:", afternoon)
```

output:

Enter the number of items: 2

Enter item name: box

Enter price of box: 15

Enter item name: pen

Enter price of pen: 10

Enter item name: pencil

Enter price of pencil: 5

Enter item to update price: box

Enter new price for box: 20

Enter an item to remove from price list: pen

Available items: ['box', 'pencil']

prices: [20.0, 5.0]

Updated item: box at 20.0

Removed 'pen' price (if existed): 10

4.3. DICTIONARY - BOOK STORE BILLING :-

Aim :- To manage a live price list and bill a customer using dictionary methods and views.

Algorithm

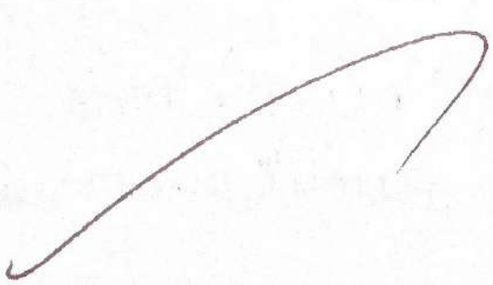
1. Start
2. Create an empty dictionary prices.
3. Ask the user for the number of items in the price list (n).
4. Repeat for each item.
5. Get the item name.
6. Get the item price.
7. Add the item and price to prices.
8. Ask the user for an item to update
9. If the item exists in prices, get the new price and update it.
10. Find the costliest item by checking each item's price.
11. Ask the user for an item to remove.
12. If given, remove that item from prices.
13. Show all available items

Python program :-

```
prices = {}
n1 = int(input("Enter number of items in price list: "))
for _ in range(n1):
    item = input("Enter item name: ")
    price = float(input(f"Enter price of {item}: "))
    prices[item] = price

# optional price revision
rev_item = input("Enter item to update price (or press Enter to skip): ")
# rev_item in prices:
    new_price = float(input(f"Enter new price for {rev_item}: "))
    prices.update({rev_item: new_price})# dict update()
#find costliest item
    max_price = price
    costliest_item = item.
```

```
# Remove out-of-stock item.
remove_item = input("enter an item to remove from
    price list (or press enter to skip): ")
removed_price = None
if remove_item:
        removed_price = prices.pop(remove_item, None)
                                        # dict.pop()
# Display results
print("\n Available items:", list(prices.keys())) # dict.keys()
print("prices:", list(prices.values())) # dict.values()
if cottage_item:
    print("Cottage item:", cottage_item, "at", max_price)
if remove_item:
    print(f"Removed '{remove_item}' price (it existed):",
    removed_price)
```



Output :-

Enter number of participation in AI Hackathon

Enter participation ID : A1

Enter participation ID : A2

Enter participation ID : A4

Enter participation ID : A5

Enter number of participants in Robotics Challenge : 4

Enter participation ID : A1

Enter participation ID : A2

Enter participation ID : A3

Enter participation ID : A7

Enter Registration ID for AI Hackathon : A1

Enter withdrawn participation ID : Robotics Challenge : A1

AI Hackathon : {'A1', 'A4', 'A5', 'A8', 'A2'}

Robotics Challenge : {'A2', 'A7', 'A3'}

Both events : {'A2'}

Only AI : {'A4', 'A8', 'A5', 'A1'}

Total unique participants : 7

4.4 SET - TECHFEST PARTICIPATION.

Two events AI Hackathon and Robotics challenge have participants' IDs stored in two sets and a late registrant to AI Hackathon.

```
# Get AI Hackthon participants
ai_ hackthon = set()
n1 = int(input("enter number of participants in AI Hackthon:"))
for _ in range(n1):
    pid = input("enter participant ID:")
    ai_ hackathon.add(pid)

#Get Robotics challenge participants
robotics_challenge = set()
n2 = int(input("enter number of participants in
        Robotics challenge:"))
for _ in range(n2):
    pid = input("enter participant ID:")
    robotics_challenge.add(pid)

# Add a late registrant
late_id = input("enter late registrant ID for AI
Hackthon (or press Enter to skip):")
if late_id:
    ai_hackthon.add(late_id) # set.add()

# Remove a withdrawn participant
remove_id = input("enter withdrawn participant
ID from Robotics challenge (or press enter to skip):")
if remove_id:
    robotics_challenge.discard(remove_id)
                # set.discard()
```

Set operations

both = ai - hackathon.intersection(robotics - challenge)

only_ai = ai - hackathon.difference(robotics - challenge)

only_robotics = robotics - challenge.difference(ai - hackathon)

unique_all = ai - hackathon.union(robotics - challenge)

VEL TECH	
E.NO.	4
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	
SIGN WITH DATE	15

Result:- Thus use of various data types like tuple, set, list, dictionary, array data types are successfully executed

output

{'id': 2, 'name': 'bob', 'department': 'engi

TASK-5 Implement various searching and sorting

5.1 operations in python programming.

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

Algorithm.

1. Input definition.
2. Define the function find_employee_by_id that takes two parameters.
   a. a list of dictionaries.
   b. an integer.
3. Iterate through the list.
4. check for matching ID.
5. Return matching Record.
6. Handle NO. match.

Program 5.1

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

output:

Before sorting:

{'name': 'Alice', 'score': 88}
{'name': 'Bob', 'score': 95}
{'name': 'Charlie', 'score': 75}
{'name': 'Diana', 'score': 85}

After sorting

{'name': 'Charlie', 'score': 75}
{'name': 'Diana', 'score': 85}
{'name': 'Alice', 'score': 88}
{'name': 'Bob', 'score': 95}

5.2. You are developing a grade management system for a school. The system maintains a list of student records, where each record is represented. your task is to implement a feature that sorts the student records by their scores using bubble sort algorithm

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
3. Track swaps.
   - Initialize a boolean variable swapped to false.

This variable will track if any swaps are made in the current pass

4. Inner loop
   Iterate from j=0 to n-i-2 (inclusive).
5. compare and swap.
6. Early termination.
7. completion.

Program 5.2

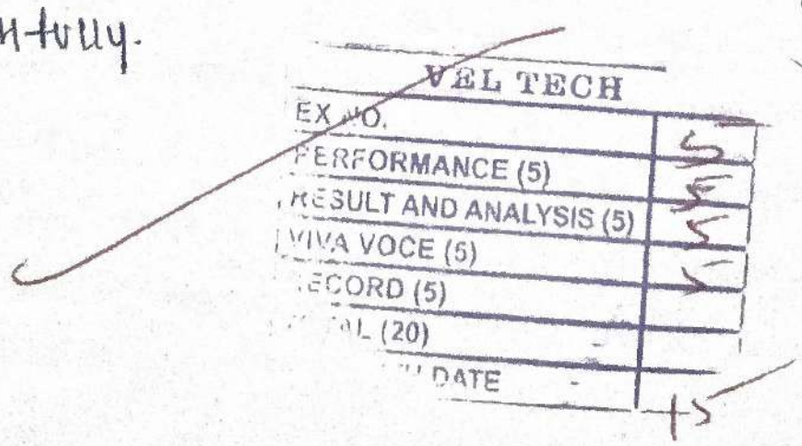
```
def bubble_sort_scores(students):
    n = len(students)
    for i in range(n):
        # Track if any swap is made in this pass
        swapped = false
        for j in range(0, n-i-1):
            if students[j]['score'] > students[j+1]['score']:
                # swap if the score of the current student
is greater than the next students[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.

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

RESULT:

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



03/08/2

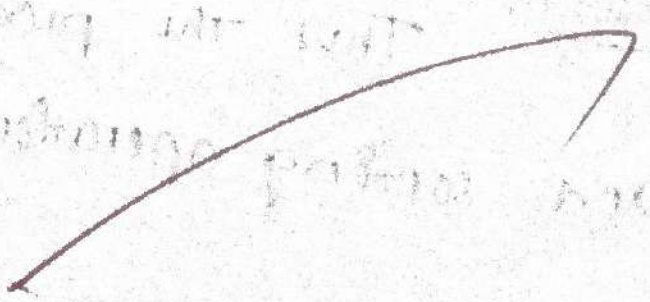
Output:-

Welcome to The Student grades Analyzer

Number of students: 4

Type of student_name 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 finder from 1 to number of students: [1, 2, 3, 4]



03/08/25 TASK 6. UTILIZING 'FUNCTIONS' CONCEPTS IN PYTHON PROGRAMMING.

6-1 You are developing a small python script to analyze and manipulate a list of student grades for a class project. Write a python program that satisfies the above requirements using the built-in function print(), len(), type(), max(), min(), sorted() reverse(), and range().

Algorithm:-

1. Start the program.
2. print a welcome message: output a simple greeting.
3. Determine and print the number of students: use len()
4. Print the type of list: use type() to show the type of student_names and student_grades list.
5. find and print highest and lowest grades.
6. Print sorted list of grades.
7. Print reverse list of grades.
8. generate and print a range of grade indices: indices from 1 to the number of students.
9. Stop

Program:-

```
def analyze_student_grades()
    # sample data.
    student_names = ["Alice", "Bob", "Charlie", "Diana"]
    student_grades = [85, 92, 78, 90]

# 1. print a welcome message.
print ("welcome to the student grades Analyze \n")
# 2. Determine and print the number of students
    num_students = len (student_names)
    print ("Number of students:", num_students)
```

output:

Arithmatic 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

calculator application to help user perform basic arithmetic operation and greet them with a personalized message

Algorithm:

1. Start the program.
2. User Input for numbers:
3. User Input for Operation The program prompts the user to choose an arithmetic Operation (addition, subtraction, multiplication, division).
4. Perform Operation: Based on the users choice the program performs the choose arithmetic operations using the defined functions.
5. Display Result: The program displays the result of the operation.
6. Stop.

6-2 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 product of two numbers"""
    return a*b
def divide (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 operation.
    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.
    user_name = "Alice"
    print("\nGreeting:")
    print(greet(user_name))

# Run the main function.
if __name__ == "__main__":
    main()
```

Result:

Thus, the python program using functions' concept was successfully executed and the output was verified.

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

Output:-

- Student details using read function is

USN No.	Name	Age.
2305	RAM	21
1920	shiva	21
2305	RAM	20
1920	shiva	21

- Output read fun function is

2305 Ram 20

find the current position of the file pointer is 29.

TASK#7 IMPLEMENT VARIOUS txt / csv file operations

10/09/25

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

Algorithm

step-1: start

step-2: Using open() method, create and write textfile "myfile.txt" with student details.

step-3: Update the new registered student details using append operation in it.

step-4: open the file in read mode and using read() method print the student details

step 5 : Using seek method print the particular student record.

step 6: Using tell method print the current position of the file

step 7: close the file.

step 8 : Stop.

PROGRAM

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

```
file = open("student1.txt","a")
input3 = input("Enter updated student details")
file.write(input3)
file = open("student1.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:")
t = file.tell()
print(t)
file.close()
```

Result :-

Thus, the python program for creating and updating student registration

output

```
#include <stdio.h>
int main()
{
```

counting cases

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

```
# program to count upper case, lower case
and digits in a file. (merge.txt)
# step 1: (reate 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")
# step-2: open the file for reading
with open ("merge.txt", "r") as rf:
        txt = f.read()
    # step 3. initialize counters.
    upper_count = 0
    lower_count = 0
    digit_count = 0
    # step 4: count upper case, lower case,
    and digits
    for char in text:
        if char.isupper():
            upper_count += 1
        elif char.islower():
            lower_count += 1
        elif char.isdigit():
            digit_count += 1
```

Step5: Print the result

```
print ("Upper case letters :", upper_count)
print ("lower case letters :", lower_count)
print ("Digits:", digit_count)
# compact output as required
print(f"{upper_count}, {lower_count}, {digit_count}")
```

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

Program

```
# step1: Read input data from grade.txt.
with open ("grade txt","r") as f:
    lines = f.readlines()
# step 2: Extract number of students
n= int (lines[0].strip()).
# Hep3: Extract weights
weights = lines [1].strip().split()
weight = [float (w) for w in weights]
# step4: Process each student's data.
students = []
for i in range (2, 2+n):
    parts = lines[i] .strip(). split()
    name = parts [0]
    marks = [int(m) for in parts [1:]]
    # calculate weight average.
total=0
for j in range (4):
    total = total + marks [j] * weights [j]
    students . append ((name, round (total, 2)))
# step 5: write results into result. txt
with open ("result.txt", "w") as f:
```

name, avg in student:

f.write(name + "->" + str(avg) + "\n")

print("Average grades have been written to result.txt")

Result:-

Thus the Implementation of various txt/csv file operations in python successfully executed and verified

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

17/09/2

TASK8: IMPCEMENTATION OF PYTHON GENERATOR AND DECORATORS

Aim :- Write a python program to implement python generators and decorators

Algorithm :-

1. Define generator function.
   Define the function number_sequence (start, end, step=1)
2. Initialize current value
3. generate sequence
4. get user input
5. Create generator object
6. print generated sequence

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)
```

[illegible]

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

Algorithm :-

1. Start the function
2. Initialize counter
3. Generate values
4. Create generator object
5. Iterate and print values.

Program

```
def my_generator(n):
    # Initialize counter
    value = 0
    # loop until counter is less than n
    while value < n:
        yield value
        # Increment the counter
        value += 1
# Iterate over the generator object produced by my_generator
for value in my_generator(3):
    # print each value produced by generator
    print(value)
```

2.2 Imagine you are working on a messaging application the needs of formal nursery

Algorithm

1. Create Dictionary
2. Define functions
3. Define greet function
4. Execute the program

Output:

Hi IAm CREATED BY A FUNCTION PASSED
AN ARGUMENT.

hi i am created by a function passed
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).upper()
    return wrapper
def lowercase_decorator (func)
    def shout (text)
        return text
    @ lowercase_decorator
    def whisper (text):
        return text
    def greet (func):
        greet = func ("I am function passed as arguments")
        print (greeting)
    greet (shout)
    greet (whisper)
```

VEL TECH	
EX.NO.	
PERFORMANCE (5)	
RESULT AND ANALYSIS (5)	
VIVA VOCE (5)	
RECORD (5)	
TOTAL (30)	

Result:- Thus the python program to implement python generator and decorator was successfully executed and the output was verified.

Output

grades list [85, 90, 78, 92, 88]

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

Invalid index. Please enter a valid index.

Q) You are developing a python program that stores a list of grade of students

Algorithm

- Start the program
- Initialize a list of grades e.g [85, 90, 78, 92, 88])
- Prompts the user to enter the index of the grade they with to view.
- Attempts to display the grade.

Program

```
# 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 wants to view try
index = int(input("Enter the no of index of the grade you want to view"))
#Attempt to display the grade to view
except IndexError:
    # Handle the case where the index is out of range
    print("Invalid index. Please enter a valid index.")
except ValueError:
    # Handle the case where the input 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.

3. you are developing a python calculator program that performs basic

Algorithm:

1. Start the program.
2. prompt the user to enter two enter the numerators
3. Attempts to divide the numerator by 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.
    numerator = float (input ("enter the numerator)
    # prompt the user to enter denominator
    # denominator = float (input ("enter a denominator")
    # Attempt to perform division
    result = numerator / denominator
    print ("Result : {result}")
except zero division error:
    # Handle division by zero error
    Print ("Error: Please enter a valid number")
# Call the function to execute the division
operation divide_numbers()
```

output.

Enter a number: 15
exception occured Invalid age

to determine if a person is eligible

Algorithm

1. Define the custom exception
2. prompt the user for input
3. check the 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 when the input value is
    less than 18"
    pass
# you need to guess this number
    number = 18
try:
    input_num = int(input("enter a number:"))
    if input_num < number:
        raise invalid Age exception
    else
        print ("eligible to vote")
except invalid Age exception:
    print ("exception occured : Invalid Age")
```

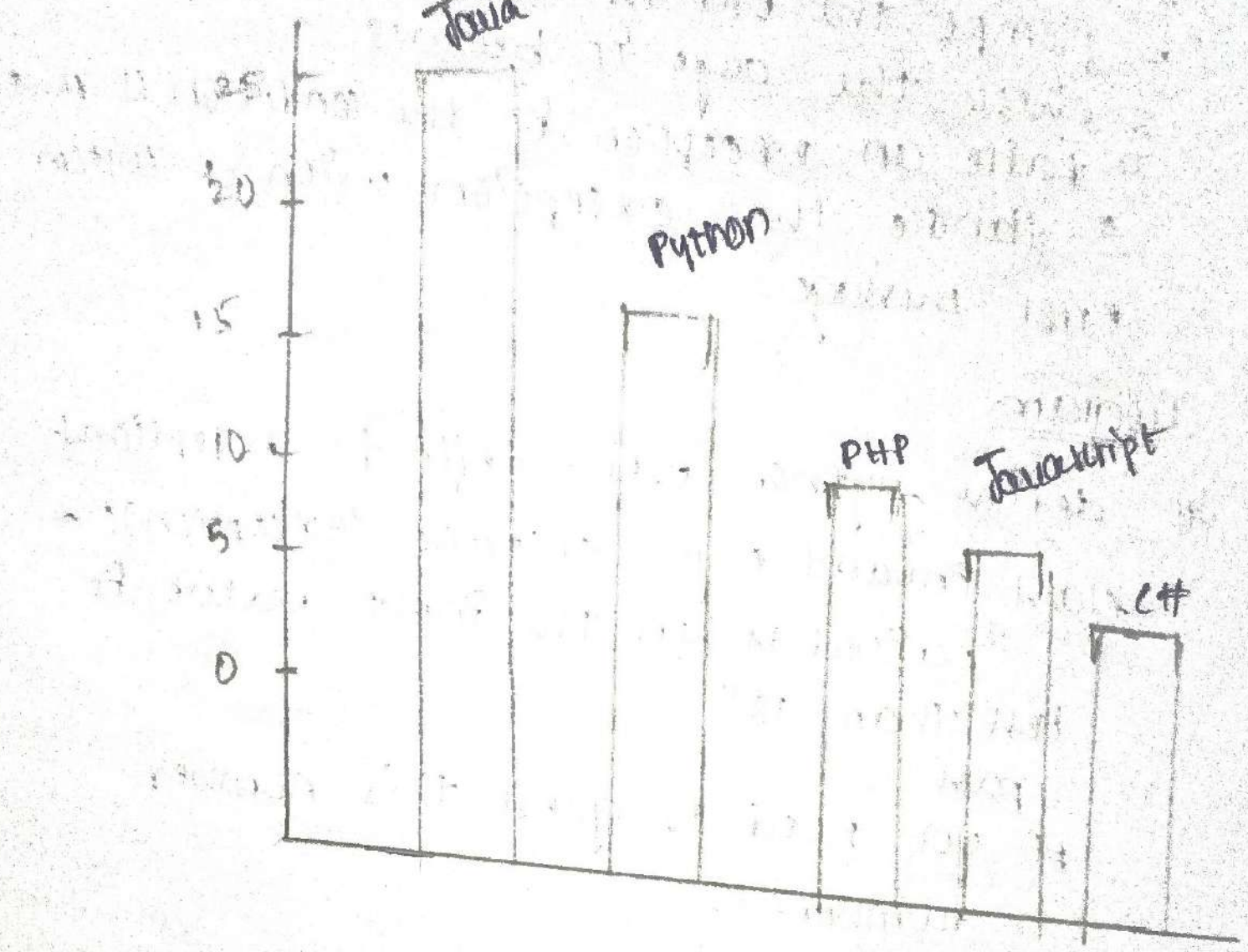
VELTECH

EX NO.	4
PERFORMANCE (5)	
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	
RECORD (5)	
TOTAL (20)	15

Result :-

Thus the program for Implement exceptions and exceptional handling is executed and verified successfully

OUTPUT:



TASK 10:- Use Matplotlib module for plotting in python

10.1 Aim :- To use matplotlib module for plotting in python

sample data:

Programming languages : Java, Python, PHP, Javascript, C#, C++

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

sample output:

Algorithm:-

1. Define two lists for programming languages and their popularity
2. Find the maximum popularity value in the list
3. Define a scaling factor to scale the bar heights.
4. For each language and popularity pair calculate

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.xlabel('Programming languages')
plt.ylabel('popularity')
plt.show()
```

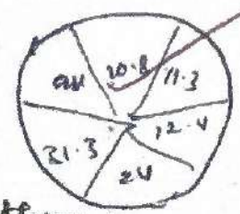
10.2 Write a python programming to create a pie chart of the popularity of programming languages

sample data:

Programming languages: Java, Python, PHP, Javascript, C#, C++

popularity = 22.2, 17.6, 8.8, 8, 7.7, 6.7

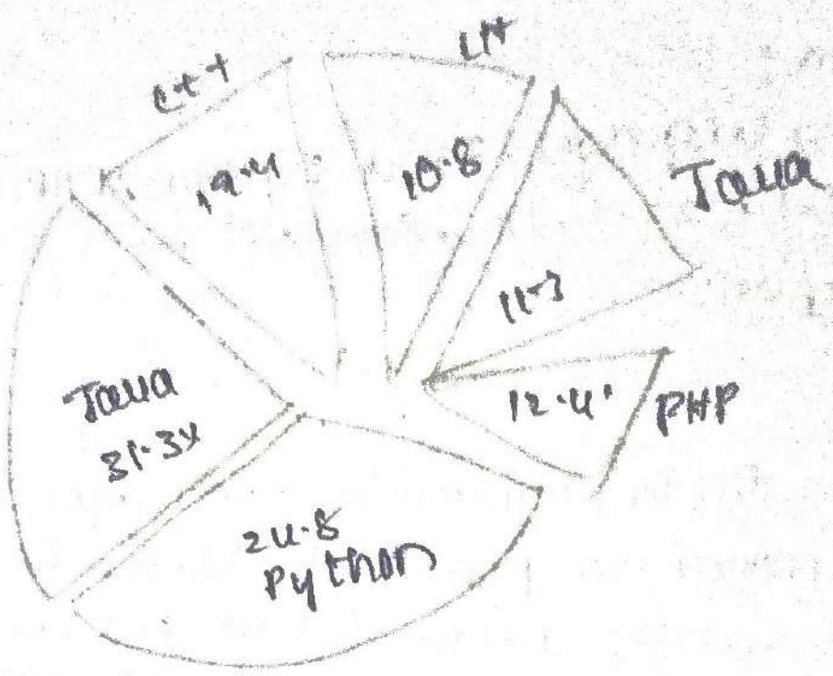
sample output



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 piechart
4. Show the pie chart

Output



Program:

```
import matplotlib.pyplot as plt
# step 1
languages = ['Java', 'Python', 'PHP', 'JavaScript', 'C#', 'C++']
popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]
# step 2
plt.pie(popularity, labels = languages, autopct='%.1f %%')
# step 3
plt.title("Popularity of programming languages")
plt.legend(languages, loc="best")
# step 4
plt.show()
```

VEL TECH	
EX NO.	10
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 via matplotlib module for plotting is executed and verified successful.

output

Hello, world!

change fonts

15/02/25

TASK 11. UI & - TKINTER MODULE FOR UI DESIGN.

Aim:- To use Tkinter module for UI Design.

Algorithm:-

1. Import tkinter module.
2. create a main window
3. Create a label with desired txt
4. Add the label to the main window using pack() method.
5. Define a function to the change font style

Program:-

```
Import tkinter as tk
# function to change font style
def change_font():
    label.configure(font=("Arial", 18, "bold"))
root = tk.Tk()
label = tk.Label(root, text="Hello, World", font
label.pack()                    =("Helvetica", 14))
button = tk.Button(root, text="Change Font", comm-
    and = change_font)
button.pack()
root.mainloop()
```

TASK 11.2 Write a python GUI program to create three single line text-box to accept a value from the user using tkinter module

Name :
UserID :
Password
Submit

Algorithm:-

1. Import the tkinter module
2. Create the main window
3. Add labels and txt-boxes to the main window
4. Set the size of the txt boxes.
5. Create a button to submit the values entered in the txt boxes

Program :-

```
import tkinter as tk
root = tk.Tk()
root.title("Text box input")
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 value3:")
entry3 = tk.Entry(root)
entry1.config(width=30)
entry2.config(width=30)
entry3.config(width=30)
def get_value():
    val1 = entry1.get()
    val2 = entry2.get()
    val3 = entry3.get()
    print("value 1:", val1)
    print("value 2:", val2)
    print("value 3:", val3)
submit_button = tk.Button(root, text = "submit", command
                          = get_value)
label1.pack()
entry1.pack()
label2.pack()
entry2.pack()
label3.pack()
entry3.pack()
submit_button.pack()
root.mainloop()
```

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

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

TASK 12

SIMULATE GAMING CONCEPTS USING PYGAME.

Aim: To simulate gaming concepts using Pygame.

Algorithm

1. import pygame package and initialize it
2. Define the window size and title
3. create a snake class which initializes the snake position, colour, and movement
4. Create a fruit class which initializes the fruit position
5. If the snake hits the fruit increase the score by 10.

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)
pygame.init()
pygame.display.set_caption('Greeks for Greeks snakes')
game_window = pygame.display.set_mode((window_x, window_y))
fps = pygame.time.Clock()
snake_position = [100, 150],
snake_body = [[100, 50],
              [90, 50],
```

```
[70,50]
]
fruit_position = [random.randrange(1, (window_x//10))*10
        random.randrange(1, (window_y//10))*10]
fruit_spawn = True
direction = 'RIGHT'
change_to = direction
score = 0
def show_score(choice, color, font, size):
 score_font = pygame.font.SysFont(font, size)
 score_surface = score_font.render('Score: '+str(score), True,
                                   color)
 score_rect = score_surface.get_rect()
def game_over():
 my_font = pygame.font.SysFont('times new roman', 50)
 game_over_surface = my_font.render(
        'Your Score is: '+str(score), True, red)
  game_over_rect = game_over_surface.get_rect()
  game_over_rect.midtop = (window_x/2, window_y/4)
  game_window.blit(game_over_surface, game_over_rect)
    pygame.display.flip()
    time.sleep(2)
    # main function
  while True:
      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_LEFT:
                  change_to = 'LEFT'
              if event.key == pygame.K_RIGHT:
                  change_to = 'RIGHT'
  if change_to == 'UP' and direction != 'DOWN': direction = 'UP'
        direction = 'UP'
  if change_to == 'DOWN' and direction != 'UP':
        direction = 'DOWN'
  if change_to == 'LEFT' and direction != 'RIGHT':
        direction = 'LEFT'
  if change_to == 'RIGHT' and direction != 'LEFT':
        direction = 'RIGHT'

    if not fruit_spawn:
      fruit_spawn = True
    show_score(1, white, 'times new roman', 20)
    pygame.display.update()
    fps.tick(snake_speed)
```

12.2

WRITE A PYTHON PROGRAM TO DEVELOP A CHESS BOARD USING PYGAME

~~sample output~~:

Aim:- write a python program to develop a chess board.

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 pieceies on the boaard by looping over
5. Draw the board and pieces on the screen.
6. start the game loop

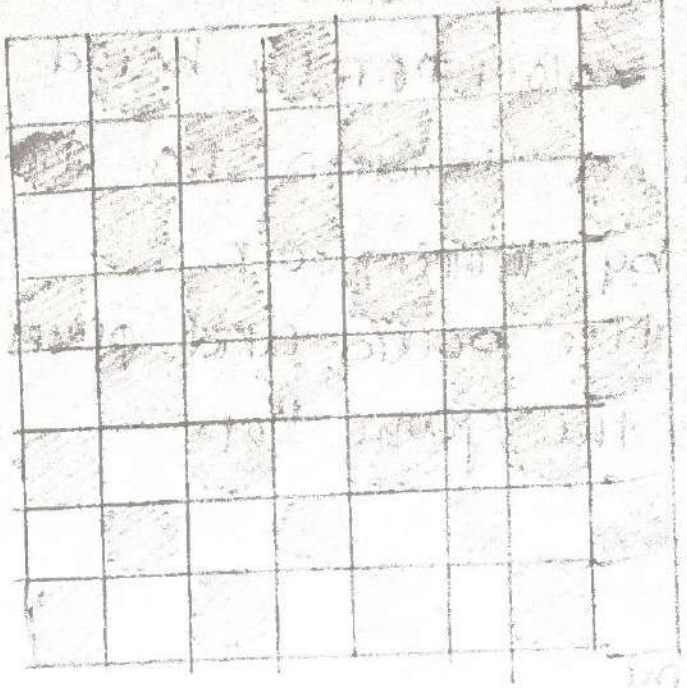
Program:-

```
import py game
# intialize py-game.
  py game .init()
# set screen size and title.
 screen - size = (640, 640)
 screen = pygame. display. set_mode (screen_size)
 pygame. display.set_caption ('chess board')

 # Define colors
  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_color = white if (row + col) % 2 == 0 else brown
              square_rect = pygame.Rect(col * 80, row * 80, 80, 80)
  pygame.draw.rect(screen, square_color, square_rect)
```

OUTPUT :-



```
def draw_pieces(board):
    piece_images = {
        'r': pygame.image.load('images/rook.png'),
        'n': pygame.image.load('images/knight.png'),
        'b': pygame.image.load('images/bishop.png'),
        'q': pygame.image.load('images/queen.png'),
        'k': pygame.image.load('images/king.png'),
        'p': pygame.image.load('images/pawn.png'),
    }
    for row in range(8):
        for col in range(8):
            piece = board[row][col]
            if piece != '.':
                piece_img = piece_images[piece]
                piece_rect = pygame.Rect(col*80, row*80,
                    80, 80)
                screen.blit(piece_img, piece_rect)

board = [
    ['r','n','b','q','k','b','n','r'],
    ['p','p','p','p','p','p','p','p'],
    ['.','.','.','.','.','.','.','.'],
    ['.','.','.','.','.','.','.','.'],
    ['.','.','.','.','.','.','.','.'],
    ['.','.','.','.','.','.','.','.'],
    ['P','P','P','P','P','P','P','P'],
    ['R','N','B','Q','K','B','N','R']
]

draw_board()
draw_pieces(board)
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            quit()
    pygame.display.update()
```

Completed

VEL TECH	
EX NO.	11
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
[illegible] (20)	

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