**Exceptions**

Python provides two very important features to handle any unexpected error and to add debugging capabilities in them.

1. Exception Handling
2. Assertions

# Assertions

An assertion is a sanity-check that you can turn on or turn off when you are done with your testing of the program.

1. An expression is tested, and if the result comes up false, an exception is raised.
2. Assertions are carried out by the assert statement

# Exceptions

Python uses special objects called exceptions to manage errors that arise during a program’s execution. If you don’t handle the exception, the program will halt and show a traceback, which includes a report of the exception that was raised.

# Handling an exception

## Using try-except Blocks

you can write a try-except block to handle the exception that might be raised.

**Syntax**

try:

You do your operations here

......................

except ExceptionI:

If there is ExceptionI, then execute this block.

except ExceptionII:

If there is ExceptionII, then execute this block.

......................

else:

If there is no exception then execute this block.

**Important points:**

1. A single try statement can have multiple except statements.
2. You can also provide a generic except clause, which handles any exception.
3. After the except clause(s), you can include an else-clause. The code in the else-block executes if the code in the try: block does not raise an exception.
4. The else-block is a good place for code that does not need the try: block's protection.

## except Clause with No Exceptions

**Syntax**

try:

You do your operations here

......................

except:

If there is any exception, then execute this block.

......................

else:

If there is no exception then execute this block.

## except Clause with Multiple Exceptions

**Syntax**

try:

You do your operations here

......................

except(Exception1[, Exception2[,...ExceptionN]]]):

If there is any exception from the given exception list,

then execute this block.

......................

else:

If there is no exception then execute this block.

## try-finally Clause

**finally:** block is a place to put any code that must execute, whether the try-block raised an exception or not.

**Syntax**

try:

You do your operations here;

......................

Due to any exception, this may be skipped.

finally:

This would always be executed.

......................

**Note:** You can provide except clause(s), or a finally clause, but not both. You cannot use else clause as well along with a finally clause.

#!/usr/bin/python3

try:

fh = open("testfile.txt", "w")

fh.write("This is my test file for exception handling!!")

except IOError:

print("Error: can\'t find file or read data")

else:

print("Written content in the file successfully")

finally:

print('Inside finally block')

Output:

Written content in the file successfully

Inside finally block

#!/usr/bin/python3

try:

fh = open("testfile.txt", "r")

fh.write("This is my test file for exception handling!!")

except IOError:

print("Error: can\'t find file or read data")

else:

print("Written content in the file successfully")

finally:

print('Inside finally block')

Output:

Error: can't find file or read data

Inside finally block

else after finally results in syntax error

#!/usr/bin/python3

try:

fh = open("testfile.txt", "r")

fh.write("This is my test file for exception handling!!")

except IOError:

print("Error: can\'t find file or read data")

finally:

print('Inside finally block')

else:

print("Written content in the file successfully")

Output:

File "learn\_exception.py", line 10

else:

^

SyntaxError: invalid syntax

# Argument of an Exception

argument gives additional information about the problem. You capture an exception's argument by supplying a variable in the except clause as follows:

try:

You do your operations here

......................

except ExceptionType as Argument:

You can print value of Argument here...

* to handle a single exception, you can have a variable follow the name of the exception in the except statement.
* to handle multiple exceptions, you can have a variable follow the tuple of the exception.

# Raising an Exception

General syntax for the raise statement is:

**Syntax**

raise [Exception [, args [, traceback]]]

* Exception is the type of exception
* argument is a value for the exception argument (default None)

# User-Defined Exceptions

Python also allows you to create your own exceptions by deriving classes from the standard built-in exceptions.

Example:

class Networkerror(RuntimeError):

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

self.args = arg

try:

raise Networkerror("Bad hostname")

except Networkerror,e:

print e.args

# END