

RuntimeError	Raised when a generated error does not fall into any category.
NotImplementedError	Raised when an abstract method that needs to be implemented in an inherited class is not actually implemented.

Assertions in Python

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

The easiest way to think of an assertion is to liken it to a **raise-if** statement (or to be more accurate, a raise-if-not statement). An expression is tested, and if the result comes up false, an exception is raised.

Assertions are carried out by the assert statement, the newest keyword to Python, introduced in version 1.5.

Programmers often place assertions at the start of a function to check for valid input, and after a function call to check for valid output.

The *assert* Statement

When it encounters an assert statement, Python evaluates the accompanying expression, which is hopefully true. If the expression is false, Python raises an *AssertionError* exception.

The syntax for assert is:

```
assert Expression[, Arguments]
```

If the assertion fails, Python uses *ArgumentExpression* as the argument for the *AssertionError*. *AssertionError* exceptions can be caught and handled like any other exception using the try-except statement, but if not handled, they will terminate the program and produce a traceback.

Example

Here is a function that converts a temperature from degrees Kelvin to degrees Fahrenheit. Since zero degrees Kelvin is as cold as it gets, the function bails out if it sees a negative temperature:

```
#!/usr/bin/python

def KelvinToFahrenheit(Temperature):
```