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.

```
# Handling exceptions using separate except blocks
try:
    a = int(input("Enter a number: "))
    b = int(input("Enter another number: "))
    result = a / b
    print("Result:", result)

    lst = [1, 2, 3]
    index = int(input("Enter index to access: "))
    print("Element:", lst[index])

except ZeroDivisionError:
    print("Error: Division by zero is not allowed.")
except ValueError:
    print("Error: Invalid input! Please enter an integer.")
except TypeError:
    print("Error: Unsupported operation between different types.")
```

```
Enter a number:  10
Enter another number:  0

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

```
# Handling all exceptions using a single except block
try:
    a = int(input("Enter a number: "))
    b = int(input("Enter another number: "))
    result = a / b
    print("Result:", result)

    lst = [1, 2, 3]
    index = int(input("Enter index to access: "))
    print("Element:", lst[index])
```

```
except (ZeroDivisionError, ValueError, TypeError) as e:
    print(f"Error occurred: {e}")
```

```
Enter a number:  10
Enter another number:  0

Error occurred: division by zero
```

02) WAP to handle following exceptions:

1. IndexError
2. KeyError

```
try:
    lst = [1, 2, 3]
    print(lst[5])

except IndexError:
    print("Error: List index out of range.")

try:
    d = {"a": 1, "b": 2}
    print(d["a"])

except KeyError:
    print("Error: Key not found in dictionary.")
```

```
Error: List index out of range.
1
```

03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

```
try:
    with open("demo.txt", "r") as file:
        content = file.read()
        print("File Content is:",content)

except FileNotFoundError:
    print("Error: File not found.")

try:
    import non_existent_module

except ModuleNotFoundError:
    print("Error: Module not found.")
```

```
File Content is: Dadhaniya Parth
Student of Darshan University
```

```
Full Stack Developer
Error: Module not found.
```

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

```
try:
    a = int(input("Enter a number: "))
    b = int(input("Enter another number: "))
    result = a / b
    print("Ans=",result)

    lst = [1, 2, 3]
    print(lst[1])

    d = {"a": 1, "b": 2}
    print(d["a"])

    import non_existent_module

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

```
Enter a number:  10
Enter another number:  2

Ans= 5.0
2
1
An error occurred: No module named 'non_existent_module'
```

05) WAP to demonstrate else and finally block.

```
try:
    a = int(input("Enter a number: "))
    b = int(input("Enter another number: "))
    result = a / b
except ZeroDivisionError:
    print("Error: Division by zero is not allowed.")
except ValueError:
    print("Error: Invalid input! Please enter an integer.")
else:
    print("Division successful! Result:", result)
finally:
    print("Execution completed. This will always run.")
```

```
Enter a number:  10.5

Error: Invalid input! Please enter an integer.
Execution completed. This will always run.
```

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.

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

```
Enter a list of grades separated by commas:  89,A,79

Error: Please enter only numeric values separated by commas.
```

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

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

num1 = int(input("Enter numerator: "))
num2 = int(input("Enter denominator: "))

print("Result:", divide(num1, num2))
```

```
Enter numerator:  10
Enter denominator:  0

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

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.

```
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)
```

```
Enter your age:  12
```

```
Error: Enter Valid Age
```

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.

```
class InvalidUsernameError(Exception):
    pass

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("Username:", username)
except InvalidUsernameError as e:
    print("Error:", e)
```

```
Enter your username:  tree
```

```
Error: Username must be between 5 and 15 characters long
```

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.

```
import math

class NegativeNumberError(Exception):
    pass

try:
    num = float(input("Enter a number: "))
    if num < 0:
        raise NegativeNumberError("Cannot calculate the square root of
a negative number")
    print("Square root:", math.sqrt(num))
except NegativeNumberError as e:
    print("Error:", e)
```

```
Enter a number:  -12

Error: Cannot calculate the square root of a negative number
```