Home (/) >> Exception Handling in Python

Jul. 11, 2013

Error Handling (/error-handling/) exceptions (/exceptions/)

Exception Handling in Python

Overview

```
In this post we will cover how Python handles errors with exceptions.
```

What is an Exception?

```
An exception is an error that happens during execution of a program. When that
error occurs, Python generate an exception that can be handled, which avoids your
program to crash.
```

Why use Exceptions?

```
Exceptions are convenient in many ways for handling errors and special conditions
in a program. When you think that you have a code which can produce an error then
you can use exception handling.
```

Raising an Exception

```
You can raise an exception in your own program by using the raise exception
statement.

Raising an exception breaks current code execution and returns the exception
back until it is handled.
```

Exception Errors

```
Below is some common exceptions errors in Python:
```

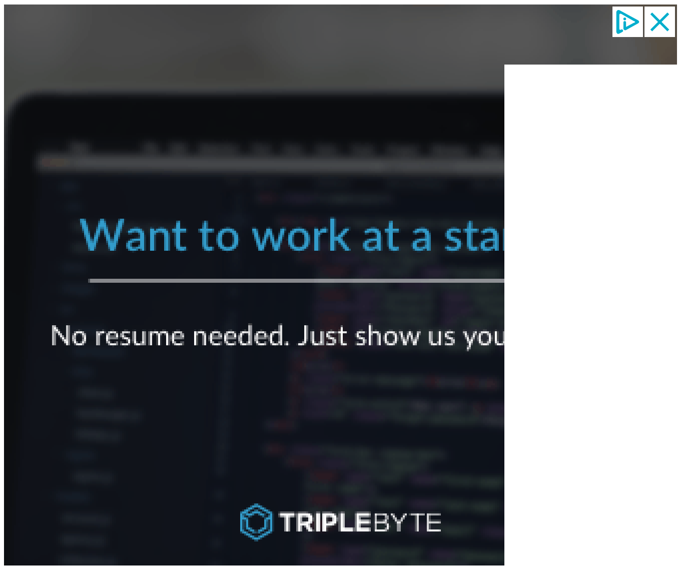
```
IOError
If the file cannot be opened.

ImportError
If python cannot find the module

ValueError
Raised when a built-in operation or function receives an argument that has the
right type but an inappropriate value

KeyboardInterrupt
Raised when the user hits the interrupt key (normally Control-C or Delete)

EOFError
Raised when one of the built-in functions (input() or raw_input()) hits an
end-of-file condition (EOF) without reading any data
```



Exception Errors Examples

```
Now, when we know what some of the exception errors means, let's see some
examples:
```

```
except IOError:
    print('An error occurred trying to read the file.')

except ValueError:
    print('Non-numeric data found in the file.')

except ImportError:
    print "NO module found"

except EOFError:
    print('Why did you do an EOF on me?')

except KeyboardInterrupt:
    print('You cancelled the operation.')

except:
    print('An error occurred.')
```

```
Try to use as few try blocks as possible and try to distinguish the failure
conditions by the kinds of exceptions they throw.
```

Set up exception handling blocks

To use exception handling in Python, you first need to have a catch-all except
clause.

The words "try" and "except" are Python keywords and are used to catch exceptions.

try-except [exception-name] (see above for examples) **blocks**

The code within the try clause will be executed statement by statement.

If an exception occurs, the rest of the try block will be skipped and the
except clause will be executed.

```
try:
    some statements here
except:
    exception handling
```

Let's see a short example on how to do this:

```
try:
    print 1/0

except ZeroDivisionError:
    print "You can't divide by zero, you're silly."
```

How does it work?

The error handling is done through the use of exceptions that are caught in try blocks and handled in except blocks. If an error is encountered, a try block code execution is stopped and transferred down to the except block.

In addition to using an except block after the try block, you can also use the finally block.

The code in the finally block will be executed regardless of whether an exception occurs.

Code Example

Let's write some code to see what happens when you not use error handling in your program.

This program will ask the user to input a number between 1 and 10, and then print the number.

```
number = int(raw_input("Enter a number between 1 - 10"))

print "you entered number", number
```

This program works perfectly fun as long as the user enters a number, but what happens if the user puts in something else (like a string)?

```
Enter a number between 1 - 10
hello
```

You can see that the program throws us an error when we enter a string.

```
Traceback (most recent call last):
  File "enter_number.py", line 1, in
    number = int(raw_input("Enter a number between 1 - 10
"))
ValueError: invalid literal for int() with base 10: 'hello'
```

```
ValueError is an exception type. Let's see how we can use exception handling to
fix the previous program
```

```
import sys

print "Lets fix the previous code with exception handling"

try:
    number = int(raw_input("Enter a number between 1 - 10
"))

except ValueError:
    print "Err.. numbers only"
    sys.exit()

print "you entered number
", number
```

```
If we now run the program, and enter a string (instead of a number), we can see
that we get a different output.
```

```
Lets fix the previous code with exception handling
Enter a number between 1 - 10
hello
Err.. numbers only
```

Try ... except ... else clause

```
The else clause in a try , except statement must follow all except clauses, and is
useful for code that must be executed if the try clause does not raise an
exception.
```

```
try:
    data = something_that_can_go_wrong

except IOError:
    handle_the_exception_error

else:
    doing_different_exception_handling
```

```
Exceptions in the else clause are not handled by the preceding except clauses.

Make sure that the else clause is run before the finally block.
```

Try ... finally clause

```
The finally clause is optional. It is intended to define clean-up actions that
must be executed under all circumstances
```

```
try:
    raise KeyboardInterrupt
finally:
    print 'Goodbye, world!'
...
Goodbye, world!
KeyboardInterrupt
```

```
A finally clause is always executed before leaving the try statement, whether an
exception has occurred or not.
```

```
Remember that if you don't specify an exception type on the except line, it will
catch all exceptions, which is a bad idea, since it means your program will ignore
unexpected errors as well as ones which the except block is actually prepared to
handle.
```

More Reading

```
http://en.wikibooks.org/wiki/Python_Programming/Exceptions (http://en.wikibooks.org/wiki/Python_P
rogramming/Exceptions)
http://www.linuxjournal.com/article/5821 (http://www.linuxjournal.com/article/5821)
http://docs.python.org/2/library/exceptions.html (http://docs.python.org/2/library/exceptions.htm
l)
http://docs.python.org/2/tutorial/errors.html (http://docs.python.org/2/tutorial/errors.html)
http://stackoverflow.com/questions/855759/python-try-else (http://stackoverflow.com/questions/855
759/python-try-else)
```