What is the use of “assert” in Python?

I have been reading some source code and in several places I have seen the usage of `assert`.

What does it mean exactly? What is its usage?

python assert assertions

edited Dec 3 '14 at 3:13 — APerson — 3,711 4 22 48

asked Feb 28 '11 at 13:11 — Hossein — 8,327 36 97 150

15 docs.python.org/release/2.5.2/ref/assert.html – Joachim Sauer Feb 28 '11 at 13:13

stackoverflow.com/questions/944592/… – Russell Dias Feb 28 '11 at 13:13

14 Answers

The `assert` statement exists in almost every programming language. When you do...

```
assert condition
```

... you're telling the program to test that condition, and trigger an error if the condition is false.

In Python, it's roughly equivalent to this:

```
if not condition:
    raise AssertionError()
```

Try it in the Python shell:

```
>>> assert True
>>> assert False
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AssertionError
```

Assertions can include an optional message, and you can disable them when you're done debugging. See here for the relevant documentation.

edited Jul 1 '16 at 16:05

answered Feb 28 '11 at 13:15 — slezica — 31.8k 11 60 114

31 Nit: assert is a statement and not a function. And unlike print, in Python 3 it's still a statement. – BobStein-VisiBone Sep 16 '14 at 16:45

22 syntax for the optional message: `assert False, "You have asserted something false."` Also see this answer for gotchas. – bahmait Jun 18 '15 at 8:07

Watch out for the parentheses. As has been pointed out above, in Python 3, `assert` is still a statement, so by analogy with `print(..)`, one may extrapolate the same to `assert(..)` or `raise(..)` but you shouldn't.

This is important because:

```
assert(2 + 2 == 5, "Houston we've got a problem")
```

won't work, unlike

```
assert 2 + 2 == 5, "Houston we've got a problem"
```

The reason the first one will not work is that `bool( (False, "Houston we've got a problem") )` evaluates to `True`.

In the statement `assert(False)`, these are just redundant parentheses around `False`, which evaluate to their contents. But with `assert(False,)` the parentheses are now a tuple, and a non-empty tuple evaluates to `True` in a boolean context.

edited Apr 6 '16 at 19:31
kmario23
4,579 1 17 38

answered Jun 11 '15 at 2:15
Evgeni Sergeev
6,833 7 46 65

I came here looking for this exact info about parens and the follow message. Thanks. – superbeck Jul 11 '16 at 18:41

As other answers have noted, `assert` is similar to throwing an exception if a given condition isn't true. An important difference is that assert statements get ignored if you compile your code with the optimization option. The documentation says that `assert expression` can better be described as being equivalent to

```
if __debug__:
   if not expression: raise AssertionError
```

This can be useful if you want to thoroughly test your code, then release an optimized version when you're happy that none of your assertion cases fail - when optimization is on, the `__debug__` variable becomes False and the conditions will stop getting evaluated. This feature can also catch you out if you're relying on the asserts and don't realize they've disappeared.

answered Feb 28 '11 at 14:10
Neil Vass
2,498 15 19

Others have already given you links to documentation.

You can try the following in a interactive shell:

```
>>> assert 5 > 2
>>> assert 2 > 5
Traceback (most recent call last):
  File "<string>", line 1, in <fragment>
builtins.AssertionError:
```

The first try does nothing, while the second raises and exception. This is the first hint: asserts are useful to check conditions that should be true in a given position of your code (usually, the beginning (preconditions) and the end of a function (postconditions)).

asserts are actually highly tied to programming by contract, which is a very useful engineering practice:

http://en.wikipedia.org/wiki/Design_by_contract

edited Jul 17 '15 at 8:19

answered Feb 28 '11 at 13:18


Baltasarq
7,410 19 40

So does that mean we can check in code in a situation like assert(2 > 5) and raise error else continue ? – user1176501 Oct 17 '13 at 6:25

1 Yes, that's it. – Baltasarq Oct 18 '13 at 16:41

14 Lose the parens, assert is not a function. – pillmuncher Feb 17 '14 at 15:27

1 Losing the parens is more important than it seems. See below. – Evgeni Sergeev Jul 16 '15 at 11:50

2 Assert actually dates back (long before "contracts") to Turing, when he wrote one of the earliest papers on how programmers might tackle the rather daunting task of creating correct programs. Finding that paper is left as an exercise for the reader, since all programmers can benefit from becoming familiar with his work. :-) turingarchive.org – Ron Burk Oct 31 '16 at 23:41

The assert statement has two forms.

The simple form, `assert <expression>` , is equivalent to

```
if __debug__:
    if not <expression>: raise AssertionError
```

The extended form, `assert <expression1>, <expression2>` , is equivalent to

```
if __debug__:
    if not <expression1>: raise AssertionError, <expression2>
```

edited Oct 15 '14 at 20:54

Colin D Bennett
4,243 2 23 37

answered Jul 10 '13 at 1:21

Bohdan
6,365 6 40 53

Assertions are a systematic way to check that the internal state of a program is as the programmer expected, with the goal of catching bugs. See the example below.

```
>>> number = input('Enter a positive number:')
Enter a positive number:-1
>>> assert (number > 0), 'Only positive numbers are allowed!'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AssertionError: Only positive numbers are allowed!
>>>
```

edited Feb 19 '14 at 16:58

answered Feb 19 '14 at 16:52

Jacob Abraham
513 5 6

1 Also, assertions can often be used in unit testing programs. stackoverflow.com/questions/1383/what-is-unit-testing – panofish Oct 2 '14 at 18:29

From docs:

```
Assert statements are a convenient way to insert debugging assertions into a program
```

Here you can read more: http://docs.python.org/release/2.5.2/ref/assert.html

answered Feb 28 '11 at 13:16

gruszczy
22.9k 13 92 137

Here is a simple example, save this in file (let's say b.py)

```
def chkassert(num):
    assert type(num) == int


chkassert('a')
```

and the result when `$python b.py`

```
Traceback (most recent call last):
  File "b.py", line 5, in <module>
```

```
    chkassert('a')
  File "b.py", line 2, in chkassert
    assert type(num) == int
AssertionError
```

answered Feb 17 '14 at 14:54

Gaurav Agarwal
6,587 22 78 140

The goal of an assertion in Python is to inform developers about **unrecoverable** errors in a program.

Assertions are not intended to signal expected error conditions, like “file not found”, where a user can take corrective action (or just try again).

Another way to look at it is to say that assertions are **internal self-checks** in your code. They work by declaring some conditions as *impossible* in your code. If these conditions don’t hold that means there’s a bug in the program.

If your program is bug-free, these conditions will never occur. But if one of them *does* occur the program will crash with an assertion error telling you exactly which “impossible” condition was triggered. This makes it much easier to track down and fix bugs in your programs.

Here’s a summary from a tutorial on Python’s assertions I wrote:

> **Python’s assert statement is a debugging aid, not a mechanism for handling run-time errors.** The goal of using assertions is to let developers find the likely root cause of a bug more quickly. An assertion error should never be raised unless there’s a bug in your program.

answered Jan 18 at 14:04

dbader
2,352 1 10 12

---

if the statement after assert is true then the program continues , but if the statement after assert is false then the program gives an error.Simple as that.

eg:

```
assert 1>0   #normal execution
assert 0>1   #Traceback (most recent call last):
             #File "<pyshell#11>", line 1, in <module>
             #assert 0>1
             #AssertionError
```

edited Jun 25 '15 at 10:06

tom
15.7k 3 27 46

answered Nov 1 '14 at 5:05

abe312
1,683 14 15

If you ever want to know exactly what a reserved function does in python, type in
`help(enter_keyword)`

Make sure if you are entering a reserved keyword that you enter it as a string.

edited Jan 7 '16 at 3:36

answered Jul 16 '15 at 3:51

ytpillai
1,268 7 24

```
def getUser(self, id, Email):

    user_key = id and id or Email

    assert user_key
```

Can be used to ensure parameters are passed in the function call.

edited Feb 19 '16 at 21:00

Cleb
6,277 9 23 44

answered Mar 24 '15 at 11:54

user2725012
41 4

---

format : assert Expression[,arguments] When assert encounters a statement,Python evaluates the expression.If the statement is not true,an exception is raised(assertionError). 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:

```
def KelvinToFahrenheit(Temperature):
    assert (Temperature >= 0),"Colder than absolute zero!"
    return ((Temperature-273)*1.8)+32
print KelvinToFahrenheit(273)
print int(KelvinToFahrenheit(505.78))
print KelvinToFahrenheit(-5)
```

When the above code is executed, it produces the following result:

```
32.0
451
Traceback (most recent call last):
  File "test.py", line 9, in <module>
    print KelvinToFahrenheit(-5)
  File "test.py", line 4, in KelvinToFahrenheit
    assert (Temperature >= 0),"Colder than absolute zero!"
AssertionError: Colder than absolute zero!
```

edited Dec 26 '14 at 15:41

answered Dec 26 '14 at 15:16

user3423267

48 1 7

```
>>>this_is_very_complex_function_result = 9
>>>c = this_is_very_complex_function_result
>>>test_us = (c < 4)

>>> #first we try without assert
>>>if test_us == True:
    print("YES! I am right!")
else:
    print("I am Wrong, but the program still RUNS!")

I am Wrong, but the program still RUNS!


>>> #now we try with assert
>>> assert test_us
Traceback (most recent call last):
  File "<pyshell#52>", line 1, in <module>
    assert test_us
AssertionError
>>>
```

answered yesterday

rianhariadi.com

1
