Exception:

-🡪 A Python program terminates as soon as it encounters an error

print ("Welcome to exception concept" 🡪 Syntax Error: unexpected EOF while parsing

print(4/0) 🡪 ZeroDivisionError: division by zero

Errors are two types of errors:

1. Syntax Errors 🡪 No way to handle syntax errors
2. Run time errors 🡪 Exceptions, There is a way to handle runtime errors

A computer screen shot of a code

AI-generated content may be incorrect.

try:

    import fabric

except Exception as e:

    print(e.\_\_class\_\_)

# <class 'ModuleNotFoundError'>

try:

    import fabric

except ModuleNotFoundError:

    print("Handled Module not found error")

$ python exception2.py

Handled Module not found error

A screenshot of a computer

AI-generated content may be incorrect.

a=int(input("Enter the number 1:"))

b=int(input("Enter the number 2:"))

lst=[2,4,6]

n=100

try:

    div=a/b

    print("Div:",div)

    print("lst 3rd Index:",lst[2])

    print(n)

except ZeroDivisionError:

    print("Zero division error occured") #  8/0. Zero division error occured

except IndexError:

    print("Index error occured")

except NameError:

    print("Name error occured")

except Exception as e:

    print("Exception occured:",e.\_\_class\_\_)

else:

    print("No exception occured") # No exception occured

finally:

    print("This is always executed ")  # This is always executed . for exception also this is will executed

CLASS 2:

Exception Handling for Known Exceptions

try:

    #print(a)

    # print(4+"hi")

    # open("1.txt","r")

    # a=eval(input("Enter the number a:"))

    # b=eval(input("Enter the number b:"))

    # a/b

    # import fabric

    11

    import math

except NameError:

    print("variable is not defined \n")

except TypeError:

    print("Adding number and string is not possible \n")

except FileNotFoundError:

    print("File not found error \n")

except ZeroDivisionError:

    print("Division with zero not possible")

except ModuleNotFoundError:

    print("Module not imported")

except Exception as e:

    print(e.\_\_class\_\_)

else:

    print("No exception")

finally:

    print("its always executed")

Output:

variable is not defined

its always executed

Class 3:

try:

    a=10

    print(a)

except NameError:

    print("variables is not defined")

except Exception as e:

    print("Exception occured",e)

else:

    print("This will excute if there is no exception")

finally:

    print("This will execute always")

Raise user defined Exceptions:

Custom Exceptions

* Raise and Assert
* Raise: Used to raise an existing exception
* Assert: Used to create an Assertion Error

raise Exception(“This is Exception”)

Error:

raise Exception("This is Exception")

Exception: This is Exception

age=23

if age>30:

    print("valid age::")

else:

    raise ValueError("Age is less than 30 ")

File "raise\_exception.py", line 6, in <module>

raise ValueError("Age is less than 30 ")

ValueError: Age is less than 30

2.

assert 3>4

File "raise\_exception.py", line 8, in <module>

assert 3>4

AssertionError

age=33

3.

try:

    assert age<30

    print("Valid age")

except:

    print("Exception occured")

4.

try:

    assert age<30

    print("Valid age")

except AssertionError:

    print("Raised with assert because age is less than 30")

except:

    print("Exception occured")

$ python raise\_exception.py

Raised with assert because age is less than 30