

S.No	Date	Title	marks	sign
1	28/08/25	Introduction to python commands, script	15/15	L
2.	30/08/25	implement conditional and looping statement	15	F12
3	06/08/25	importing and creating python module and package in python program	15	
4.	13/08/25	use various datatypes list tuples and dictionaries in python programming	15 15 15	10/18
5.	20/08/25	implement various searching and sorting operation in python	15	
6	03/08/25	utilizing functions concepts in python program	15	
7.	10/09/25	implement python various file operation	15	
8	17/09/25	Implement Python generators 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.	25/10/25	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 required modules: math, random, os, sys, statistics, pathlib.
- 2) math & random:
 - compute sqrt(5) radians (70), random float in [0,0,10], factorial(5), gcd(5,15), abs(-10), pow(3,5), log base 3 of 2, log10(a) for a=100 and check: non/infinity;
- 3) os & sys:
 - create c:\pythonlab if not present and print current working directory.
 - create c:\python\slot 524 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, 7, 9, 4, 25, 6] compute mean, median, mode, stdev.
- 5) Print neatly formatted results.

(F)

Python program:

```
# program to calculate total expenses of karan  
# 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 karan : £ total_expense")
```

(4)

'sample input:

(values are already assigned in the program, no manual
input required)

Books = £150

Groceries = £220

Transport = £90

(5)

sample output.

Total expenses incurred by karan = £460

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

Introduction to python, commands, scripts, Tags ^{easy} [01,5]

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

④ Aim:- To write a python program that calculates the total amount spent by Karan on books, groceries and transport.

Ans studio : write it

(\Rightarrow input) | display = print

Muskaan studio : write it

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.

: for two

Ans: 29 (Ans) value 220m when you

Result: hence the python program for finding the area of triangle with all sides using heron's formula, is done executed successfully. books, groceries transport has been done successfully using python.



Python program to calculate BMI

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 kilograms: 70

Enter your height in meters: 1.75

Output:

Your Body Mass Index (BMI) is: 22.86

Q.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 height (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

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

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

Result: Hence the python program for calculating the Body mass 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

$$\text{area} = \text{math.sqrt}(s \times (s-a) \times (s-b) \times (s-c))$$

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

11.62

1.3

Maya wants to calculate the area of scalene triangle with sides of length 8cm, 6cm, and 10cm. 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:
$$\text{Area} = \sqrt{s(s-a)(s-b)(s-c)}$$
- 5) display the area of the triangle
- 6) End the program.

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

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



PROGRAM to find the grade of student of given score

```

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):
    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 score: 60

The Grade is D.

STUDENT INFORMATION	
EX NO.	PERFORMANCE (%)
1	RESULTS AND ANALYSIS (3)
2	AVERAGE (%)
3	RECORD (4)
4	TOTAL (12)
5	GEN WITH DATE

Note: Hence give reason based on for finding grade

wherever you find any error in the program which is due to wrong code of input or output

Explanatory example and Q

30/1²⁵ 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 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 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 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:

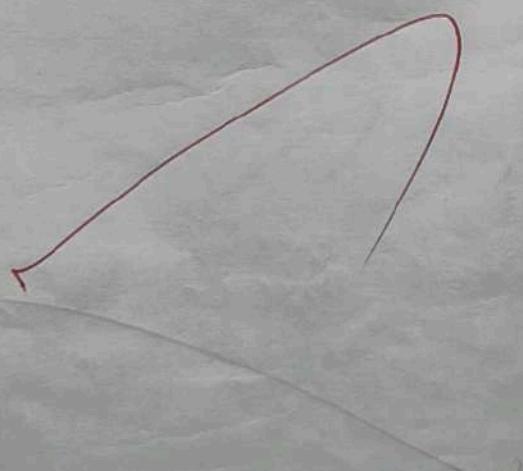
```
# Battery Health checker  
percentage = int(input("Enter battery percentage :"))  
  
if percentage >= 90:  
    print("Excellent battery Health")  
elif percentage >= 70:  
    print("Good battery Health")  
elif percentage >= 40:  
    print("Average battery Health")  
else:  
    print("Poor battery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

Battery charge percentage (integer)

Sample output:

```
Enter battery percentage: 85  
Good battery Health
```

Note: 11



2.2 The electronics maintenance team at a data center needs a tool to assess the health status of UPS back batteries based on their current charge percentage. You are asked to develop a Python input and categories the battery health 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 program that uses ladderized if-elif-else statement!

Algorithm:

1. Accept battery percentage from the user
2. Use ladderized if-elif-else to determine the health category
 - If percentage $\geq 90 \rightarrow$ "Excellent Battery Health"
 - If $70 \leq$ percentage $\leq 90 \rightarrow$ "Good Battery Health"
 - If $40 \leq$ percentage $< 70 \rightarrow$ "Average Battery Health"
 - If percentage $< 40 \rightarrow$ "Poor Battery Health"

Program. And try to write code of height of boat for ; integer (1,6) :

height = int(input("Enter height of visitor [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 visitor 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	95

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

```
printf("current working directory : %s\n", cwd())  
target_dir = path(r"C:\python\slot\52\ii")  
target_dir.mkdir(parents=True, exist_ok=True)  
os.chdir(target_dir)  
printf("changed into : %s\n", target_dir)  
printf("Directory contents : %s\n", os.listdir())  
printf("Python version : %s\n", sys.version)  
print("In- STARTISTICS")  
data1 = [5, 6, 8, 10]  
data2 = [2, 5, 2, 8, 3, 9, 4, 2, 5, 6]  
printf("mean %s", stats.mean(data1))  
printf("median %s", stats.median(data1))  
printf("mode %s", stats.mode(data2))  
printf("std dev %s", stats.stdev(data2))  
Expected sample output:  
-- MATH & RANDOM --  
sqrt(5) = 2.236067997749979  
radians(30) = 0.5235987755982988  
random() in [0,1] = 0.38446887175646646 ← will change  
radint(2,6) = 6  
pi = 3.141592653589793  
ceil(2.5) = 3 factorial(5) = 120 gcd(5,15) = 5. abs(-10) = 10  
pow(2,5) = 32, log base 2 of 2 = 0.6309297535714574  
log(100) = 2.0  
isinf(20) = True , is an (now) = True.  
-- OS & SYS --  
created/ ensured: c:\python\slot  
Current working directory : C:\... (your currentpath)  
created/ensured & changed into: C:\python\slot\52\ii:  
directory contents of C:\python\slot\52\ii: []
```

Python version: 2xx (details)

... STATISTICS -

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

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

mode ([2, 5, 7, 2, 18, 7, 9, 4, 2, 5, 6]) = 2

stddev([2, 5, 7, 2, 8, 7, 9, 4, 2, 5, 6]) = 2.0715633783201093

Program :

cardfun

import random

def func():

cards = []

for i in range (1, 52):

cards.append(i)

shuffled_cards = random.sample(cards, k=52)

print ("In\n", shuffled_cards, "\nIn")

mymod.py

import random

card fun func

output:

Restart:

C:\Users\student\MAT 2VC G83>APPdata\local\program

python3.11\lib\site-packages\cardpack\mymod.py

[5, 24, 13, 22, 20, 21, 28, 5, 4, 7, 34, 49, 14, 50, 27, 40, 15, 35, 17, 18, 33, 39, 36, 42, 12, 6, 16, 19, 48, 29, 29, 2, 27, 11, 31, 46, 28, 21, 32, 8, 25, 30, 27, 26, 10, 43, 47, 3, 44, 52, 143, 97]

3.2

create a python package named card pack containing a module card fun 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 card pack.

Step 3:- To create a module card fun 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

call each function using myrandom function - name ??

6. Point one result of all functions
((d,0) 0.0, plonim " : noitibba") true

: fufed

21: noitibba

22: noitibba

0.0: noitibba

0.5: noitibba



S.L

Program : (mymath)

```

def add(a,b):
    return a+b

def subtraction(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.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 separate module. Additional 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

u.2 output: 14:00 present ? . true
All lab slots : (9, 11, 14, 16, 18),
IS 14:00 present ? . true

14:00 occurs 2 times(s)

First occurrence position (1-based) : 1

mornins slots : (9, 11, 14)

afternoon slots : (14, 16, 18)

14:00 occurs 2 times(s)
First occurrence position (1-based) : 1
mornins slots : (9, 11, 14)
afternoon slots : (14, 16, 18)

14:00 occurs 2 times(s)
First occurrence position (1-based) : 1
mornins slots : (9, 11, 14)
afternoon slots : (14, 16, 18)

14:00 occurs 2 times(s)
First occurrence position (1-based) : 1
mornins slots : (9, 11, 14)
afternoon slots : (14, 16, 18)

14:00 occurs 2 times(s)
First occurrence position (1-based) : 1
mornins slots : (9, 11, 14)
afternoon slots : (14, 16, 18)

(14:00) true, (14:00) true

```

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 strings. 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
exists = (query in slots)
freq = slots.count(query) # tuple.count()
first_pos = slots.index(query) if exists else "NA" # tuple.index()

morning = slots[:2]
afternoon = slots[2:]

print ("All labs slots:", slots)
print (f"Is {query} : oopresent?", exists)
print (f"{query} : occurs {freq} times(s)")
print ("First occurrence position(1-based):", first_pos)
print ("Morning slots:", morning)
print ("Afternoon slots:", afternoon)

```

4.3 output

Enter number of items in price list : 2
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

Customer details list of dictionary is ({"id": 1, "name": "John", "age": 25, "city": "New York"}, {"id": 2, "name": "Anna", "age": 22, "city": "Chicago"}, {"id": 3, "name": "Mike", "age": 30, "city": "Los Angeles"}, {"id": 4, "name": "Sarah", "age": 28, "city": "Houston"}, {"id": 5, "name": "David", "age": 32, "city": "Phoenix"}]

[{"id": 2, "name": "John", "age": 25, "city": "New York"}, {"id": 3, "name": "Mike", "age": 30, "city": "Los Angeles"}, {"id": 4, "name": "Sarah", "age": 28, "city": "Houston"}, {"id": 5, "name": "David", "age": 32, "city": "Phoenix"}]

[{"id": 2, "name": "John", "age": 25, "city": "New York"}, {"id": 3, "name": "Mike", "age": 30, "city": "Los Angeles"}, {"id": 4, "name": "Sarah", "age": 28, "city": "Houston"}, {"id": 5, "name": "David", "age": 32, "city": "Phoenix"}]

(["id": 2, "name": "John", "age": 25, "city": "New York"}, {"id": 3, "name": "Mike", "age": 30, "city": "Los Angeles"}, {"id": 4, "name": "Sarah", "age": 28, "city": "Houston"}, {"id": 5, "name": "David", "age": 32, "city": "Phoenix"}])

(["id": 2, "name": "John", "age": 25, "city": "New York"}, {"id": 3, "name": "Mike", "age": 30, "city": "Los Angeles"}, {"id": 4, "name": "Sarah", "age": 28, "city": "Houston"}, {"id": 5, "name": "David", "age": 32, "city": "Phoenix"}])

(["id": 2, "name": "John", "age": 25, "city": "New York"}, {"id": 3, "name": "Mike", "age": 30, "city": "Los Angeles"}, {"id": 4, "name": "Sarah", "age": 28, "city": "Houston"}, {"id": 5, "name": "David", "age": 32, "city": "Phoenix"}])

(["id": 2, "name": "John", "age": 25, "city": "New York"}, {"id": 3, "name": "Mike", "age": 30, "city": "Los Angeles"}, {"id": 4, "name": "Sarah", "age": 28, "city": "Houston"}, {"id": 5, "name": "David", "age": 32, "city": "Phoenix"}])

(["id": 2, "name": "John", "age": 25, "city": "New York"}, {"id": 3, "name": "Mike", "age": 30, "city": "Los Angeles"}, {"id": 4, "name": "Sarah", "age": 28, "city": "Houston"}, {"id": 5, "name": "David", "age": 32, "city": "Phoenix"}])

20/02/25
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 definition.
2. define the function find_employee_by_id that takes two parameters.
 - a. list of dictionaries (employees) where each dictionary represents an employee record with keys id, name and department.
 - b. An integer (target id), representing the employee ID to be searched
3. iterate through the list.

use a loop for to iterate through each dictionary in the employee list.
and check for matching ID.

4. Return matching Record:

If a match is found return the current dictionary

5. 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 employees:  
        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 printer has printer's logo 'department of Engineering'
 {id: 2, 'name': 'Job', 'department': 'Engineering'}

2: institution entity has printer's logo 'department of Engineering'

3: company staff employee has printer's logo 'department of Engineering'

mbirend

4: company staff employee has printer's logo 'department of Engineering'

5: company staff employee has printer's logo 'department of Engineering'

6: company staff employee has printer's logo 'department of Engineering'

7: company staff employee has printer's logo 'department of Engineering'

8: company staff employee has printer's logo 'department of Engineering'

9: company staff employee has printer's logo 'department of Engineering'

10: company staff employee has printer's logo 'department of Engineering'

11: company staff employee has printer's logo 'department of Engineering'

12: company staff employee has printer's logo 'department of Engineering'

13: company staff employee has printer's logo 'department of Engineering'

14: company staff employee has printer's logo 'department of Engineering'

15: company staff employee has printer's logo 'department of Engineering'

16: company staff employee has printer's logo 'department of Engineering'

17: company staff employee has printer's logo 'department of Engineering'

18: company staff employee has printer's logo 'department of 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 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 represents 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 adjacent elements [i.e. $\text{student}[i]$ and $\text{student}[i+1]$]:

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 student[i]['score'] > student[j+1]['score']:

Swap if the score of the current student is greater than the next student

~~student[i] = student[j+1], student[j+1] = student[i]~~

swapped = True.

If no two elements were swapped, the list is already sorted

if not swapped:

break.

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

Before sorting:

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

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

shoves previous elements to the right

After sorting:

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

shoves previous elements to the left

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

shoves previous elements to the left

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

shoves previous elements to the left

Example usage

```
students = [  
    {'name': 'Alice', 'score': 88},  
    {'name': 'Bob', 'score': 95},  
    {'name': 'Charlie', 'score': 93},  
    {'name': 'Diana', 'score': 85}  
]  
  
print ("Before sorting: ")  
for student in students:  
    print (student)  
  
bubble_sort_scores(students)  
print ("After sorting: ")  
for student in students:  
    print (student)
```

VEL TECH	
EX NO.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
I. (20)	
DATE	15/10/2023

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] + one additional

(entweder vorne, mit) training

unsorted ni trubutz vor

(trubutz) training

(entweder vorne, mit) training

unsorted ni trubutz vor

(trubutz) training

ANSWER

- EX 10
- 1. PERFORMANCE (2)
- 2. RESULTS AND ANALYSIS (3)
- 3. DATA SOURCE (2)
- RECORDS (2)

entweder vorne oder hinterne position vor manchen sind unten : hier

entweder vorne oder hinterne position vor manchen sind unten : hier

Task no: To utilize "functions" concepts in Python programming

Date: 03/08/25

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

Q1) 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: use `s max()` and `s min()`.
6. Print sorted list of grades: use `s sorted()` to sort the grades.
7. Print reversed list of grades: use `s 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", "Bob", "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-names list: ", type(student_names))
```

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-grade)

#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()

6.2. 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 operation (addition, subtraction, multiplication, division).

5. Display result: The program displays the results 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 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.

#greeting

Greeting the user.

```
User-name = "Alice"
```

```
Print ("In Greeting :")
```

```
Print (greet (User-name))
```

Run the main function

```
if __name__ == "__main__":
```

```
    main()
```

Step 4: Open the file in read mode and using read() method print the student details.

Step 5: Using seekoff print the particular student record.

Step 6: Using tell method print the current position of the file.

Step 7: Close the file.

Step 8: Exit.

Program:

```
file = open('student.txt', 'r')  
marks = input("Enter student name: ")
```

file.write(marks)

file.seek(0)

file.read()

file.close()

VEL TECH	
EXPO.	6
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VA VOCE (5)	5
RECORD (5)	
C.I.L (20)	
DATE	15

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

Task 7.1 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("student 1.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 student details with for new")
    file.write(input2)
    file.write("\n")
file = open("student 1.txt", "a")
input3 = input("Enter updated student details\n")
file.write(input3)
file = open("student 1.txt", "a")
input4 = input("Enter updated student details\n")
file.write(input4)
```

output -

student details using fixed read function is

STU 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

29
traversing boundary condition : 29

boundary condition : 29
29 : 29
29 : 29
29 : 29

(`w`, "IIT Institute") : 29

(`w`, "IIT Institute") : 29
(`w`, "IIT Institute") : 29

(`w`, "IIT Institute") : 29

((`I`, "IIT Institute") : 29)

((`O`, "IIT Institute") : 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("In 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:

filenames merge.txt

output: 5,4,8,4.

("v" takes pushit) into -dit

(("i" is pushed onto stack while "int" is taken)) train

("h" is taken) train

("n") train

("o") 1002. dit

("i" is read from stack while "o" is taken) train

("and boy. dit" -nil)

("nil") nil -nil

("end") train

("end") 1002. dit

("i" is written (pushed into stack) without taking) train

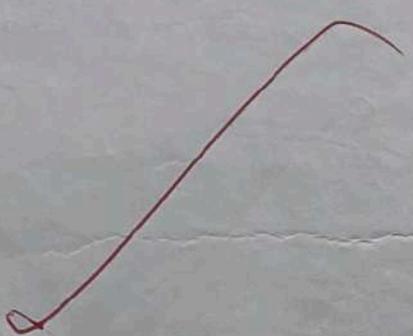
("i" written. dit") train

("int" dit to writing train while both nil) train

("nil. dit")

("") train

"1002. dit"



enitohu kip enitohu kip enitohu kip enitohu kip
was enitohu kip that enitohu kip enitohu kip

all the 1002. dit

Task: 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, lowercase, 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 highlevel 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():

lower_count += 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
Harvard - 71520.0

Ravi - 788-0

Digitized by srujanika@gmail.com

: true("w": "txt-severn") over (new

Digitized by srujanika@gmail.com

enbaar tot dat aantal : enige

Two "x" into zero" rule also

(1) $\text{loss}_t = \text{task}$

卷之三

$$0 = \log v_0 + v_0 w_0$$

~~Test of significance~~

2009-10-23

~~1980-1981~~

1985-1986

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's data

```
students = []
```

```
for i in range (2, n+1):
```

```
    parts = lines[i].strip().split()
```

```
    name = 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 + "-" + 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)	5
TOTAL (20)	15
STUDENT DATE	

15

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

Output:

Enter the starting number: 1

Enter the ending number: 50

Enter the step size: 5

1
6
11
16
21
26
31
36
41
46

NUMBER	EX-NO
1	1
6	6
11	11
16	16
21	21
26	26
31	31
36	36
41	41
46	46

Enter the starting number: 1

Enter the ending number: 50

Enter the step size: 5

Output to screen format: console

(String class) main

String format: System.out.

(FileOutputStream+EOF+CloseWriter) - PrintWriter

(CharacterWriter+PrintWriter) - PrintWriter

PrintWriter.println("Hello world");

[7] = 2 * stepsize

: (N+5, 5) sequence

(Hello, 5) print, [i]print = string

[0] * 5 = 0 * 5 = 0

[5] * 5 = 5 * 5 = 25 * 5 = 125

multiple of 5 sequence

0 = 0

: 0 in range (N)

[1] * 5 = 1 * 5 = 5 * 5 = 25

((C:\Hello, main) + 5) * 5 = 100 * 5 = 500

txt.println("Hello, main") * 5 = 2500

: 250 * ("Hello, main" * 5) = 12500

: 12500 * 5 = 62500

("N" + (main * 5) + "Hello, main") * 5 = 62500

multiple of 5 sequence

: 62500 * 5 = 312500

multiple of 5 sequence

: 312500 * 5 = 1562500

multiple of 5 sequence

: 1562500 * 5 = 7812500

multiple of 5 sequence

: 7812500 * 5 = 39062500

multiple of 5 sequence

: 39062500 * 5 = 195312500

multiple of 5 sequence

: 195312500 * 5 = 976562500

multiple of 5 sequence

: 976562500 * 5 = 4882812500

multiple of 5 sequence

: 4882812500 * 5 = 24414062500

multiple of 5 sequence

: 24414062500 * 5 = 122070312500

multiple of 5 sequence

: 122070312500 * 5 = 610351562500

multiple of 5 sequence

: 610351562500 * 5 = 3051757812500

multiple of 5 sequence

: 3051757812500 * 5 = 15258789062500

multiple of 5 sequence

: 15258789062500 * 5 = 76293945312500

multiple of 5 sequence

: 76293945312500 * 5 = 381469726562500

multiple of 5 sequence

: 381469726562500 * 5 = 1907348632812500

multiple of 5 sequence

: 1907348632812500 * 5 = 9536743164062500

multiple of 5 sequence

: 9536743164062500 * 5 = 47683715820312500

multiple of 5 sequence

: 47683715820312500 * 5 = 238418579101562500

multiple of 5 sequence

: 238418579101562500 * 5 = 1192092895507812500

multiple of 5 sequence

: 1192092895507812500 * 5 = 5960464477539062500

multiple of 5 sequence

: 5960464477539062500 * 5 = 29802322387695312500

multiple of 5 sequence

: 29802322387695312500 * 5 = 149011611938476562500

multiple of 5 sequence

: 149011611938476562500 * 5 = 745058059692382812500

multiple of 5 sequence

: 745058059692382812500 * 5 = 3725290298461914062500

multiple of 5 sequence

: 3725290298461914062500 * 5 = 18626451492309570312500

multiple of 5 sequence

: 18626451492309570312500 * 5 = 93132257461547851562500

multiple of 5 sequence

: 93132257461547851562500 * 5 = 465661287307739257812500

multiple of 5 sequence

: 465661287307739257812500 * 5 = 2328306436538696289062500

multiple of 5 sequence

: 2328306436538696289062500 * 5 = 11641532182693481445312500

multiple of 5 sequence

: 11641532182693481445312500 * 5 = 58207660913467407226562500

multiple of 5 sequence

: 58207660913467407226562500 * 5 = 291038304567337036132812500

multiple of 5 sequence

: 291038304567337036132812500 * 5 = 1455191522836685180664062500

multiple of 5 sequence

: 1455191522836685180664062500 * 5 = 7275957614183325903320312500

multiple of 5 sequence

: 7275957614183325903320312500 * 5 = 36379788070916629516601562500

multiple of 5 sequence

: 36379788070916629516601562500 * 5 = 181898940354583147583007812500

multiple of 5 sequence

: 181898940354583147583007812500 * 5 = 909494701772915737915039062500

multiple of 5 sequence

: 909494701772915737915039062500 * 5 = 4547473508864578689575195312500

multiple of 5 sequence

: 4547473508864578689575195312500 * 5 = 22737367544322893447875976562500

multiple of 5 sequence

: 22737367544322893447875976562500 * 5 = 113686837721614467239379882812500

multiple of 5 sequence

: 113686837721614467239379882812500 * 5 = 5684341886080723361968994445312500

multiple of 5 sequence

: 5684341886080723361968994445312500 * 5 = 28421709430403616809844972226562500

multiple of 5 sequence

: 28421709430403616809844972226562500 * 5 = 142108547152018084049224861132812500

multiple of 5 sequence

: 142108547152018084049224861132812500 * 5 = 710542735760090420246122430664062500

multiple of 5 sequence

: 710542735760090420246122430664062500 * 5 = 3552713678800452101230612153320312500

multiple of 5 sequence

: 3552713678800452101230612153320312500 * 5 = 17763568394002260506153060766601562500

multiple of 5 sequence

: 17763568394002260506153060766601562500 * 5 = 88817841970011302530765303833007812500

multiple of 5 sequence

: 88817841970011302530765303833007812500 * 5 = 444089209850056512653826519165039062500

multiple of 5 sequence

: 444089209850056512653826519165039062500 * 5 = 2220446049250028256326913095825195312500

multiple of 5 sequence

: 2220446049250028256326913095825195312500 * 5 = 1110223024625001412663295654912507812500

multiple of 5 sequence

: 1110223024625001412663295654912507812500 * 5 = 55511151231250007063314780275512539062500

multiple of 5 sequence

: 55511151231250007063314780275512539062500 * 5 = 27755575615625000353165790137755195312500

multiple of 5 sequence

: 27755575615625000353165790137755195312500 * 5 = 1387778780781250001765828950887775976562500

multiple of 5 sequence

: 1387778780781250001765828950887775976562500 * 5 = 6938893903906250000882914750443880445312500

multiple of 5 sequence

: 6938893903906250000882914750443880445312500 * 5 = 34694469519531250000441475250221992226562500

multiple of 5 sequence

: 34694469519531250000441475250221992226562500 * 5 = 173472347597656250000220737625110961132812500

multiple of 5 sequence

: 173472347597656250000220737625110961132812500 * 5 = 867361737988281250000110368812554805664062500

multiple of 5 sequence

: 867361737988281250000110368812554805664062500 * 5 = 4336808689944495000005518440625277402812500

multiple of 5 sequence

: 4336808689944495000005518440625277402812500 * 5 = 216840434497224750000275922031251387012500

multiple of 5 sequence

: 216840434497224750000275922031251387012500 * 5 = 10842021724861237500001379610156256935012500

multiple of 5 sequence

: 10842021724861237500001379610156256935012500 * 5 = 54210108624406187500006898050781253475012500

multiple of 5 sequence

: 54210108624406187500006898050781253475012500 * 5 = 2710505431220309375000344902539062517375012500

multiple of 5 sequence

: 2710505431220309375000344902539062517375012500 * 5 = 135525271561015468750001724512547812506875012500

multiple of 5 sequence

: 135525271561015468750001724512547812506875012500 * 5 = 6776263578050773437500008622562740312534375012500

multiple of 5 sequence

: 6776263578050773437500008622562740312534375012500 * 5 = 338813178902538671875000043112813720625171875012500

multiple of 5 sequence

: 338813178902538671875000043112813720625171875012500 * 5 = 16940658945126933593750000215564068631250859375012500

multiple of 5 sequence

: 16940658945126933593750000215564068631250859375012500 * 5 = 847032947256336679687500001077820343156254296875012500

multiple of 5 sequence

: 847032947256336679687500001077820343156254296875012500 * 5 = 4235164736281683398437500005389101715812521484375012500

multiple of 5 sequence

: 4235164736281683398437500005389101715812521484375012500 * 5 = 2117582368140841699218750000269455085790625107484375012500

multiple of 5 sequence

Date: 10/01/15 Task 3: Implement Python generators and decorators.

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

Q.1: Write a python program that include a generator function to produce a sequence of numbers. The generator should be able to:

a. produce a sequence of 'n' numbers, 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 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.

Q.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: no working suspension or movement needed in either; min. fuel usage hhp

2

clustering outcomes shown following results in terms of
available adhesives not used in maximum transmission. In this
regards, fixed tooth behavior under maximum loadings is similar

to 1

as regards f

and hence

without clustering within

(eg. 0, 1, 2, 3, 4) where movement is limited. Note that at 3 this

: when known oscillations

are relevant at f.

: fixed behaviour is best

(eg. 0, 1, 2, 3, 4) since adhesives will be much more than
interlocking boundary

: mixed behaviour is best

: fixed behaviour is best

adhesive boundary

adhesive boundary

: random

(1, 2, 3, 4, 5, 6) - depends randomly on

fixed = from 0

: fixed + some slippage

slippage by 1

fixed + 1 extra

fixed + 2 extra

fixed + 3 extra

fixed + 4 extra

fixed + 5 extra

fixed + 6 extra

fixed + 7 extra

fixed + 8 extra

fixed + 9 extra

fixed + 10 extra

fixed + 11 extra

fixed + 12 extra

fixed + 13 extra

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:
 - produce current value of the counter yield value
 - increment the counter.
4. Create generator object.
5. Iterate and print values.

8.1 (b) Program.

```
def my_generator(n):  
    # initialize counter  
    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 my_generator():  
        # 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 user's preferences. we can choose to have their messages

Algorithm:

1. Create decorators:
 - Define uppercase-decorator to convert the result of a function.
2. Define functions:
 - Define shout function to receive input.
3. Define greet function:
 - Define greet function what
4. Execute the program.
 - call greet(shout) to print the greetings in uppercase.

OUTPUT:
HI, I AM CREATED BY A FUNCTION PASSED AS AN ARGUMENT

Hi, I am created by a function passed as an argument

Hi, I am created by a function passed as an argument

moreover (d)

(a) whenever you do
whatever you do

and then whatever you do

when this happens when you do something

it's not what you do

but what you do is what you do

because it's not what you do it's what you do

so it's not what you do it's what you do

so it's not what you do it's what you do

so it's not what you do it's what you do

so it's not what you do it's what you do

Program :

```
def uppercase_decorator(func):
```

```
    def wrapper(text):
```

```
        return func(text).upper()
```

```
    return wrapper
```

```
def lower_case_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):
```

greet = func("Hi, I am created by a function passed as an argument")

```
print(greet)
```

```
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

Result: Thus the python program to implement python generator and decorators was successfully 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.

(empty) list - empty list
(text) recursive list
empty (list) recursive
empty list recursive
(text) tail list
empty list recursive
empty list recursive
(empty) tree list
(empty) tree list
(empty) tree list
(empty) tree list

EX NO
1
2
3
4

overhead terminal or memory address allocation error
due to stack overflow from linked list has occurred
during list traversal

Date: 24/09/25

Task 9: Implement Exception and Exceptional handling in Python.

Aim: To implement exceptions and Exceptional handling in Python

Algorithm:

1. Starts the program
2. Initializes a list of grades (e.g. [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, catches the index error.

Programm:

Initialize the list of grades

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

Displays 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.

if index < 0 or index > len(grades) - 1:
 print("Invalid index. Please enter a valid range.")

print("Invalid index. Please enter a valid index.")

Handle the case where the input is not an integer

print("Invalid input. Please enter a numerical index.")



outputs ~~transform into working numbers~~ - part
numbers

Enter the numerator: 10

Enter the denominator: 0

ERROR!

Error: division by zero is not allowed

working with

(ex: 10/2) or where both numbers

are not zero the ratio of numbers

representing data that is equal or 1 (perfectly

equal ratios, same ratios (1:1), etc.)

: working

above is all the definitions

(1:1, 2:2, 3:3, etc.) - when

tell where not equal to

(where "not equal") tell

(not equal) when both numbers are not equal to each other

(ex: 10/3) or when both numbers are not equal to each other

(ex: 10/3) or when both numbers are not equal to each other

(ex: 10/3) or when both numbers are not equal to each other

(ex: 10/3) or when both numbers are not equal to each other

(ex: 10/3) or when both numbers are not equal to each other

(ex: 10/3) or when both numbers are not equal to each other

(ex: 10/3) or when both numbers are not equal to each other

(ex: 10/3) or when both numbers are not equal to each other

(ex: 10/3) or when both numbers are not equal to each other

(ex: 10/3) or when both numbers are not equal to each other

(ex: 10/3) or when both numbers are not equal to each other

(ex: 10/3) or when both numbers are not equal to each other

(ex: 10/3) or when both numbers are not equal to each other



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(f"Result: {result}")

    except ZeroDivisionError:
        # Handle division by zero error
        print("Error: Division by zero is not allowed")

    # Handle invalid input that is not a number
    # print('Error: please enter valid numbers.')
    # call def function to execute the division operation
    # divide_numbers()
```

outputs: input validation makes program more robust
Enter a number : 15
Exception occurred: Invalidage.

Description of exception: Input validation makes program more robust.
Input validation makes program more robust by catching invalid inputs and providing feedback.

No input validation makes program less robust.
Program - which has no validation - can accept any input from user.

Without validation makes program less robust.
Program - which has no validation - can accept any input from user.

Validation makes program more robust.
Program - which has validation - can accept only valid inputs.

Validation makes program more robust.
Program - which has validation - can accept only valid inputs.

Validation makes program more robust.
Program - which has validation - can accept only valid inputs.

Validation makes program more robust.
Program - which has validation - can accept only valid inputs.

Validation makes program more robust.
Program - which has validation - can accept only valid inputs.

Validation makes program more robust.
Program - which has validation - can accept only valid inputs.

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 InvalidAgeException(Exception):
    "Raised when the 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 InvalidAgeException
    else:
        print("Eligible to vote")
except InvalidAgeException:
    print("Exception occurred : 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
GRADE	A

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

Date: 8/10/25

Task 10: Use matplotlib module for plotting in python

python

Aim: To use matplotlib module for plotting in python

Problem 10.1: Write a python program to display bar chart of popularity of programming languages.

Sample data:

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

Sample output:

Algorithm:

1. Define two lists for programming languages and their popularity respectively.
2. Find the maximum popularity value in the list.
3. 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, 7.7, 6.7]
plt.bar(languages, popularity, color='blue')
plt.title('Popularity of programming Languages')
plt.xlabel('Popularity')
plt.show()
```

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

Sample data:

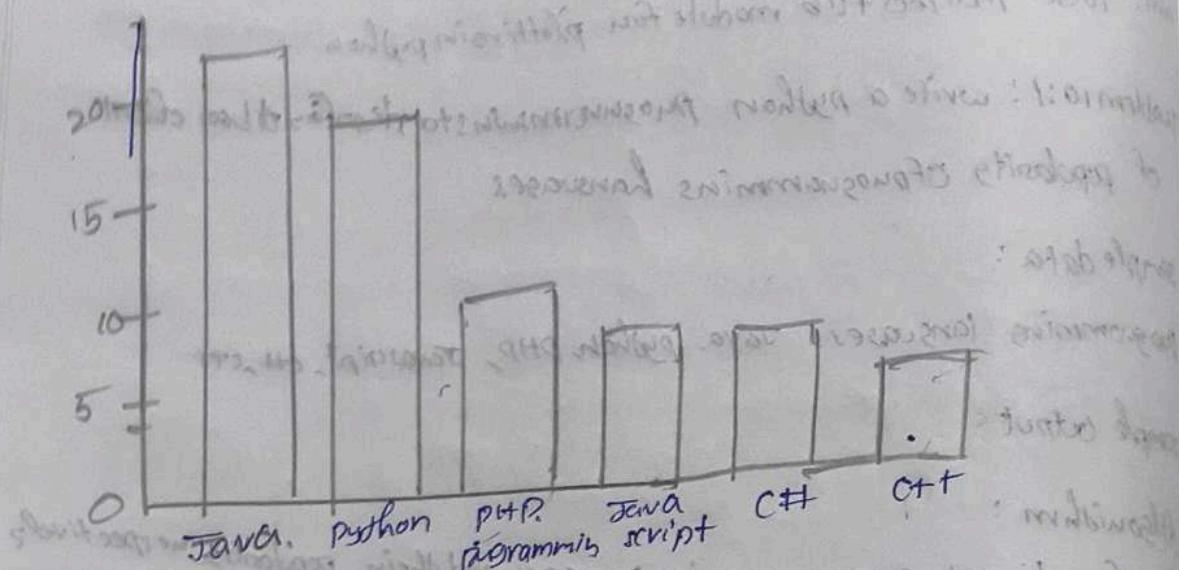
Programming languages: Java, Python, PHP, JavaScript, C++, C#.

Popularity: 22.2, 17.6, 8.8, 8.7, 7.7, 6.7.

Sample output:

output: (mitte) und einem diktatorischen Stil: 01/2020

Markus



diktatorisch Noten gibt

Platz an jedem Tisch zu tun

(z.B. 1., 2., 3., 4., 5., 6., 7., 8.) = 8 Gruppen

(z.B. S. 1., S. 2., S. 3., S. 4., S. 5.) = 5 Gruppen

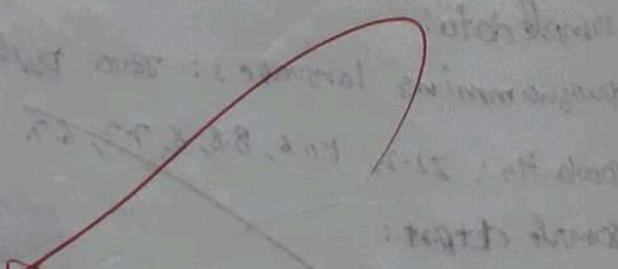
(z.B. jeder Schüler hat einen Platz = 100% der Gruppen zuordnen)

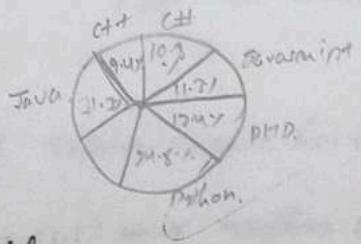
(Klassenzimmer) 100% der Gruppen

(Tische + R

Examen zu überzeugen,

etwa 100% der Gruppen





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
```

Step 1:

```
languages = ['Java', 'python', 'PHP', 'Javascript', 'C#', 'C/C++']
```

```
popularity = [22.2, 17.6, 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(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)	5
Total (20)	15
GRADE	A

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

output :

Hello, world!

code :

multiple threads can access same memory at the same time
without disturbing each other's work. Using a mutex
blocks, signals, not threads split up the
processing among them.

Process takes up disk, memory, CPU

[CPU, RAM, memory, CPU, memory, CPU] : processes
[CPU, RAM, memory, CPU, memory, CPU] : threads

Each thread has its own stack, memory = shared, communication = signal

(coordination communication between threads) threads
(thread safe communication) shared, no
signals

(multithreaded)

HOSTNAME	
1	0.0.0.1
2	0.0.0.2
3	0.0.0.3
4	0.0.0.4

multiple host stations share system resources - threads
multiple threads have memory between them

Date: 15/10/25

Topic: - we learnt module for redesign.

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()D.
6. start the mainloop.

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", 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()
```

Tutorial 2: Write a Python GUI program to create three single text-boxes to accept a value from the user using tkinter module.

Name

username

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 for submit.
5. Get the values.
6. Close the main window when the button is clicked.

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 get the values entered in the text boxes

~~def get_values():~~

~~val1 = entry1.get()~~

~~val2 = entry2.get()~~

~~val3 = entry3.get()~~

~~print("Value 1: ", val1)~~

~~print("Value 2: ", val2)~~

~~print("Value 3: ", 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)	5
TOTAL (20)	15
SIGN WITH DATE	

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

Tues 12 - simulate蛇 game concepts using pygame
date: 15/10/25

Aim: To simulate蛇 game concepts using pygame

make game:

problem 1: write a python program to create a snake game using pygame
practices.

conditions:

1. set the window size

2. create a snake

3. make the snake

4. make the snake move in the directions when left right down and up keys is pressed

5. If the snake hits the window, then over.

Algorithm:

1. Import pygame package and initialize it.

2. Define window size and title.

3. Create a snake which initializes the snake position, color and movement.

4. Create a function to check if the snake collides with the window or not 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)

initializing pygame

pygame.init()

Initialize game window

pygame.display.set_caption("Reversi for two snakes")

game_window = pygame.display.set_mode((window-X, window-Y))

```
#FPS (frames per second) controlled by 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(0, window_x // 10) * 10, random.randrange(0, window_y // 10) * 10]
fruit_spawn = True
#setting default snake direction towards right
direction = 'right'
change_to = direction
#initial score
score = 0
#display score function
def show_score(choice, color, font_size):
    #creating font object score_font
    score_font = pygame.font.SysFont('times new roman', 50)
    #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(f'your score is : {score}', True, red)
        #create a rectangular object for the text
        game_over_rect = game_over_surface.get_rect()
        #surface object
        game_over_rect.midtop = (window_x / 2, window_y / 2)
        #bit 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
        #exit the program with

```

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

If 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 the score

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:

: justify

o: score

output:

score 50



Chlorophyll - green in leaves
chlorophyll - green - plant
- green - green - green - plant

green - green - green - plant

'green' - ok - good

green - green - green - plant

'green' - ok - good

chlorophyll - green - green -

chlorophyll - green - green -

chlorophyll - green -

'green' - ok - good

green - green - green -

'green' - green -

green - green - green -

green - green - green -

green - green - green -

chlorophyll - green - green -

```

fruit-position [random.randrange(1,(windowx//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[0], 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 snake-body[1:]:
    if snake-position[0] == block[0] and snake-position[1] == block[1]:
        game over()

# Display score continuously
show-score('white', 'times new roman', 20)

# Refresh, game screen
pygame.display.update()

# Frame per second / Refresh rate
fps = 60 // (snake-speed)

```

problem 2: write a python program to develop a chess 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 initial state
6. Draw the board and pieces on the screen
7. Start the game loop.

Program:

import pygame

Initialize 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 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)

Define function to draw the pieces.

def draw_board():

piece_images = {

'r': pygame.image.load('image/rook.png'),

'n': pygame.image.load('image/knight.png')

'b': pygame.image.load('image/bishop.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 initial state of the board

board = [[r, n, b, k, ' ', b, n, r],

[P, p, P, p, P, p, P, P],

[., ., ., ., ., ., ., .],

[., ., ., ., ., ., ., .],

[., ., ., ., ., ., ., .]]

output:

(Br.-W.M.) shown the following names - Anna

$$(0,0,0) = \text{black}$$

(See *Notes* - 1911)

(0.05, 5%) μ g/ml

without without evidence

(7) broad-winkled

(b) *senior i wing ut*

(ၬ) အကျဉ်းလျှော့

الآن = ٢٠١٩ - ٢٠١٨

मुख्य - विद्या = विद्यालय

ପ୍ରକାଶ ମାଟେଲି - ଓହାରାପୁଣି

卷之六

161 broad - 100 335

(Eng. name) book, note, writing, letter, etc.

• ④ զնո՞ւ ո՞ւ առ ո՞ւ

:(८) विषय नी लो वो

[10] [www] board = 3331q

二、一九四九年

[1951] उत्तरी राज्यों के विभिन्न विधानसभा चुनावों में विजयी

(08.08.08) WED. 08 AUGUST 2008

Exercises

W. W. Denslow

Oct 19 1951

```

[ 'P', 'P', 'P', 'P', 'P', 'P', 'P' ]
[ 'R', 'N', 'B', 'Q', 'K', 'B', 'R' ]
]
# draw the board and pieces
draw_board()
draw_pieces(board)

```

start the game.

```

while True:
    for event in pygame.event.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 ps game. is executed and verified successfully.

Completed