Theory: Assert statement

17 minutes 0 / 5 problems solved

Skip this topic Start practicing

341 users solved this topic. Latest completion was about 10 hours ago.

Testing exists in a great variety of programming languages, its aim is to check whether your code contains errors and if so, what their causes are. It does not have to be hard, you can start with easy steps. In this topic, we are going to deal with the **assert statement**. It is a useful debugging tool that uses Boolean logic and checks whether a given expression is true or false. If the condition is `True`, a program will keep running. Otherwise, it will return the `AssertionError`.

§1. Syntax of assert statements

The `assert` keyword can be used in two ways.

```
# 1
assert <condition>

# 2
assert <condition>, <message>
```

The `<condition>` attribute shows what you want to test, so it is always required. When the condition is false, the `AssertionError` is raised. The optional `<message>` attribute may be used to specify the message displayed with the error. Now let's illustrate assert statements in action.

```
# 1
word = input("Enter a word: ")
assert word != "cat"
print("Your word is", word)

# 2
word = input("Enter a word: ")
message = "'cat' is the wrong word!"
assert word != "cat", message

print("Your word is", word)
```

For instance, if you enter the word `dog` in both examples, the condition `word != 'cat'` will be `True`, and you will have the following output:

```
# Your word is dog
```

However, if you happen to enter `cat`, the `AssertionError` will be raised. In the second case, an error message will also appear.

```
# 1
# Traceback (most recent call last):
#   File "main.py", line 2, in <module>
#     assert word != "cat"
# AssertionError

# 2
# Traceback (most recent call last):
#   File "main.py", line 3, in <module>

#     assert word != "cat", message

# AssertionError: 'Cat' is the wrong word!
```

As you can see, the `AssertionError` is a built-in error, so you can handle it with the `try`-`except` block.

Current topic:

Assert statement

Topic depends on:

✓ Exception handling 3★

Table of contents:

```
try:
    word = input("Enter a word: ")
    message = "'Cat' is a wrong word!"
    assert word != "cat", message
    print("Your word is", word)
except AssertionError as err:
    print(err)
```

We can also check several variables or use more sophisticated logical expressions with the help of the `assert` keyword:

```
x = 4
y = 2
assert (x ** 2 / y ** 2) - y != 2, "There are wrong values!"
print(x, y)
# Traceback (most recent call last):
#   File "main.py", line 3, in
#     assert (x ** 2/ y ** 2) - y != 2, "There are wrong values!"
# AssertionError: There are wrong values!
```

Finally, the `assert` keyword can be used with functions. In the example below the `AssertionError` is raised when the parameter `2` given to the function does not fulfill the condition in the `test_mark(i)`.

```
def test_mark(i):
    message = "This student got a bad mark!"
    assert i > 4, message
    return i


print(test_mark(5))  # 5
print(test_mark(2))
# Traceback (most recent call last):

#   File "main.py", line 7, in

#     print(test_mark(i))

#   File "main.py", line 3, in test_mark

#     assert i > 2, message

# AssertionError: This student got a bad mark!
```

§2. Assert vs Raise

You may have noticed that `raise` and `assert` are similar to each other. What is the actual difference between them?

- `raise` is used for raising an exception;
- `assert` is used for raising an exception if the given condition is `False`.

Let's analyze some examples.

```
word = input("Enter a word: ")
if word != "cat":
    print(word)
else:
    raise Exception("There is a wrong word!")
```

As you can see, the `raise` keyword together with the `if`-`else` statement is very similar to the `assert` keyword, but their purposes are different. In the first case, `raise` is used mainly for data validation, while `assert` is used for debugging.

Python documentation (The Assert Statement paragraph) also provides an explanation of how the `assert` keyword works. The first example shows raising the error without a message and the second one — with the message.

```
# 1
if __debug__:
   if not condition:
       raise AssertionError

# 2
if __debug__:
   if not condition:
       raise AssertionError, message
```

The `__debug__` is a built-in variable that is `True` by default and is only set to `False` when Python is started in an optimization mode, so the first lines can be interpreted as `if True`.

Mind the difference between the `raise` and the `assert` and use them wisely.

§3. Summary

In this topic, you have learned some basics of the `assert` keyword, a tool for program testing:

- the purpose of the `assert` is to check whether a condition is `False`;
- to use the `assert`, we should specify a mandatory attribute `<condition>` and an optional `<message>`;
- as opposed to the `raise`, the `assert` is used for debugging.

Now, let's proceed to the tasks.

Report a typo

28 users liked this theory. 0 didn't like it. What about you?

Start practicing

Comments (0) Hints (0) Useful links (0) Show discussion