**Exception Handling**

**Error:**

* Something goes wrong in program.
* Occurs at compile time

**Exception:**

* Exceptions is an event which occurs during execution of program and disrupt the normal flow of program’s instructions

**How to handle Exception:**

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.

**Example:**

try:

fob = open("test", "w")

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

except IOError:

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

else:

print ("Write operation is performed successfully on the file")

fob.close()

try:

fob = open("test", "r")

fob.write("It's my test file to verify exception handling in Python!!")

except IOError:

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

else:

print ("Write operation is performed successfully on the file")s

**Handling all type of exceptions:**

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.

**Handling Multiple Exception:**

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:**

This clause allows defining statements that we want to execute, no matters whether the try block has raised an exception or not.

try:

You do your operations here;

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

Due to any exception, this may be skipped.

finally:

This would always be executed.

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

try:

fob = open('test', 'w')

fob.write("It's my test file to verify try-finally in exception handling!!"

)

print 'try block executed'

finally:

fob.close()

print 'finally block executed'

try:

#fob = open('test', 'r')

#fob.write("It's my test file to verify try-finally in exception handling!!" )

print ('try block executed')

except:

print ('except block executed')

else:

print ('ddfdf')

finally:

print ('finally block executed')

**Raise Exception with Arguments**

**Syntax:**

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

* Under the “Exception” – specify its name.
* The “args” is optional and represents the value of the exception argument.
* The final argument, “traceback,” is also optional and if present, is the traceback object used for the exception.
* raise MemoryError

try:

a = int(input("Enter a positive integer value: "))

if a <= 0:

raise ValueError("This is not a positive number!!")

except ValueError as ve:

print(ve)

**Custom Exceptions:**

**Programmer creates himself.**

#define Python user-defined exceptions

class Error(Exception):

"""Base class for other exceptions"""

pass

class InputTooSmallError(Error):

"""Raised when the entered alpahbet is smaller than the actual one"""

pass

class InputTooLargeError(Error):

"""Raised when the entered alpahbet is larger than the actual one"""

pass

#our main program

#user guesses an alphabet until he/she gets it right

#you need to guess this alphabet

alphabet = 'm'

while True:

try:

apb = raw\_input("Enter an alphabet: ")

if apb < alphabet:

raise InputTooSmallError

elif apb > alphabet:

raise InputTooLargeError

break

except InputTooSmallError:

print("The entered alphabet is too small, try again!")

print('')

except InputTooLargeError:

print("The entered alphabet is too large, try again!")

print('')

print("Congratulations! You guessed it correctly.")

**Built in Exceptions**

BaseException

+-- SystemExit

+-- KeyboardInterrupt

+-- GeneratorExit

+-- Exception

+-- StopIteration

+-- StopAsyncIteration

+-- ArithmeticError

| +-- FloatingPointError

| +-- OverflowError

| +-- ZeroDivisionError

+-- AssertionError

+-- AttributeError

+-- BufferError

+-- EOFError

+-- ImportError

| +-- ModuleNotFoundError

+-- LookupError

| +-- IndexError

| +-- KeyError

+-- MemoryError

+-- NameError

| +-- UnboundLocalError

+-- OSError

| +-- BlockingIOError

| +-- ChildProcessError

| +-- ConnectionError

| | +-- BrokenPipeError

| | +-- ConnectionAbortedError

| | +-- ConnectionRefusedError

| | +-- ConnectionResetError

| +-- FileExistsError

| +-- FileNotFoundError

| +-- InterruptedError

| +-- IsADirectoryError

| +-- NotADirectoryError

| +-- PermissionError

| +-- ProcessLookupError

| +-- TimeoutError

+-- ReferenceError

+-- RuntimeError

| +-- NotImplementedError

| +-- RecursionError

+-- SyntaxError

| +-- IndentationError

| +-- TabError

+-- SystemError

+-- TypeError

+-- ValueError

| +-- UnicodeError

| +-- UnicodeDecodeError

| +-- UnicodeEncodeError

| +-- UnicodeTranslateError

+-- Warning

+-- DeprecationWarning

+-- PendingDeprecationWarning

+-- RuntimeWarning

+-- SyntaxWarning

+-- UserWarning

+-- FutureWarning

+-- ImportWarning

+-- UnicodeWarning

+-- BytesWarning

+-- ResourceWarning