Karthick VM

Batch – CIS 1.3

Mock Assessment 2 -Python

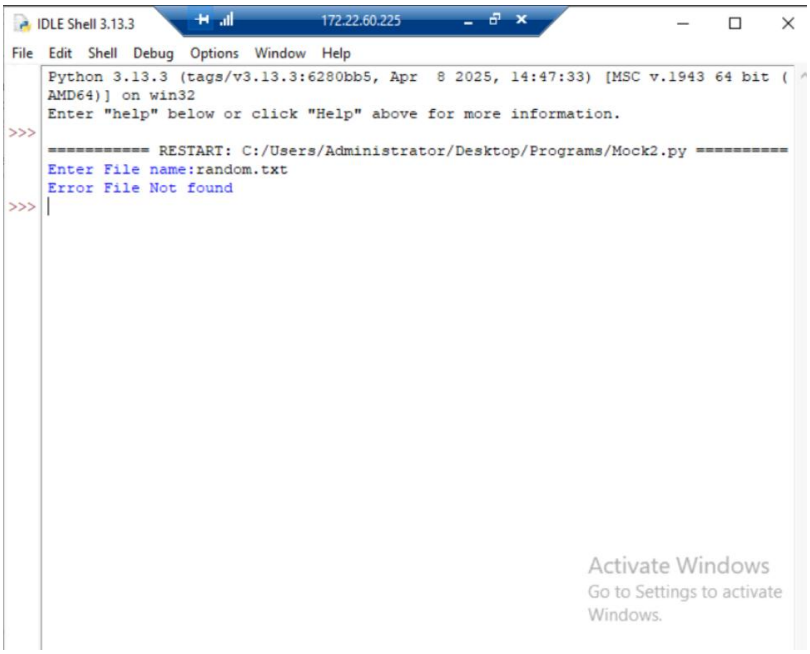
Task1:

Handling Multiple Exceptions with Input and Files Your system accepts input and reads files, and needs to manage potential errors gracefully.

Tasks:

1. Prompt the user to enter a file name and try to open it in read mode. o If the file doesn't exist, catch FileNotFoundError and print: "Error: File not found."
    - Program to open file and error handling if file not found

- Except block is initiated as file is not found

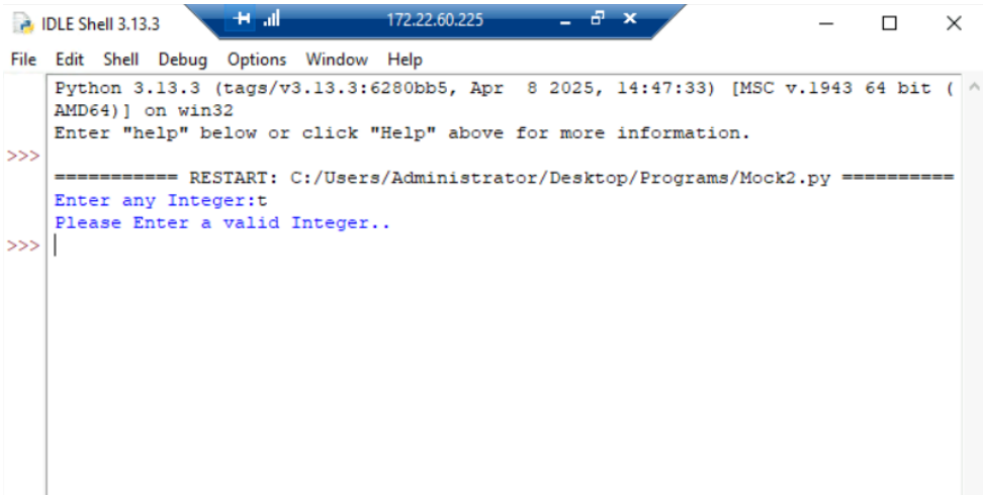


- Output is printed when actually the file is found.

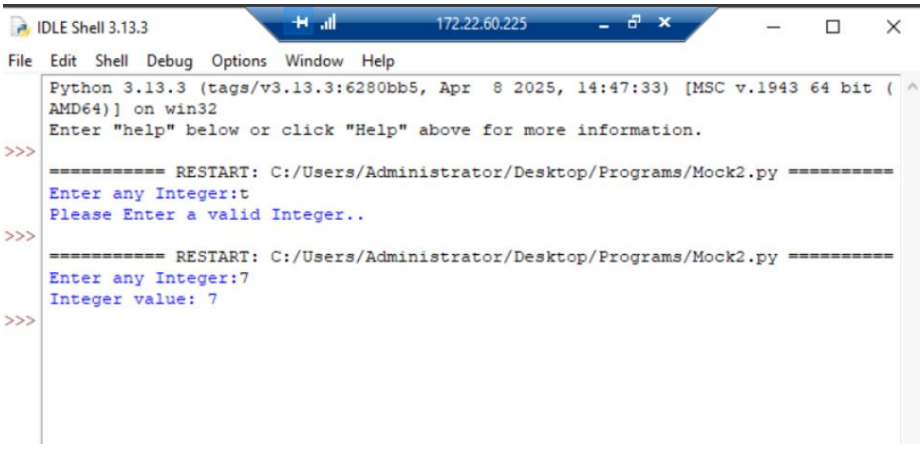
2. Ask the user to input an integer.
   o If a non-integer is entered, catch ValueError and print: "Error: Please enter a valid integer."
   - program

```
try:
    val=int(input("Enter any Integer:"))
    print("Integer value:",val)
except ValueError as e:
    print("Please Enter a valid Integer..")
```


   - except block is initiated as char value is given as input



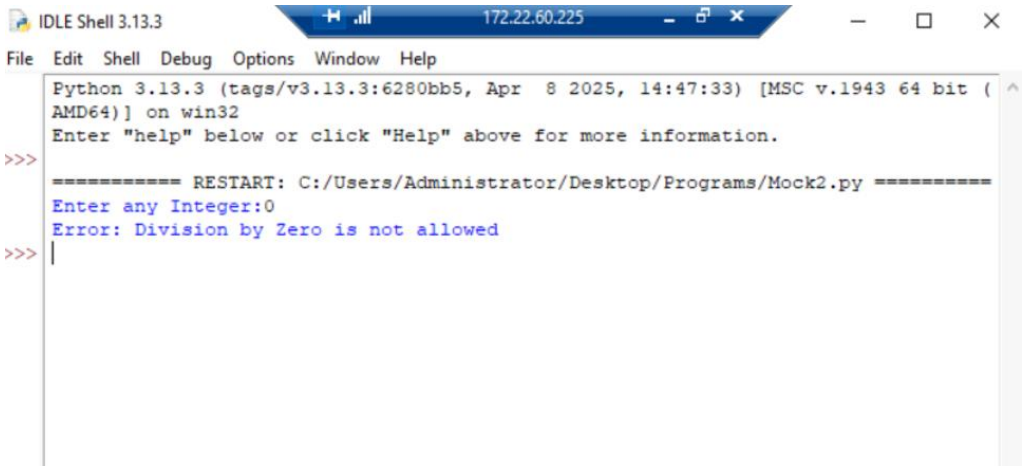
   - Value is Printed as integer value is given as input.



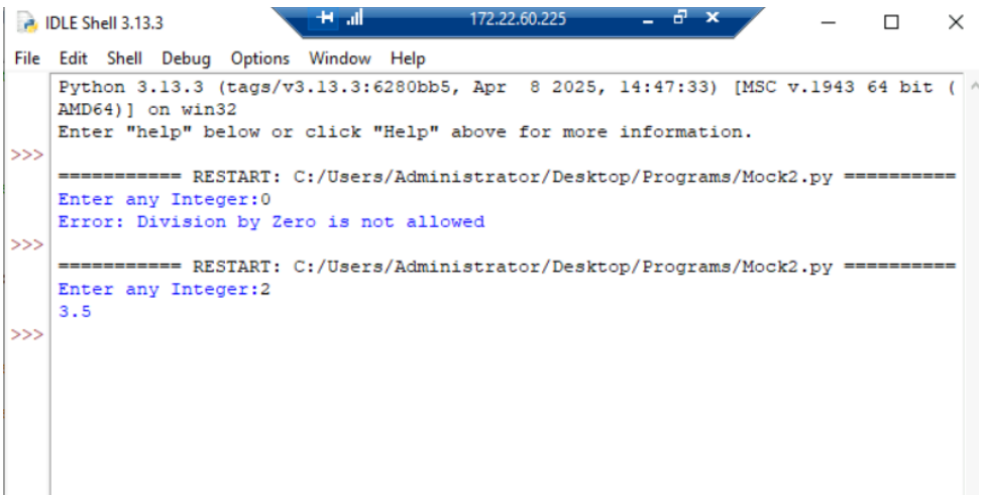
3. Ask the user for a divisor and perform a division using the previously entered integer. o If division by zero occurs, handle ZeroDivisionError and print: "Error: Division by zero is not allowed."
    - program

```
try:
    val=int(input("Enter any Integer:"))
    re=7/val
    print(re)
except ValueError as e:
    print("Please Enter a valid Integer..")
except ZeroDivisionError as e:
    print("Error: Division by Zero is not allowed")
```


- Except block is initiated as we try to divide using zero.



- When int value is given the value is divided by given value.



Task2:

Student Data Management Create a program in Python to manage student data. The program should meet the following requirements:

• Menu-Driven Interface: Use a while loop to present a menu with three options:

1. Add a new student.
2. Calculate a student's percentage.
3. Exit the program.

```
data={}
i=True
while(i):
    print("1.Add a New Student\n 2.Calculate Student Percentage\n 3.Exit")
    inp=int(input("Enter Choice:"))
    if(inp==1):
        Studdata(data)
    if(inp==2):
        percentage(data)
    if(inp==3):
        i=False
```


Ln: 13 Col: 24

• Add Student Functionality:

o Create a function that prompts the user for a student's name.

o Use the name as a key to store data in a dictionary.

o Prompt the user to enter the student's marks for three subjects.

o Store these marks as a list which will be the value associated with the student's name in the dictionary.

- Function to add student data in dictionairy “data”.
- There Student name acts as key value and marks are added as a list.

```
def Studdata(data):
    name=input("Enter name:")
    marks=[]
    for i in range(3):
        mark=int(input("Enter mark:"))
        marks.append(mark)
    data[name]=marks
    print("Student Added Successfully !\n"data)
```

Calculate Percentage Functionality:

o Create a separate function that asks the user to enter a student's name. o Retrieve the student's marks from the dictionary using the provided name

o Calculate the student's percentage based on the three subjects. o Display the student's name and their calculated percentage on the screen.

- Function to calculate the percentage of student.

```
def percentage(data):
    name=input("Enter name:")
    if name in data:
        marks=data[name]
        total=sum(marks)
        percentage=(total/300)*100
        print("Name:",name)
        print("Percentage:",percentage)
    else:
        print("Not Found..")
```

- Output:

Task3: Write a python program to find the sum of n natural numbers.