Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 10

Jeet Bhalodi (23031701006)
11-02-2025

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 [47]:
```
try:
    num = int(input("Enter a number: "))
    result = 10 / num
    print(result)
except ZeroDivisionError:
    print("Error: Division by zero is not allowed.")
except ValueError:
    print("Error: Invalid input, please enter a number.")
except TypeError:
    print("Error: Type error encountered.")


# try:
#     num = int(input("Enter a number: "))
#     result = 10 / num
#     print(result)
```

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

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

02) WAP to handle following exceptions:

1. IndexError
2. KeyError

In [21]:
```
try:
    my_list = [1, 2, 3]
    print(my_list[5])

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

try:
    dct = {'a': 10}
    print(dct['b'])
except KeyError:
    print("Error: Key not found in dictionary.")
```

```
Error: Index out of range.
Error: Key not found in dictionary.
```

03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

In [31]:
```
try:
    fp = open('jeet.txt', 'r')
    content = fp.read()

except FileNotFoundError:
    print("Error: The file was not found.")

try:
    import my_jeet_module

except ModuleNotFoundError:
    print("Error: The module was not found.")
```

```
Error: The file was not found.
Error: The module was not found.
```

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

In [33]:
```
try:
    num = int(input("Enter a number: "))
    result = 10 / num
```

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


try:
    my_list = [1, 2, 3]
    print(my_list[5])
except Exception as err:
    print(f"An error occurred: {err}")

try:
    dct = {'a': 10}
    print(dct['b'])
except Exception as err:
    print(f"An error occurred: {err}")

try:
    fp = open('jeet.txt', 'r')
    content = fp.read()
except Exception as err:
    print(f"An error occurred: {err}")

try:
    import my_jeet_module
except Exception as err:
    print(f"An error occurred: {err}")
```

```
An error occurred: division by zero
An error occurred: list index out of range
An error occurred: 'b'
An error occurred: [Errno 2] No such file or directory: 'jeet.txt'
An error occurred: No module named 'my_jeet_module'
```

05) WAP to demonstrate else and finally block.

In [39]:
```
try:
    num = int(input("Enter a number: "))
    result = 10 / num
    print(f"Result: {result}")

except ZeroDivisionError:
    print("Error: Division by zero is not allowed.")
except ValueError:
    print("Error: Invalid input, please enter a number.")
else:
    print("The try block executed successfully without any exceptions.")
finally:
    print("This will always execute.")
```

```
Result: 5.0
The try block executed successfully without any exceptions.
This will always execute.
```

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 [43]:
```
try:
    grades_input = input("Enter a list of grades separated by commas: ")

    grades = [int(grade.strip()) for grade in grades_input.split(",")]

    print(f"The converted list of grades is: {grades}")

except Exception as err:
    print("Error: ",err)
```

```
Error:  invalid literal for int() with base 10: 'a'
```

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

In [76]:
```
def divide(a, b):
    try:
        result = a / b
        return result
    except ZeroDivisionError:
        return "Error: Division by zero is not allowed."

a = int(input("Enter the numerator: "))
b = int(input("Enter the denominator: "))

output = divide(a, b)
print(f"The result is: {output}")
```

```
The result is: 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.

In [65]:
```
try:
    age = int(input("Enter your age: "))
    if age < 18:
        raise ValueError("Enter Valid Age")
    else:
```

```
        print(f"Your age is: {age}")
except ValueError as err:
    print(f"Error: {err}")
```

```
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.

In [59]:
```
class InvalidUsernameError(Exception):
    def __init__(self, message):
        self.message = message

try:
    username = input("Enter your username: ")
    if len(username) < 5 or len(username) > 15:
        raise InvalidUsernameError("Username must be between 5 and 15 characters lo
    else:
        print(f"Your username is: {username}")

except InvalidUsernameError as e:
    print(f"Error: {e}")
```

```
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.

In [71]:
```
import math

class NegativeNumberError(Exception):
    def __init__(self, message):
        self.message = message

try:
    number = float(input("Enter a number: "))
    if number < 0:
        raise NegativeNumberError("Cannot calculate the square root of a negative n
    else:
        sqrt = math.sqrt(number)
        print(f"The square root of {number} is: {sqrt}")

except NegativeNumberError as e:
    print(f"Error: {e}")
```

```
except ValueError:
    print("Error: Invalid input, please enter a valid number.")
```

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

In [ ]: