(https://www.darshan.ac.in/)

Python Programming - 2101CS405

22010101142

SACHIN PATADIYA

Lab - 9

Exception Handling

A

01) WAP to handle divide by zero exception.

In [1]:
```
def divide_numbers(dividend, divisor):
    try:
        result = dividend / divisor
        print("Result of division:", result)
    except ZeroDivisionError:
        print("Error: Division by zero is not allowed.")

dividend = 10
divisor = 0

divide_numbers(dividend, divisor)
```

```
Error: Division by zero is not allowed.
```

02) Write a Python program that inputs a number and generates an error message if it is not a number.

In [3]:
```
def check_number():
    try:
        number = float(input("Enter a number: "))
        print("Input is a valid number:", number)
    except ValueError:
        print("Error: Input is not a valid number.")

check_number()
```

```
Enter a number: 22
Input is a valid number: 22.0
```

03) WAP to handle file not found Exception

In [6]:
```
def read_file(file_name):
    try:
        with open(file_name, 'r') as file:
            content = file.read()
            print("File content:\n", content)
    except FileNotFoundError:
        print("Error: File not found or path is incorrect.")

file_name = input("Enter the file name: ")

read_file(file_name)
```

```
Enter the file name: temp.txt
Error: File not found or path is incorrect.
```

04) WAP to handle type Exception.

In [ ]:
```

```

05) WAP to demonstrate valueError and indexError with example.

In [9]:
```
def demonstrate_value_error():
    try:
        # Try to convert a non-integer string to an integer
        value = int("abc")
    except ValueError:
        print("Error: Unable to convert 'abc' to an integer.")

def demonstrate_index_error():
    try:
        # Try to access an index that doesn't exist in a list
        my_list = [1, 2, 3]
        element = my_list[5]
    except IndexError:
        print("Error: Index out of range. The list has fewer elements.")

# Demonstrate ValueError
print("Demonstrating ValueError:")
demonstrate_value_error()

# Demonstrate IndexError
print("\nDemonstrating IndexError:")
demonstrate_index_error()
```

```
Demonstrating ValueError:
Error: Unable to convert 'abc' to an integer.

Demonstrating IndexError:
Error: Index out of range. The list has fewer elements.
```

06) WAP to domonstrate else and finally block.

In [10]:
```
def divide_numbers(dividend, divisor):
    try:
        result = dividend / divisor
    except ZeroDivisionError:
        print("Error: Division by zero is not allowed.")
    else:
        print("Division successful! Result:", result)
    finally:
        print("This will always execute, regardless of whether an exceptior

dividend = 10
divisor = 2
divide_numbers(dividend, divisor)

print("\n")

dividend = 10
divisor = 0
divide_numbers(dividend, divisor)
```

```
Division successful! Result: 5.0
This will always execute, regardless of whether an exception occurred or
not.


Error: Division by zero is not allowed.
This will always execute, regardless of whether an exception occurred or
not.
```

07) Create a short program that prompts the user for a list of grades separated by commas. Split the string into individual grades and use a list comprehension to convert each string to an integer. You should use a try statement to inform the user when the values they entered cannot be converted.

In [12]:
```
def convert_grades(grades_str):
    try:
        grades = [int(grade) for grade in grades_str.split(',')]
        print("Converted grades:", grades)
    except ValueError:
        print("Error: One or more values entered are not valid grades.")

grades_input = input("Enter grades separated by commas: ")

convert_grades(grades_input)
```

```
Enter grades separated by commas: a,b,
Error: One or more values entered are not valid grades.
```

Type *Markdown* and LaTeX: α^2

B

01) WAP to Raising User Generated Exception.

In [14]:
```
class CustomException(Exception):
    def __init__(self, message):
        self.message = message

def check_number(number):
    if number < 0:
        raise CustomException("Number must be positive.")

try:
    number = int(input("Enter a positive number: "))
    check_number(number)
    print("You entered a positive number.")
except CustomException as e:
    print("Custom Exception:", e.message)
except ValueError:
    print("Error: Please enter a valid integer.")
```

```
Enter a positive number: sachin
Error: Please enter a valid integer.
```

02) WAP to raise your custom Exception.

In [19]:
```
class MyCustomException(Exception):
    def __init__(self,message):
        self.message=message

def devideByZero(a,b):
    if b==0:
        MyCustomException("You can not devide ")
try:
    devideByZero(10,0)
except MyCustomException as e:
    print(e)
```