`assert` Statement

The built-in keyword called `assert` borrows the meaning from the word **assertion**.

> **assertion**: a confident and forceful statement of fact or belief.

In Python, `assert` is a built-in Python keyword which we can use to check if a certain condition is `True` or `False`. It is a commonly used keyword for debugging Python programs. Imagine writing a function that returns a value. One of the ways in which you can test whether or not the function returns an expected value is to use an `if` statement; compare the returned value with an expected value.

```
# foo.py

def multiply_by_two(param):
    """Given an integer value param, return param multiplied by 2."""
    return param * 2

if __name__ == "__main__":
    # Test if multiply_by_two() function behaves in an expected way.
    ret = multiply_by_two(2)
    if ret != 4:
        print "%s is not equal to 4" % ret
```

The above example shows a way to test your functions by checking the returned value against an expected value by using the `if` statement. There are many Python programs used in practice that use this approach to handle unexpected output.

However, in a situation where you **must** have your Python program to **strictly** behave in a certain way, it is often advised to use the `assert` statement or a Python `unittest` library to conduct proper tests. In the rest of this document, we will focus on the usage of the `assert` statement for debugging your Python code.

Let's revisit the above example, but this time, we will implement the code with the `assert` keyword.

```python
# bar.py

def multiply_by_two(param):
    """Given an integer value param, return param multiplied by 2."""
    return param * 2

if __name__ == "__main__":
    # Test if multiply_by_two() function behaves in an expected way.
    ret = multiply_by_two(2)
    assert ret == 4
```

At a first glance, the `assert` statement is much simpler than the `if` statement, but there is one major difference between the two. The `assert` statement **raises** an **AssertionError** exception and immediately terminates the program. In many cases during the development stage of your program, such behavior is very much desirable, especially for the parts of the code that **must** behave in a very specific and predictable manner.

Main Takeaway

The key point to remember about the `assert` statement is the following.

> `assert` statement is a convenient way to debug programs.

> When writing an `assert` statement, there always follows a **conditional expression**. `assert <cond-expression>`