Darshan UNIVERSITY

योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 10

Charmi Bhalodiya

23010101020

4B 448 8th batch

Exception Handling

01) WAP to handle following exceptions:

1. ZeroDivisionError
2. ValueError
3. TypeError

Note: handle them using separate except blocks and also using single except block too.

In [8]:
```
try:
    num1 = int(input("Enter the first number: "))
    num2 = int(input("Enter the second number: "))
    result = num1 / num2
    print("Result:", result)
except ZeroDivisionError:
    print("Error: Division by zero is not allowed.")
except ValueError:
    print("Error: Invalid input! Please enter integers only.")
except TypeError:
    print("Error: Type mismatch encountered.")
except Exception as e:
    print(f"An unexpected error occurred: {e}")
```

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

02) WAP to handle following exceptions:

1. IndexError
2. KeyError

In [10]:
```
try:
    num1 = int(input("Enter the first number: "))
    num2 = int(input("Enter the second number: "))
    result = num1 / num2
    print("Result:", result)

    my_list = [1, 2, 3]
    index = int(input("Enter an index to access: "))
    print("List value:", my_list[index])

    my_dict = {"a": 1, "b": 2}
    key = input("Enter a dictionary key to access: ")
    print("Dictionary value:", my_dict[key])
```

```
except ZeroDivisionError:
    print("Error: Division by zero is not allowed.")
except ValueError:
    print("Error: Invalid input! Please enter integers only.")
except TypeError:
    print("Error: Type mismatch encountered.")
except IndexError:
    print("Error: List index out of range.")
except KeyError:
    print("Error: Key not found in dictionary.")
except Exception as e:
    print(f"An unexpected error occurred: {e}")
```

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

03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

In [12]:
```
try:
    num1 = int(input("Enter the first number: "))
    num2 = int(input("Enter the second number: "))
    result = num1 / num2
    print("Result:", result)

    my_list = [1, 2, 3]
    index = int(input("Enter an index to access: "))
    print("List value:", my_list[index])

    my_dict = {"a": 1, "b": 2}
    key = input("Enter a dictionary key to access: ")
    print("Dictionary value:", my_dict[key])

    file_name = input("Enter the filename to open: ")
    with open(file_name, 'r') as file:
        print(file.read())

    module_name = input("Enter the module name to import: ")
    __import__(module_name)

except ZeroDivisionError:
    print("Error: Division by zero is not allowed.")
except ValueError:
    print("Error: Invalid input! Please enter integers only.")
except TypeError:
    print("Error: Type mismatch encountered.")
except IndexError:
    print("Error: List index out of range.")
except KeyError:
    print("Error: Key not found in dictionary.")
except FileNotFoundError:
    print("Error: The specified file was not found.")
except ModuleNotFoundError:
    print("Error: The specified module could not be found.")
except Exception as e:
    print(f"An unexpected error occurred: {e}")
```

```
Error: Invalid input! Please enter integers only.
```

04) WAP that catches all type of exceptions in a single except block.

In [13]:
```
try:
    num1 = int(input("Enter the first number: "))
    num2 = int(input("Enter the second number: "))
    result = num1 / num2
    print("Result:", result)

    my_list = [1, 2, 3]
    index = int(input("Enter an index to access: "))
    print("List value:", my_list[index])

    my_dict = {"a": 1, "b": 2}
    key = input("Enter a dictionary key to access: ")
    print("Dictionary value:", my_dict[key])

    file_name = input("Enter the filename to open: ")
    with open(file_name, 'r') as file:
        print(file.read())

    module_name = input("Enter the module name to import: ")
```

```
    __import__(module_name)

except Exception as e:
    print(f"An error occurred: {e}")
```

```
Result: 0.5
List value: 2
An error occurred: '1'
```

05) WAP to demonstrate else and finally block.

In [14]:
```
try:
    num1 = int(input("Enter the first number: "))
    num2 = int(input("Enter the second number: "))
    result = num1 / num2
    print("Result:", result)

    my_list = [1, 2, 3]
    index = int(input("Enter an index to access: "))
    print("List value:", my_list[index])

    my_dict = {"a": 1, "b": 2}
    key = input("Enter a dictionary key to access: ")
    print("Dictionary value:", my_dict[key])

    file_name = input("Enter the filename to open: ")
    with open(file_name, 'r') as file:
        print(file.read())

    module_name = input("Enter the module name to import: ")
    __import__(module_name)

except Exception as e:
    print(f"An error occurred: {e}")
else:
    print("Execution completed successfully without any exceptions.")
finally:
    print("Execution of the try-except block is finished.")
```

```
Result: 5.0
List value: 3
Dictionary value: 1
An error occurred: [Errno 2] No such file or directory: 'Downloads/29012025112910PM/Python Programming - Lab - 1
0.ipynb'
Execution of the try-except block is finished.
```

06) 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 [15]:
```
try:
    grades_input = input("Enter a list of grades separated by commas: ")
    grades = [int(grade.strip()) for grade in grades_input.split(",")]
    print("Grades list:", grades)
except ValueError:
    print("Error: Please enter only numeric values separated by commas.")
else:
    print("Grades successfully converted.")
finally:
    print("Execution finished.")
```

```
Grades list: [2]
Grades successfully converted.
Execution finished.
```

07) WAP to create an udf divide(a,b) that handles ZeroDivisionError.

In [16]:
```
def divide(a, b):
    try:
        result = a / b
        return result
    except ZeroDivisionError:
        return "Error: Division by zero is not allowed."

try:
    grades_input = input("Enter a list of grades separated by commas: ")
```

```
    grades = [int(grade.strip()) for grade in grades_input.split(",")]
    print("Grades list:", grades)
except ValueError:
    print("Error: Please enter only numeric values separated by commas.")
else:
    print("Grades successfully converted.")
finally:
    print("Execution finished.")
```

```
Grades list: [3]
Grades successfully converted.
Execution finished.
```

08) WAP that gets an age of a person form the user and raises ValueError with error message: "Enter Valid Age" :

If the age is less than 18.

otherwise print the age.

In [17]:
```
def divide(a, b):
    try:
        result = a / b
        return result
    except ZeroDivisionError:
        return "Error: Division by zero is not allowed."

try:
    grades_input = input("Enter a list of grades separated by commas: ")
    grades = [int(grade.strip()) for grade in grades_input.split(",")]
    print("Grades list:", grades)
except ValueError:
    print("Error: Please enter only numeric values separated by commas.")
else:
    print("Grades successfully converted.")
finally:
    print("Execution finished.")

try:
    age = int(input("Enter your age: "))
    if age < 18:
        raise ValueError("Enter Valid Age")
    print("Your age is:", age)
except ValueError as e:
    print("Error:", e)
```

```
Grades list: [10]
Grades successfully converted.
Execution finished.
Your age is: 18
```

09) WAP to raise your custom Exception named InvalidUsernameError with the error message : "Username must be between 5 and 15 characters long":

if the given name is having characters less than 5 or greater than 15.

otherwise print the given username.

In [18]:
```
class InvalidUsernameError(Exception):
    pass

def divide(a, b):
    try:
        result = a / b
        return result
    except ZeroDivisionError:
        return "Error: Division by zero is not allowed."

try:
    grades_input = input("Enter a list of grades separated by commas: ")
    grades = [int(grade.strip()) for grade in grades_input.split(",")]
    print("Grades list:", grades)
except ValueError:
    print("Error: Please enter only numeric values separated by commas.")
else:
    print("Grades successfully converted.")
finally:
    print("Execution finished.")

try:
    age = int(input("Enter your age: "))
```

```
    if age < 18:
        raise ValueError("Enter Valid Age")
    print("Your age is:", age)
except ValueError as e:
    print("Error:", e)

try:
    username = input("Enter your username: ")
    if len(username) < 5 or len(username) > 15:
        raise InvalidUsernameError("Username must be between 5 and 15 characters long")
    print("Your username is:", username)
except InvalidUsernameError as e:
    print("Error:", e)
```

```
Error: Please enter only numeric values separated by commas.
Execution finished.
Your age is: 20
Your username is: charmi
```

10) WAP to raise your custom Exception named NegativeNumberError with the error message : "Cannot calculate the square root of a negative number" :

if the given number is negative.

otherwise print the square root of the given number.

In [1]:
```
import math

class NegativeNumberError(Exception):
    """Custom exception for negative numbers."""
    pass

def calculate_square_root(num):
    if num < 0:
        raise NegativeNumberError("Cannot calculate the square root of a negative number")
    return math.sqrt(num)

try:
    number = float(input("Enter a number: "))
    result = calculate_square_root(number)
    print(f"The square root of {number} is {result}")
except NegativeNumberError as e:
    print("Error:", e)
except ValueError:
    print("Invalid input! Please enter a valid number.")
```

```
The square root of 10.0 is 3.1622776601683795
```

In [ ]: