
Python Programming
High performance. Delivered.
Module 12 - Exceptions
accenture
Strategy | Digital | Technology | Operations

Agenda

Exceptions in Python

- Exception Handling
- **Try, Except and Finally**
- Built-in exceptions
- Custom exceptions

Module Objectives

At the end of this module, you will be able to understand

- Exceptions ,
- Exception handling in Python
- Built-in Exceptions in Python and
- Custom Exceptions in Python

Exceptions in Python

- Errors can also occur at runtime and these are called exceptions. When ever an error occurs during runtime, Python creates an exception object.
- If not handled properly, it prints a message and the program is terminated.

- **Example 1**

```
a = 10;
b = 0;
c = a/b;

print ("C = ",c);
```

OUTPUT

```
Traceback (most recent call last):
  File "Exceptions.py", line 6, in <module>
    c = a/b;
ZeroDivisionError: division by zero
```

Exceptions in Python – (Contd)

Example 2

```
fh = open("XXXX.txt");
```

```
OUTPUT
Traceback (most recent call last):
  File "Exceptions.py", line 1, in <module>
    fh = open("XXXX.txt");
FileNotFoundError: [Errno 2] No such file or directory: 'XXXX.txt'
```

Built-in Exceptions

- Illegal operations can raise exceptions. There are plenty of built-in exceptions in Python that are raised when corresponding errors occur.
- Following are common built-in exceptions with the cause of error

Built-in Exceptions in Python

Exception	Cause of Error
AssertionError	Raised when assert statement fails.
AttributeError	Raised when attribute assignment or reference fails.
EOFError	Raised when the input() functions hits end-of-file condition.
FloatingPointError	Raised when a floating point operation fails.
GeneratorExit	Raise when a generator's close() method is called.
ImportError	Raised when the imported module is not found.
IndexError	Raised when index of a sequence is out of range.
KeyError	Raised when a key is not found in a dictionary.
KeyboardInterrupt	Raised when the user hits interrupt key (Ctrl+c or delete).
MemoryError	Raised when an operation runs out of memory.
NameError	Raised when a variable is not found in local or global scope.
NotImplementedError	Raised by abstract methods.
OSError	Raised when system operation causes system related error.
OverflowError	Raised when result of an arithmetic operation is too large to be represented.

Built-in Exceptions in Python

Exception	Cause of Error
ReferenceError	Raised when a weak reference proxy is used to access a garbage collected referent.
RuntimeError	Raised when an error does not fall under any other category.
StopIteration	Raised by next() function to indicate that there is no further item to be returned by iterator.
SyntaxError	Raised by parser when syntax error is encountered.
IndentationError	Raised when there is incorrect indentation.
TabError	Raised when indentation consists of inconsistent tabs and spaces.
SystemError	Raised when interpreter detects internal error.
SystemExit	Raised by sys.exit() function.
TypeError	Raised when a function or operation is applied to an object of incorrect type.

Built-in Exceptions in Python

Exception	Cause of Error
UnboundLocalError	Raised when a reference is made to a local variable in a function or method, but no value has been bound to that variable.
UnicodeError	Raised when a Unicode-related encoding or decoding error occurs.
UnicodeEncodeError	Raised when a Unicode-related error occurs during encoding.
UnicodeDecodeError	Raised when a Unicode-related error occurs during decoding.
UnicodeTranslateError	Raised when a Unicode-related error occurs during translating.
ValueError	Raised when a function gets argument of correct type but improper value.
ZeroDivisionError	Raised when second operand of division or modulo operation is zero.

Exception Handling in Python – Try, Except and Finally

- When an exception occurs in Python, it causes the current process to stop and passes it to the calling process until it is handled.
- If not handled, the program is terminated.
- The programmer identifies a critical operation which can raise exception and places this statement inside the try clause and the code that handles exception is written in except clause. The code in the except clause is called the “handler”.
- When an exception is thrown from the try block, then that exception is caught in the Except block.

Exception Handling - Example

```
import sys;
print("Hello, Welcome to Exception Handling...");
try:
    x = int(input("Enter an integer: "));
    r = 1/x;
    print("Mathetical Operation Successful....");
except:
    print("Sorry Something went Wrong....");
    print("Exception Handled....");

print("========== END of Program ================");
```

OUTPUT

Option1:

```
Hello, Welcome to Exception Handling...
Enter an integer: 40
Mathetical Operation Successful....
========== END of Program ============
```

Option2:

```
Hello, Welcome to Exception Handling...
Enter an integer: 0
Sorry Something went Wrong....
Exception Handled....
======== END of Program ============
```

Catching Specific Exceptions in Python

- If the code in the try block throws more than one exception, then multiple exceptions can be handled in multiple except blocks.
- If no specific exception is mentioned , then the except clause can handle any exception. This is not a good programming practice as it will catch all exceptions and handle every case in the same way.
- It is possible to specify which exceptions an except clause will catch. A try clause can have any number of except clauses to handle them differently but only one will be executed in case an exception occurs.

Catching Specific Exceptions in Python -Example

```
import sys;
print("Hello, Welcome to Exception Handling...");

try:
    x = int(input("Enter an integer: "));
    r = 1/x;
    print("Mathetical Operation Successful....");
    fh = open("XXXX.txt");
    print("File Operation Successful....");
except ZeroDivisionError :
    print("Sorry Something went Wrong....");
    print("ZeroDivisionError Exception Handled....");
except FileNotFoundError:
    print("Sorry Something went Wrong....");
    print("FileNotFoundError Exception Handled....");
print("========== END of Program ================");
```

Catching Specific Exceptions in Python -Example

OUTPUT

Option1:

```
Hello, Welcome to Exception Handling...
Enter an integer: 10
Mathetical Operation Successful....
Sorry Something went Wrong....
FileNotFoundError Exception Handled....
======= END of Program =====
```

Option2:

```
Hello, Welcome to Exception Handling...
Enter an integer: 0
Sorry Something went Wrong....
ZeroDivisionError Exception Handled....
====== END of Program ========
```

Raising Exceptions

Exceptions are automatically raised when corresponding errors occur at run time,

Exceptions can also be forcefully raised using the keyword raise.

An optional value can also be passed to the to the exception to clarify.

Raising Exceptions - Example

```
import sys;
print("Hello, Welcome to Exception Handling...");

try:
    x = int(input("Enter an integer: "));
    if x == 0:
        raise ValueError("Denominator cannot be ZERO!");

    r = 1/x;
    print("Mathetical Operation Successful....");
    fh = open("XXXX.txt");
    print("File Operation Successful....");
except ValueError:
    print("Sorry Something went Wrong....");
    print("ValueError Exception Handled....");
except ZeroDivisionError:
    print("Sorry Something went Wrong....");
    print("ZeroDivisionError Exception Handled....");
except FileNotFoundError:
    print("Sorry Something went Wrong....");
    print("FileNotFoundError Exception Handled....");
print("======== END of Program ============");
```

OUTPUT

```
Hello, Welcome to Exception Handling...
Enter an integer: 0
Sorry Something went Wrong....
ValueError Exception Handled....
======= END of Program =========
```

Argument of an Exception

- An exception can have an *argument*, which is a value that gives additional information about the problem. The contents of the argument vary by exception.
- You can capture an exception's argument by supplying a variable in the except clause as follows
- Example:

```
try:
   statements;
   ......................
except ExceptionType, Argument:
   You can print value of Argument here...
```

try...finally

- The try statement in Python can have an optional finally clause.
- Finally clause is executed no matter whether an exception is thrown or not.

Try Finally - Example

```
import sys;
print("Hello, Welcome to Exception Handling...");

try:
    x = int(input("Enter an integer: "));
    if x == 0:
        raise ValueError("Denominator cannot be ZERO!");
    r = 1/x;
    print("Mathetical Operation Successful....");
except ValueError:
    print("Sorry Something went Wrong....");
    print("ValueError Exception Handled....");
except ZeroDivisionError:
    print("Sorry Something went Wrong....");
    print("ZeroDivisionError Exception Handled....");
except FileNotFoundError:
    print("Sorry Something went Wrong....");
    print("FileNotFoundError Exception Handled....");
finally:
    print("====== I am Finally Block ==========");

print("======== END of Program ============");
```

- OUTPUT

Option1: **Hello, Welcome to Exception Handling...** **Enter an integer: 0** **Sorry Something went Wrong....** **ValueError Exception Handled....** **==== I am Finally Block ====** **====== END of Program ====**	**Option2:** **Hello, Welcome to Exception Handling...** **Enter an integer: 40** **Mathetical Operation Successful....** **==== I am Finally Block ====** **===== END of Program ======**

User-Defined Exception

- Users can define their own exception by creating a new class in Python.
- This exception class has to be derived, either directly or indirectly, from Exception class.
- Most of the built-in exceptions are also derived form this class.

User-Defined Exception - Example

```
class LoginErrors(Exception):
   """Base class for other LoginErrors"""
   pass
class BlankUsernameError(LoginErrors):
   """Raised when the Username is Blank"""
   pass

class WrongUsernameError(LoginErrors):
   """Raised when the Username is Wrong"""
   pass

class BlankPasswordError(LoginErrors):
   """Raised when the Username is Blank"""
   pass

class WrongPassWordError(LoginErrors):
   """Raised when the Password is Wrong"""
   pass
```

User-Defined Exception – (Contd)

- In this example, we have created a user-defined exception called LoginErrors which is derived from the Exception class.
- This new exception can be raised, like other exceptions, using the raise statement with an optional error message.
- When we are developing a large Python program, it is a good practice to place all the user-defined exceptions that our program raises in a separate file as exceptions.py or errors.py.
- User-defined exception class can implement everything a normal class can do, but we generally make them simple and concise. Most implementations declare a custom base class and derive others exception classes from this base class. This concept is made clearer in the following example.

User-Defined Exception – Example

- In the below example, we have defined user-defined exceptions to catch all kinds of UserLogin errors and handled it.

```
class LoginErrors(Exception):
   """Base class for other LoginErrors"""
   pass
class BlankUsernameError(LoginErrors):
   """Raised when the Username is Blank"""
   pass

class WrongUsernameError(LoginErrors):
   """Raised when the Username is Wrong"""
   pass

class BlankPasswordError(LoginErrors):
   """Raised when the Username is Blank"""
   pass

class WrongPassWordError(LoginErrors):
   """Raised when the Password is Wrong"""
   pass
```

User-Defined Exception – Example

```
try:
    uname = input("Enter Your Username: ");
    pword = input("Enter Your Password: ");
    if uname == '':
        raise BlankUsernameError
    elif uname != "accenture":
        raise WrongUsernameError
    elif pword == '':
        raise BlankPasswordError
    elif pword != "bangalore":
        raise WrongPassWordError
    print("Congratulations! You Login is Successful...")

except BlankUsernameError:
    print("Sorry , Your Username cannot be Blank!");
    print();
except WrongUsernameError:
    print("Sorry Username Doesnot Exists!");
    print();
except BlankPasswordError:
    print("Sorry , Your Password cannot be Blank!");
    print();
except WrongPassWordError:
    print("Sorry Username Doesnot Exists!")
    print()
```

User-Defined Exception – Example

OUTPUTS:

Option1: **Enter Your Username:** **Enter Your Password:** **bangalore** **Sorry , Your Username cannot be Blank!**	**Option2:** **Enter Your Username:** **Accenture** **Enter Your Password:** **bangalore** **Sorry Username Doesnot Exists!**
Option3: Enter Your Username: accenture Enter Your Password: Sorry , Your Password cannot be Blank!	Option4: Enter Your Username: accenture Enter Your Password: bangalore Congratulations! You Login is Successful...

Questions