

Austin Bingham
COFOUNDER - SIXTY NORTH
@austin_bingham



Robert Smallshire
COFOUNDER - SIXTY NORTH
@robsmallshire

Assertion Syntax

assert condition [, message]

Boolean expression

If False, raises AssertionError Optional error message Included in exception payload

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

Assertions monitor invariants.

A failing assertion points to a programming error.

```
>>> assert 5 > 2, "You are in a defective universe!"
>>>
```

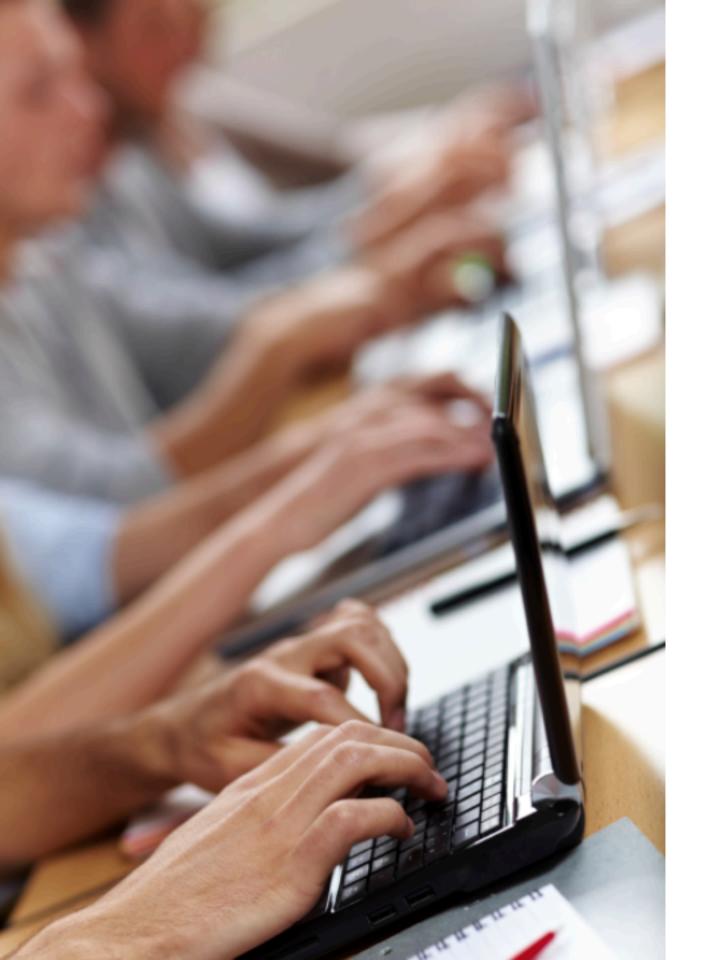
Document Assumptions



Assertions are best used to document conditions your program takes for granted

There are good and bad places to use assertions

```
mod ) is mod.py
                                                                                         🥏 words 🔻 🕨 🍎 🕟 😘 🔳 🔍
ill mod.py ×
                print("Remainder 1")
 6
            elif r == 2:
                print("Remainder 2")
            elif r == 3:
                print("Remainder 3")
10
            else:
                assert False, "This should never happen"
Python Console
```



Use assertions to check that your implementation is correct

In other words, check if the programmer has made a mistake

Do not use assertions to validate arguments

Class Invariants

```
sorted_set ) & sorted_set.py
                                                                                                    🥏 words ▼ 🕨 🍎 🕠 🕟 🗊 🔲 🔾
sorted_set.py ×
                   self._set = sorted(set(self._set))
12
13
                   assert self._is_unique_and_sorted()
14
             def contains(self, x):
15
16
                   assert self._is_unique_and_sorted()
17
                   index = bisect_left(self._set, x)
                   return index != len(self._set) and self._set[index] == x
18
19
             def _is_unique_and_sorted(self):
20
         SortedSet > contains()
 Python Console >
 =>>> from sorted_set import *
     >>> s = SortedSet([4, 3, 2, 1, 4, 3, 2, 1])
     >>> s._set
     [1, 2, 3, 4]
     >>> s.add(10)
   S.add(-16)
     >>> s.add(2001)
     >>> s._set
     [-16, 1, 2, 3, 4, 10, 2001]
     >>>
                                                                                        1:1 LF UTF-8 4 spaces Python 3.8 (code_editor) 🔓 🤼 🗇
Replay server listening on port 14415: You just opened sorted_set (a minute ago)
```

Assertions and Performance

Assertion cost

The SortedSet invariant checks are relatively expensive

Duplication

In some cases, more checks are made than are strictly necessary

Performance

While not necessarily bad, this can be detrimental to performance

python -0

Use Python's -0 argument to turn on basic optimizations.

In particular, this flag disables assertions.

Optimized Mode

```
$ python -m timeit -n 1 -s "from random import randrange; from sorted_set import
SortedSet; s = SortedSet(randrange(1000) for _ in range(2000))" "[s.contains(i)
for i in range(1000)]"
1 loop, best of 5: 283 msec per loop
$ python -O -m timeit -n 1 -s "from random import randrange; from sorted_set imp
ort SortedSet; s = SortedSet(randrange(1000) for _ in range(2000))" "[s.contains
(i) for i in range(1000)]"
1 loop, best of 5: 710 usec per loop
$
```

Only use -0 if performance concerns demand it.

Running in production with assertions helps flush out problems in your code.

Side-effects in Assertions



The absence or presence of assertions should not affect the correctness of your program.

Do not use assertion conditions that have side-effects.

Side-effects in Assertions

```
assert my_list.pop(item)
modifies the list
not run in optimized mode!
```



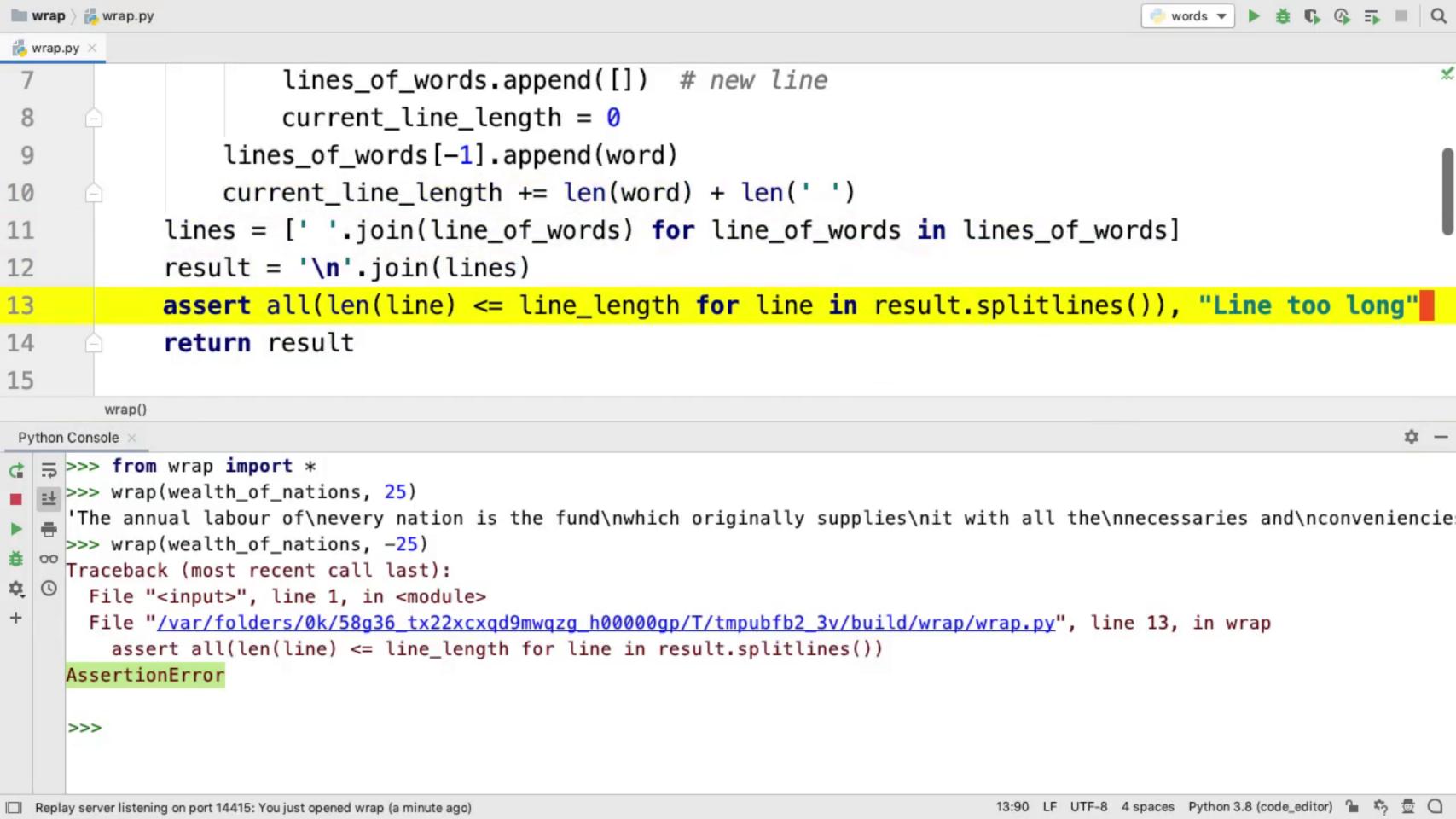
```
wrap \ wrap.py
                                                                                                 lines = [' '.join(line_of_words) for line_of_words in lines_of_words]
11
             result = '\n'.join(lines)
12
             assert all(len(line) <= line_length for line in result.splitlines())</pre>
13
             return result
14
15
16
        wealth_of_nations = "The annual labour of every nation is the fund which or" \
17
             "iginally supplies it with all the necessaries and conveniencies of life wh" \
18
             "ich it annually consumes, and which consist always either in the immediate" \
19
        wrap()
 Python Console X
                                                                                    Special Variables
                                                                                    ou wealth_of_nations = {str} 'The annual labour of ever... View
    Python Console
     >>> from wrap import *
    >>> wrap(wealth_of_nations, 25)
     Traceback (most recent call last):
       File "<input>", line 1, in <module>
       File "/var/folders/0k/58g36 tx22xcxqd9mwqzg h00000gp/T/tmpt1kauwdi/build/wrap/w
        assert all(len(line) <= line_length for line in result.splitlines())
     AssertionError
     >>>
                                                                                    1:1 LF UTF-8 4 spaces Python 3.8 (code_editor) 🚡
Replay server listening on port 14415: You just opened wrap (2 minutes ago)
```

Counting Bug

```
current_line_length = len('pelagic')
current_line_length += len('argosy')
```

"pelagic argosy"

space not accounted for



Conceptual Mismatch



Assertions should be used to detect errors in your implementation

The client is clearly at fault when passing bad data

Users will interpret assertion failures as errors in the implementation

```
wrap \ wrap.py
wrap.py ×
        def wrap(text, line_length):
              if line_length < 1:</pre>
                   raise ValueError("line_length {} is not positive".format(line_length))
             words = text.split()
              if (max(map(len, words))) > line_length:
                   raise ValueError("line_length must be at least as long as the longest word")
 9
        wrap() > if (max(map(len, words))) > lin...
 Python Console >
       File "/var/folders/0k/58g36_tx22xcxqd9mwqzg_h00000gp/T/tmp5hskw4op/build/wrap/wrap.py", line 3, in wrap
         raise ValueError("line_length {} is not positive".format(line_length))
     ValueError: line_length -25 is not positive
     >>> wrap('The next train to Llanfairpwllgwyngyllgogerychwyrndrobwllllantysiliogogogoch is at 16:32', 25)
     Traceback (most recent call last):
       File "<input>", line 1, in <module>
       File "/var/folders/0k/58g36_tx22xcxqd9mwqzg_h00000gp/T/tmp5hskw4op/build/wrap/wrap.py", line 16, in wrap
         assert all(len(line) <= line_length for line in result.splitlines()), "Line too long"
     AssertionError: Line too long
     >>>
                                                                                         8:85 LF UTF-8 4 spaces Python 3.8 (code_editor) 🚡 🤻
Replay server listening on port 14415: You just opened wrap (a minute ago)
```

Summary



Code with no obvious deficiencies
Not obviously without deficiencies
Assertions for enforcing conditions
Fixed error in code
Assertion triggered on invalid input
Added guards to check arguments