Exception Handling

------------------

In any programming language there 2 types of errors are possible.

1) Syntax Errors

2) Runtime Errors

1)Syntax Errors

---------------

The errors which occurs because of invalid syntax are

called syntax errors.

ex 1)

x = 10

if x==10

print("Hello")

SyntaxError:invalid syntax

ex 2) print "Hello"

SyntaxError:Missing parentheses in call to print

Note:- Programmer is responsible to correct these syntax errors.once all syntax errors are corrected then only

program execution will be started.

2) Runtime Errors:

------------------

Also known as exceptions.

While executing the program if something goes wrong because

of end user input or programming logic or memory problems etc

then we will get Runtime Errors.

ex 3) print(10/0) ==>ZeroDivisionError:divison by zero

print(10/"ten") ==>TypeError:unsupported operand

types for /: 'int' and 'str'

Note:-Exception handling concept applicable for runtime errors but not syntax errors.

What is Exception?

An unwanted and unexpected event that distrubs normal flow

of program is called exception.

ex:- ZeroDivisionError

TypeError

ValueError

FileNotFoundError

EOFError

SleepingError

its highly recommended to handle exceptions.the main objective of exception handling is graceful termination of the program(i.e we should not block our resources and we

sholud not miss anything)

Exception handling does not mean repairing exception.

We have to define alternative way to continue rest of the

program normally.

Default Exception Handling in Python:

------------------------------------

Every exception in python is an object.For every exception type the corresponding classes are available.

Whenever an exception occurs Python Virtul Machine(PVM) will

create the corresponding exception object and will check for

handling code.If handling code is not available the python

interpreter terminates the program abnormally and prints

corresponding exception information to the console.

The rest of program won't be executed.

ex 4) Print("Hello")

print(10/0)

print("Hi")

Python Exception Hierarchy

--------------------------

Every exception in python is a class.

All exception class are child classes of BaseException.i.e every exception class extends BaseException either directly

or indirectly.Hence BaseException acts as root for python

exception Hierarchy.

Most of times being programmer we have to concentrate exception and its child classes.

Customized Exception Handling by using try-except

-------------------------------------------------

It is highly recommended to handle exceptions.

The code which may raise exception is called risky code and

we have to take risky code inside try block.

The corresponding handling code we have to take inside

except block.

Syntax:-

try:

Risky Code

except <name of exception class>:

Handling code/Alternative code

Without try-except:

------------------

ex 5)

print("Hello")

print(10/0)

print("Hi")

Abnormal termination/Non Graceful Termination.

With try-except:

----------------

ex 6)

print("Hello")

try:

print(10/0)

except ZeroDivisionError:

print(10/2)

print("Hi")

Normal termination/graceful termination

Control Flow in try-except:

---------------------------

syntax:

try:

stmt-1

stmt-2

stmt-3

except <exception class>:

stmt-4

stmt-5

Case-1)If there is no exception

1,2,3,5 and Normal Termination.

Case-2)If an exception raised at stmt-2 and corresponding except block matched 1,4,5 Normal Termination.

Case-3)If an exception raised at stmt-2 and corresponding

except block not matched 1,Abnormal Termination

Case-4)If an exception raised at stmt-4 or stmt-5 then its

always abnormal termination.

Conclusions:

-----------

1) within the try block if anywhere exception raised then rest of the try block wont be executed eventhough we handled

that exception.hence we have to take only risky code inside the try block and length of the try block sholud be as less

as possible.

2) In addition to try block,there may be a chance of raising

exceptions inside except and finally blocks also.

3)If any statement which is not part of try block raises an exception then its always abnormal termination.

How to print exception information

----------------------------------

ex 7)

try:

print(10/0)

except ZeroDivisionError as msg:

print("exception msg is:",msg)

try with multiple except blocks

-------------------------------

The way of handling exception is varied from exception to exception.hence for every exception type a sperate except

block we have to provide.i.e try with multiple except blocks

is possible and recommended to use.

syntax:

try:

stmts

except <exception class>:

perform alernative

except <exception class>:

perform alternative

ex 8)

try:

x = int(input("Enter First Number:"))

y = int(input("Enter Second Number:"))

print(x/y)

except ZeroDivisionError:

print("Can't divide with zero")

except ValueError:

print("Please provide int value only")

Note:- If try with multiple except blocks availabe then the

order of these except blocks is important.Python interpreter

will always consider from top to bottom untill matched except

block indentified.

Single except block that can handle multiple exceptions

-------------------------------------------------------

we can write a single except block that can handle multiple

different types of exceptions.

syntax:

except(Exception1,Exception2,Exception3,...) or

except(Exception1,Exception2,Exception3,...) as msg:

() are mandatory and this group of exceptions internally

consider as tuple.

ex 9)

try:

x = int(input("Enter First Number:"))

y = int(input("Enter Second Number:"))

print(x/y)

except (ZeroDivisionError,ValueError) as msg:

print("Please provide valid numbers only and problem is:",msg)

Default except block:

--------------------

We can use default except block to handle any type of exceptions.

In default except block generally we can print normal error

messages.

syntax:

except:

statemets

ex 10)

try:

x = int(input("Enter First Number:"))

y = int(input("Enter Second Number:"))

print(x/y)

except ZeroDivisionError:

print("Can't divide with zero")

except:

print("Default Except:Please provide int value only")

Note:-if try with multiple except blocks available then default except blocks should be last,otherwise we will get

SyntaxError.

ex 11)

try:

print(10/0)

except:

print("Default Except")

except ZeroDivisionError:

print("ZeroDivisionError")

Note:

The following are various possible combinations of except

blocks.

1) except ZeroDivisionError:

2) except ZeroDivisionError as msg:

3) except(ZeroDivisionError,ValueError):

4) except(ZeroDivisionError,ValueError) as msg:

5) except:

finally block:

--------------

1)It is not recommended to maintain clean up code(Resource Deallocating code or Resource Releasing code)inside try block

because there is no guarentee for the execution of every statement inside try block always.

2)It is not recommended to maintain clean up code inside excpet block,because if there is no exception then except

block won't be executed.

Hence we required some place to maintain clean up code which

should be executed always irrespective of whether exception raised or not raised and wether exception handled or not handled.such type of best place is nothing but finally block.

Hence the main purpose of finally block is to maintain clean up code.

syntax:

try:

Risky Code

except:

Handling Code

finally:

Cleanup Code

The speciality of finally block is it will be executed always

whether exception raised or not raised and wether exception

handler or not handled.

ex 12)

try:

print("try")

except:

print("except")

finally:

print("finally")

ex 13)

try:

print("try")

print(10/0)

except ZeroDivisionError:

print("except")

finally:

print("finally")

ex 14)

try:

print("try")

print(10/0)

except NameError:

print("except")

finally:

print("finally")

Note:-There is only one situation where finally block won't

be executed i.e whenever we are using os.\_exit(0) function.

whenever we are using os.\_exit(0) function then PVM itself

will be shoutdown.in this case finally won't be executed.

ex 15)

import os

try:

print("try")

os.\_exit(0)

except NameError:

print("except")

finally:

print("finally")

Note:os.\_exit(0)

where 0 represents status code and it indicates normal termination.

Control flow in try-except-finally

----------------------------------

syntax:

try:

stmt-1

stmt-2

stmt-3

except:

stmt-4

finally:

stmt-5

stmt-6

Case-1:if there is no exception

1,2,3,5,6 Normal Termination

Case-2:If an exception raised at stmt-2 and the corresponding

except block matched.

1,4,5,6 Normal Termination

Case-3:If an exception raised at stmt-2 and the corresponding

except block not matched.

1,5 Abnormal Termination

Case-4:If an exception raised at stmt-4 then it always abnormal termination but before finally block will be executed.

Case-5:If an exception raised at stmt-5 or stmt-6 then it's always abnormal termination.

Nested try-except-finally blocks:

---------------------------------

We can take try-except-finally blocks inside try or except or

finally blocks.i.e nesting of try-except-finally is possible.

syntax:

try:

statements

try:

statements

except:

statements

except:

statements

Generally risky code we have to take inside outer try block

and too much risky code we have to take inside inner try block.inside try block if an exception raised then inner

except block is responsible to handle.if it is unable to handle then outer except block is responsible to handle.

ex 16)

try:

print("outer try block")

try:

print("inner try block")

print(10/0)

except ZeroDivisionError:

print("inner except block")

finally:

print("inner finally block")

except:

print("Outer except block")

finally:

print("Outer finally block")

control flow in nested try-except-finally:

-----------------------------------------

try:

stmt-1

stmt-2

stmt-3

try:

stmt-4

stmt-5

stmt-6

except X:

stmt-7

finally:

stmt-8

stmt-9

except Y:

stmt-10

finally:

stmt-11

stmt-12

Case-1:If there is no exceptions

1,2,3,4,5,6,8,9,11,12 Normal Termination

Case-2:If an exception raised at stmt-2 the corresponding

except block matched 1,10,11,12 Normal termination.

Case-3:If an exception raised at stmt-2 the corresponding

except block not matched 1,11 Abnormal termination.

Case-4:If an exception raised at stmt-5 and inner except block matched

1,2,3,4,7,8,9,11,12 Normal Termination

Case-5:If an exception raised at stmt-5 and inner except block not matched

1,2,3,4,8,10,11,12 Normal Termination

Case-6:If an exception raised stmt-5 and both inner and outer

except blocks are not matched

1,2,3,4,8,11 Abnormal Termination.

Case-7:If an exception raised at stmt-7 and corresponding except block matched 1,2,3,,,,8,10,11,12 Normal Termination.

Case-8:If an exception raised at stmt-7 and corresponding except block not matched 1,2,3,,,,8,11 Abnormal Termination.

Case-9:If an exception riased at stmt-8 and corresponding except block matched 1,2,3,,,,10,11,12 normal termination.

Case-10:If an exception riased at stmt-8 and corresponding except block not matched 1,2,3,,,,11 Abnormal termination.

Case-11:If an exception raised at stmt-9 and corresponding except block matched 1,2,3,,,,8,10,11,12 Normal termination.

Case-12:If an exception raised at stmt-9 and corresponding except block not matched 1,2,3,,,,8,11 AbNormal termination.

Case-13:If an exception raised at stmt-10 then it is always AbNormal termination finally block(stmt-11) will be executed.

Case-14:If an exception raised at stmt-11 and stmt-12 then it is always AbNormal termination.

Note:-If the control entered into try block then compulsary

finally block will be executed.

if the control not entered into try block then finally block

won't be executed.

else block with try-except-finally:

----------------------------------