26/9/25

Task 9: Implement Exception and Exception handling in python

a) Division Error handling

Aim :- To implement exception handling in python using try except finally block to manage division operation and handles division by zero errors.

Algorithm :-

1. prompt the user to input numerator & denominator
2. use a try block to peryorm the division
3. Catch ZeroDivisionError in the except & display an error message.
4. use a finally block to print a message

program :-

```
try:
    a=float(input("Enter numerator:"))
    b= float(input("Enter denominator:"))
    print("Result:", a/b)
except ZeroDivisionError:
    print("Error: Division by zero is not allowed.")
finally:
    print("Execution complete")
```

Result :-

Thus, a python program to implement exception handling & division by zero error is executed

output :-

```
Enter numerator : 10
Enter denominator : 2
Result : 5.0
Execution complete
```

Task 9(b): Custom Exception for Age validation 26/9/25

Aim: To validate a person's age for voting eligibility using a user defined exception in python

Algorithm:

1. Define a custom exception InvalidAgeError
2. prompt the user to enter their age
3. If age is less than 18, raise InvalidAgeError
4. otherwise, print "Eligible to vote"
5. Handle the exception and display the message.

Program:

```
class InvalidAgeError(Exception):
    pass

try:
    age = int(input("Enter your age: "))
    if age < 18:
        raise InvalidAgeError("Age must be atleast 18 to vote.")
    print("Eligible to vote.")
except InvalidAgeError as e:
    print("Error:", e)
```

VELTECH	
EX No.	9
PERFORMANCE (5)	5
RESULT AND ANALYSIS (3)	5
VIVA VOCE (3)	5
RECORD (1)	1
TOTAL (15)	15
SIGN WITH DATE	

Result: Thus, python program to validate a person's age for voting eligibility using user defined exception is executed.

Sample output:

case 1:- Valid Age

Enter your age : 25
Eligible to vote.

case 2: Invalid Age:

Enter your age : 15
Error : Age must be at least 18 to vote

Task 12 :- Simulate Gaming concept using python 17/10/25

Aim :- To Simulate a basic car dodging game using pygame with movement, obstacle, collision detection and game over logic

Algorithm :-

- Initialize a pygame and Create a game window.
- Draw a car (rectangle) that moves left/right with arrow keys.
- Generate falling obstacles.
- update position and check for collision
- Display "Gameover" on collision and Exit.

program code

```
import pygame, random, sys
pygame.init()

width, height = 400, 600
win = pygame.display.set_mode((width, height))
pygame.display.set_caption("Car Dodging Game")
clock = pygame.time.clock()

WHITE, RED, BLUE = (255,255,255), (255,0,0), (0,0,255)
car = pygame.rect(180, 500, 40, 60)
enemy = pygame.Rect(random.randint(0,360), 0, 40, 60)
Speed = 5

While True:
    win.fill(WHITE)
    for event in pygame.event.get():
    if event.type == pygame.QUIT: sys.exit()

    Keys = pygame.Key.get_pressed()
    if Keys[pygame.K_LEFT] and car.Left > 0: car.
                                    move-up(-5,0)
```

output :

When you run this program :

A black game window opens

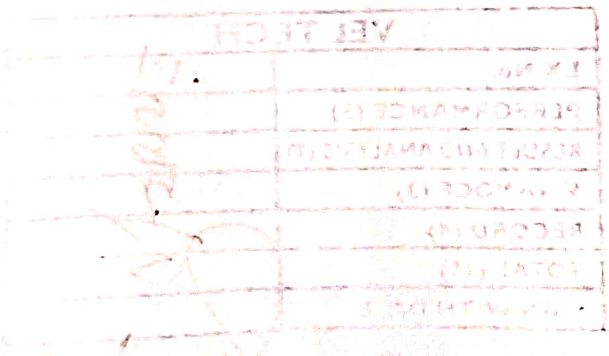
A blue car rectangle appears at bottom

Red blocks fall from the top

You can move the car left & right using arrow keys.

If your car hits an obstacle, the screen displays :

"Game over!" for 2 seconds and then exist



```
if keys[pygame.K_RIGHT] and car.right < width:
        car.move_ip(5,0)

enemy.move_ip(0, speed)
if enemy.top > height:
    enemy.top = 0
    enemy.left = random.randint(0, 360)

if car.colliderect(enemy):
    font = pygame.font.SysFont(None, 50)
    text = font.render("Game over", True, RED)
    win.blit(text, (120,250))
    pygame.display.update()
    pygame.time.wait(2000)
    sys.exit()

pygame.draw.rect(win, BLUE, car)
pygame.draw.rect(win, RED, enemy)
pygame.display.update()
clock.tick(30)
```

VELTECH	
EX No	12
PERFORMANCE (5)	5
RESULT AND ANALYSIS (3)	5
VIVA VOCE (3)	5
RECORD (4)	5
TOTAL (15)	15
SIGN WITH DATE	

Result: Thus, python program to simulate Gaming concept is executed