**EXCEPTIN HANDLING**

Exceptions are runtime errors.

When the python program is executing, and something goes wrong, an exception is raised and if we don’t handle the exception, it will cost the below:

1. Abnormal Termination of a program
2. Will display Informal / Un Friendly information to the end user
3. If we have resources like db connection / file streams or network connection, the we have improper shutdown of the resources.

In Python, we can exception is represented using a class. There are different type of inbuild exceptions that we can use in our application.

Also we can define our own exception types and use them for our business needs.

To handle the exception, the code will be inside the try and except block;

try:

code block

except Zero Division Error:

<if the code in try raises an exception and the execution will go to the except block.>

else:

Will execute, if an exception is not raised.

finally:

If we have an except or else block and have finally block, this will be executed.

All the cleanup code can be written in finally block.

A diagram of a error

Description automatically generated

**Code:**

#ENTER TWO NUMBER FROM THE ENDUSER USING RECURSSION

v1,v2 = [int(x) for x in input(*"Enter 2 numbers: "*).split(*","*)]

v3=v1/v2

print(v3)

**Output:**

If the value for v1 > 0 and v2 > 0

Enter 2 numbers: 5,3

1.6666666666666667

If the value of v2 ==0 or both the values v1 and v2 == 0

Enter 2 numbers: 5,0

Traceback (most recent call last):

File "C:\Users\kamal\OneDrive\Desktop\Python\Python Examples\o\_exceptionHandling\demo.py", line 5, in <module>

v3=v1/v2

~~^~~

ZeroDivisionError: division by zero

If the value of v1 == 0

Enter 2 numbers: 0,5

0.0

**Code with Try and Except**

try:

#ENTER TWO NUMBER FROM THE ENDUSER USING RECURSSION

v1,v2 = [int(x) for x in input(*"Enter 2 numbers: "*).split(*","*)]

v3=v1/v2

print(v3)

except ZeroDivisionError:

print(*"The value entered for value v2 is Zero, which is not allowed"*)

print(*"Temporarily assgined the value of v2 = 1"*)

v2=1

print(v1/v2)

**Output:**

Enter 2 numbers: 4,0

The value entered for value v2 is Zero, which is not allowed

Temporarily assgined the value of v2 = 1

4.0

If the except block is not added, then these block of codes will not get executed.

print(*"Temporarily assgined the value of v2 = 1"*)

v2=1

print(v1/v2)

**Code with Try , Except and Finally**

try:

fl=open(*"fileEgs1"*,*"w"*)

#ENTER TWO NUMBER FROM THE ENDUSER USING RECURSSION

v1,v2 = [int(x) for x in input(*"Enter 2 numbers: "*).split(*","*)]

v3=v1/v2

fl.write(*"Writing %d in the file "*%v2)

except ZeroDivisionError:

print(*"The value entered for value v2 is Zero, which is not allowed"*)

finally:

fl.close()

print(*"file closed"*)

**Output:**

Enter 2 numbers: 5,2

file closed

**Note:**

The above file is created as below:

A screenshot of a computer

Description automatically generated

**Code with Try , Except , Else and Finally**

try:

fl=open(*"fileEgs1"*,*"w"*)

#ENTER TWO NUMBER FROM THE ENDUSER USING RECURSSION

v1,v2 = [int(x) for x in input(*"Enter 2 numbers: "*).split(*","*)]

v3=v1/v2

fl.write(*"Writing %d in the file "*%v2)

except ZeroDivisionError:

print(*"The value entered for value v2 is Zero, which is not allowed"*)

s=*"The Division value is Zeor.. Hence Error"*

fl.write(s)

else:

print(*"Entered a non zero number...."*)

finally:

fl.close()

print(*"file closed"*)

**Output:**

Enter 2 numbers: 4,0

The value entered for value v2 is Zero, which is not allowed

file closed

A screenshot of a computer

Description automatically generated

**CUSTOM EXCEPTIONS**

**Code:**

class **OverLimitException**(Exception):

def **\_\_init\_\_**(*self*,msg):

*self*.overLimitMsg=msg

def **withDrawing**(amount):

if(amount>500):

msg = *"You cannot withdraw more than 500£ on single transaction !!!!!"*

raise OverLimitException(msg)

else:

print(*"The amount is of %d withdrawn from your account...."* %amount)

a1,a2 = [int(x) for x in input(*"Enter 2 amounts: "*).split(*","*)]

print()

withDrawing(a1)

print()

withDrawing(a2)

**Output:**

Enter 2 amounts: 250,700

The amount is of 250 withdrawn from your account....

Traceback (most recent call last):

File "C:\Users\kamal\OneDrive\Desktop\Python\Python Examples\o\_exceptionHandling\customException.py", line 17, in <module>

withDrawing(a2)

File "C:\Users\kamal\OneDrive\Desktop\Python\Python Examples\o\_exceptionHandling\customException.py", line 8, in withDrawing

raise OverLimitException(msg)

OverLimitException: You cannot withdraw more than 500£ on single transaction !!!!!

**Code with Multiple Exception:**

class **YoungException**(Exception):

def **\_\_init\_\_**(*self*,msg):

*self*.youngLimitMsg=msg

class **OldException**(Exception):

def **\_\_init\_\_**(*self*,msg):

*self*.oldLimitMsg=msg

age=int(input(*"Enter the age for licence : "*))

if age < 18:

raise YoungException(*"Entered age is less 18 for licence!!!"*)

elif age > 75:

raise OldException(*"Entered age is less 18 for licence!!!"*)

else:

print(*"Age between between 18 and 75.. Can apply for the licence...."*)

print()

**Output:**

If age is >75

Enter the age for licence : 79

Traceback (most recent call last):

File "C:\Users\kamal\OneDrive\Desktop\Python\Python Examples\o\_exceptionHandling\licenceEligbigible.py", line 16, in <module>

raise OldException("Entered age is less 18 for licence!!!")

OldException: Entered age is less 18 for licence!!!

If age is >18

Enter the age for licence : 25

Age between between 18 and 75.. Can apply for the licence....

If age is <18

Enter the age for licence : 16

Traceback (most recent call last):

File "C:\Users\kamal\OneDrive\Desktop\Python\Python Examples\o\_exceptionHandling\licenceEligbigible.py", line 13, in <module>

raise YoungException("Entered age is less 18 for licence!!!")

YoungException: Entered age is less 18 for licence!!!

**LOGGING**

The hierarchy of the logging is as below

It will show only the first there and the last two will be ignore, if not updated in the file.

**Code:**

import logging

logging.critical(*"Critical"*)

logging.error(*"Error"*)

logging.warn(*"Waring"*)

logging.info(*"Information"*)

logging.debug(*"Debug"*)

**Output:**

CRITICAL:root:Critical

ERROR:root:Error

C:\Users\kamal\OneDrive\Desktop\Python\Python Examples\o\_exceptionHandling\logingDemo.py:5: DeprecationWarning: The 'warn' function is deprecated, use 'warning' instead

logging.warn("Waring")

WARNING:root:Waring

**Code to update in the log file:**

import logging

#THIS WILL CREATE A LOG IN THE SAME PROJECT FOLDER AND SAVE THE LOGS WHICH ARE ABOVE THE DEBUG OPTION

logging.basicConfig(filename=*"sampleLog1.log"*,level=logging.DEBUG)

logging.critical(*"Critical"*)

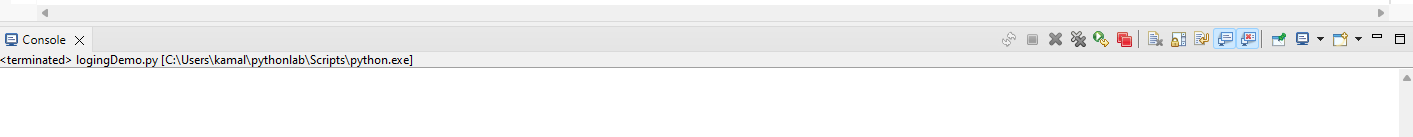
logging.error(*"Error"*)

logging.warning(*"Waring"*)

logging.info(*"Information"*)

logging.debug(*"Debug"*)

**Output in console:**



**File I the folder:**

A computer screen shot of a computer code

Description automatically generated

**Output in fodler:**

CRITICAL:root:Critical

ERROR:root:Error

WARNING:root:Waring

INFO:root:Information

DEBUG:root:Debug

IN THE CODE IF WE CHANGE TO

#THIS WILL CREATE A LOG IN THE SAME PROJECT FOLDER AND SAVE THE LOGS WHICH ARE ABOVE THE DEBUG OPTION

logging.basicConfig(filename=*"sampleLog1.log"*,level=logging.CRITICAL)

THE LOG FILE WILL ONLY HAVE

CRITICAL:root:Critical

**EXCEPTION HANDLING USING LOGGING**

**Code:**

import logging

logging.basicConfig(filename=*"sampleLog1.log"*,level=logging.DEBUG)

try:

fl=open(*"fileEgs1"*,*"w"*)

#ENTER TWO NUMBER FROM THE ENDUSER USING RECURSSION

v1,v2 = [int(x) for x in input(*"Enter 2 numbers: "*).split(*","*)]

logging.info(*"Division in Progress"*)

v3=v1/v2

fl.write(*"Writing %d in the file "*%v2)

except ZeroDivisionError:

print(*"The value entered for value v2 is Zero, which is not allowed"*)

s=*"The Division value is Zeor.. Hence Error"*

fl.write(s)

logging.error(*"Division by Zero Error"*)

else:

print(*"Entered a non zero number...."*)

finally:

fl.close()

print(*"file closed"*)

**Output in Console:**

Enter 2 numbers: 25,5

Entered a non zero number....

file closed

Enter 2 numbers: 25,0

The value entered for value v2 is Zero, which is not allowed

file closed

**Output in Log File:**

INFO:root:Division in Progress

INFO:root:Division in Progress

ERROR:root:Division by Zero Error

**ASSERT STATEMENT**

**Code:**

xyz = int(input(*"Enter the even number for xyz : "*))

assert xyz%2 == 0, *"----------You have entered an Odd Number-------"*

print(*"Code after Assertion"*)

**Output in Console:**

Enter the even number for xyz : 5

Traceback (most recent call last):

File "C:\Users\kamal\OneDrive\Desktop\Python\Python Examples\o\_exceptionHandling\assertDemo.py", line 3, in <module>

assert xyz%2 ==0, "----------You have entered an Odd Number-------"

AssertionError: ----------You have entered an Odd Number-------

**The code after the assert statement will not execute, if the condition is false. To overcome this, use in try block as below:**

**Code:**

try:

xyz = int(input(*"Enter the even number for xyz : "*))

assert xyz%2 == 0, *"----------You have entered an Odd Number-------"*

except AssertionError as obj:

print(obj)

print(*"Code after Assertion"*)

**Output in Console:**

Enter the even number for xyz : 5

----------You have entered an Odd Number-------

Code after Assertion

**QUIZ**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

A screenshot of a computer

Description automatically generated

A screenshot of a computer error

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

ASSIGNMENT

**Code 1**

class **InvalidLengthException**(Exception):

def **\_\_init\_\_**(*self*,msg):

*self*.invalidExcep=msg

def **passwordChecking**():

try:

pwd = input(*"Please enter a password : "*)

if (len(pwd) < 8):

raise InvalidLengthException(*"Password Invalid - The length is less than 8 Chars !!!!"*)

except InvalidLengthException as obj:

print(obj)

passwordChecking()

**Code 2**

import logging

logging.basicConfig(filename="logging.log",level=logging.ERROR)

class InvalidPasswordException(Exception):

    def \_\_init\_\_(self,msg):

        self.msg = msg

password=input("Please Enter a password:")

try:

    if len(password) >= 8:

        print("Password Is Valid")

    else:

        raise InvalidPasswordException("")

except InvalidPasswordException:

    print("Password Is Invalid!!!")

    logging.error("Password Is Invalid!!!")

print("Program Continues")

**Code 3**

class InvalidPasswordException(Exception):

def \_\_init\_\_(self):

pass

def check\_password():

p = input("Enter Password: ")

try:

if len(p) < 8:

raise InvalidPasswordException

print("Login Successful")

except InvalidPasswordException:

print("Enter atleast 8 numbers in password")

print("After Exception")

check\_password()