Darshan UNIVERSITY
योग: कर्मसु कौशलम्

Python Programming - 2301CS404

Lab - 10

Dhol Namra

Enroll:23010101407

29-01-2025

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 [34]:
```
try:

    n1 = int(input("Enter the numerator: "))
    n2 = int(input("Enter the denominator: "))
    print(n1/n2)


    result_add = n1 + "5"
    print(f"Result of addition: {result_add}")

except ZeroDivisionError:
    print("ZeroDivisionError")
except ValueError:
    print("ValueError")
except TypeError:
    print("TypeError")


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

```
2.0
TypeError
```

02) WAP to handle following exceptions:

1. IndexError
2. KeyError

In [51]:
```
try:
    list = [1,2,3,3,4,5,9]
    print(type(list))
    print(list[1])
    d1 = {1:"asdf",2:"asdfg"}
    print(d1[3])
# except IndexError:
#     print("IndexError")
# except KeyError:
#     print("KeyError")
except (IndexError, KeyError) as e:
        print(f"An error occurred: {e}")
```

```
<class 'list'>
2
An error occurred: 3
```

03) WAP to handle following exceptions:

1. FileNotFoundError
2. ModuleNotFoundError

In [60]:
```
try:
    # fp = open("abc.txt","r")
    # print(fp.read())
    # fp.close()
    import asdfghjkl
except FileNotFoundError:
    print("fileNotFound")
except ModuleNotFoundError:
    print("module not found")
```

```
module not found
```

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

In [67]:
```
try:
    fp = open("abc.txt","r")
    print(fp.read())
    fp.close()
```

```
except Exception as err:
    print(err)
```

```
[Errno 2] No such file or directory: 'abc.txt'
```

05) WAP to demonstrate else and finally block.

In [87]:
```
try:
    fp = open("abc.txt","r")
    print(fp.read())
    fp.close()
except Exception as err:
    print(err)
else:
    print("else block excuted")
finally:
    print("finally block excuated")
```

```
else block excuted
finally block excuated
```

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 [107…
```
    grade_input = input("Enter a list of grades separated by commas: ")

    try:
        grade_strings = grade_input.split(",")

        grades = [int(grade.strip()) for grade in grade_strings]

        print("Grades entered:", grades)

    except ValueError:
        print("ValueError")
```

```
ValueError
```

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

In [99]:
```
def Divide(a,b):
    try:
        print(a/b)
    except Exception as error:
        print("Error accurd :",error)

Divide(5,0)
```

```
Error accurd : division by zero
```

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 [142…

```
try:
    age = int(input("Enter a list of grades separated by commas: "))
    if(age > 18):
        print(age)
    else:
        raise ValueError
except ValueError:
               print("ValueError")
```

```
ValueError
```

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 [182…

```
try:
    s1 = input("enter a user name :;")
    if(len(s1) >= 5  and len(s1) <=15):
        print(s1)
    else:
        raise ValueError("abcd")
except ValueError as err:
    print(err)
```

```
abcd
```

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 [172…

```
import math

class NegativeNumberError(Exception):
    def __init__(self, message="Cannot calculate the square root of a negative n
        self.message = message
        super().__init__(self.message)

def calculate_square_root(number):
    if number < 0:
        raise NegativeNumberError()
    print(f"Square root: {math.sqrt(number)}")

try:
    number = float(input("Enter a number: "))
    calculate_square_root(number)
except NegativeNumberError as e:
    print(f"Error: {e}")
```

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

In [ ]: