Ch11: Debugging &Unit Testing

305172 Computer Programming Laboratory
Jiraporn Pooksook
Naresuan University

Exercise

- Write a code to calculate the factorial of a non-negative number n.

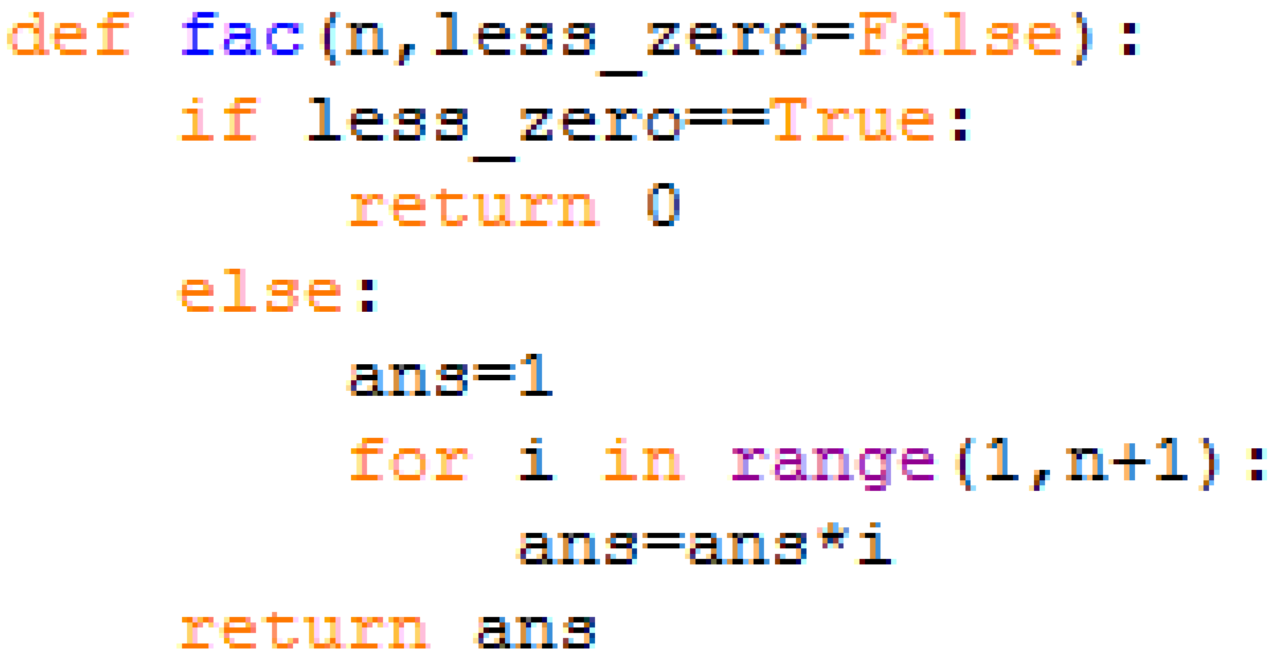
  Input: 5

  output: 120

  Input: 0

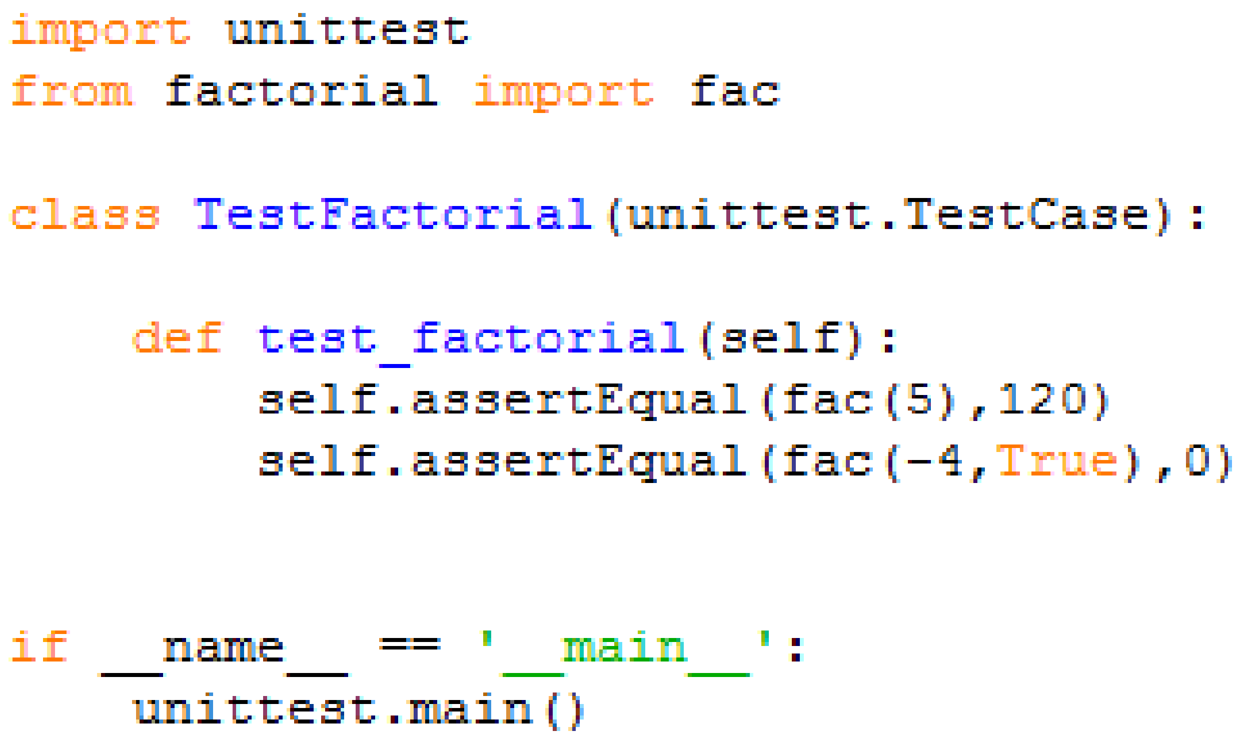
  output: 1

Factorial Code



```python
def fac(n,less_zero=False):
    if less_zero==True:
        return 0
    else:
        ans=1
        for i in range(1,n+1):
            ans=ans*i
    return ans
```

Unit Test Code



```python
import unittest
from factorial import fac

class TestFactorial(unittest.TestCase):

    def test_factorial(self):
        self.assertEqual(fac(5),120)
        self.assertEqual(fac(-4,True),0)


if __name__ == '__main__':
    unittest.main()
```

What is Unit Testing?

- *In computer programming, unit testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine if they are fit for use.*

Write a Unit Test in Python

```
import unittest
from factorial import fac

class TestFactorial(unittest.TestCase):

    def test_factorial(self):
        self.assertEqual(fac(5),120)
        self.assertEqual(fac(-4,True),0)


if __name__ == '__main__':
    unittest.main()
```

Import a function that we want to test from the source file

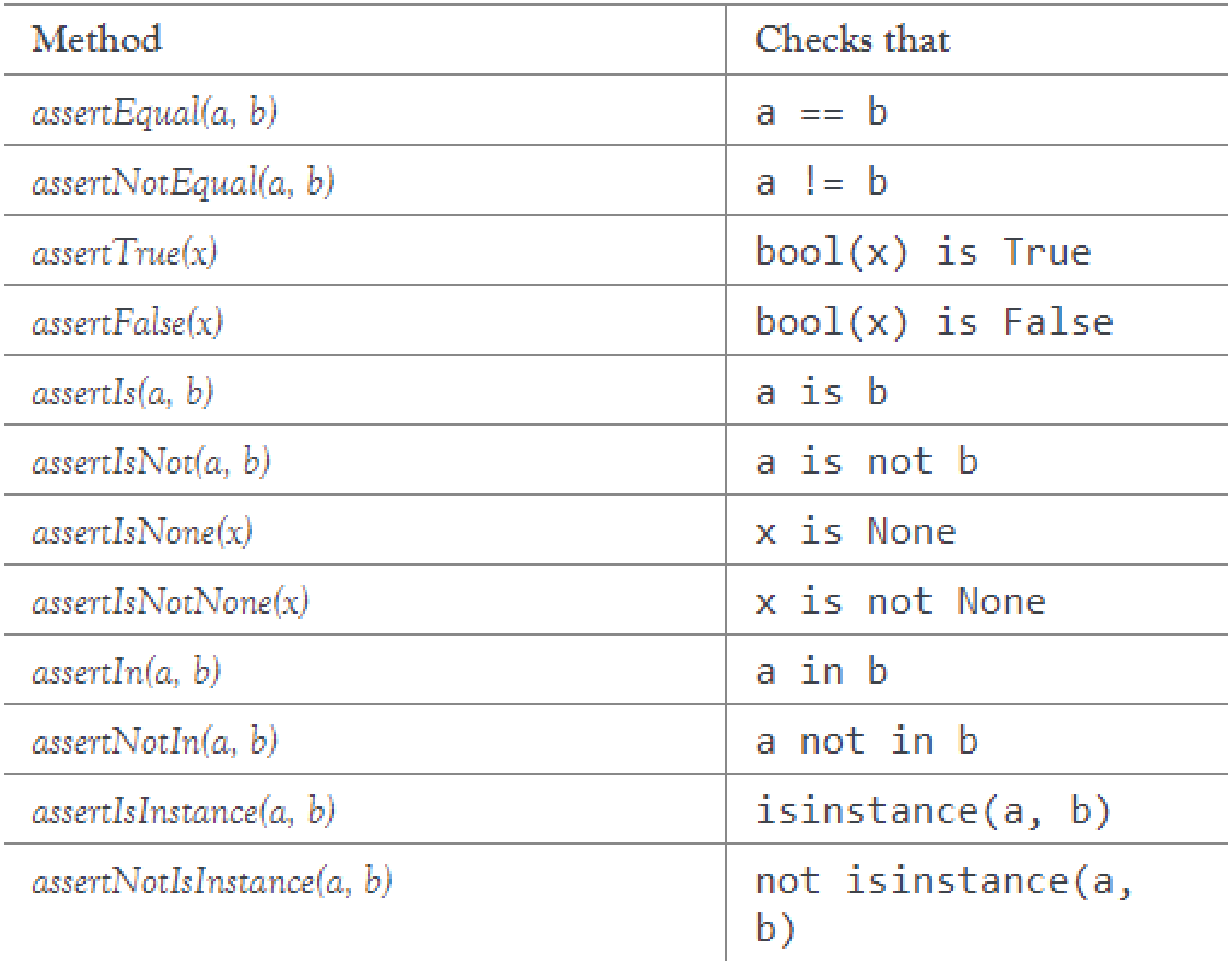
unittest library

Create a Test class extended from TestCase

assert statement

Run the test

Assert Statements



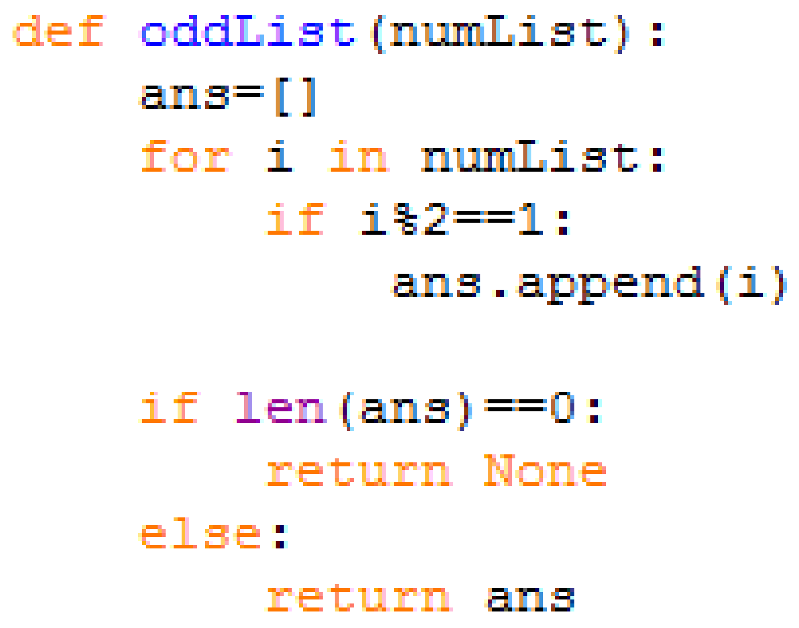
Method	Checks that
assertEqual(a, b)	`a == b`
assertNotEqual(a, b)	`a != b`
assertTrue(x)	`bool(x) is True`
assertFalse(x)	`bool(x) is False`
assertIs(a, b)	`a is b`
assertIsNot(a, b)	`a is not b`
assertIsNone(x)	`x is None`
assertIsNotNone(x)	`x is not None`
assertIn(a, b)	`a in b`
assertNotIn(a, b)	`a not in b`
assertIsInstance(a, b)	`isinstance(a, b)`
assertNotIsInstance(a, b)	`not isinstance(a, b)`

Exercise

- Write a code that receive a list of number as an input and then it returns a list of odd numbers as an output.
- Write a test code to test 2 examples:
  - Input: [1,2,3]
    - Output: [1,3]
  - Input: [2,4,6]
    - Output : None

Exercise

- Write a code that receive a list of number as an input and then it returns a list of odd numbers as an output.

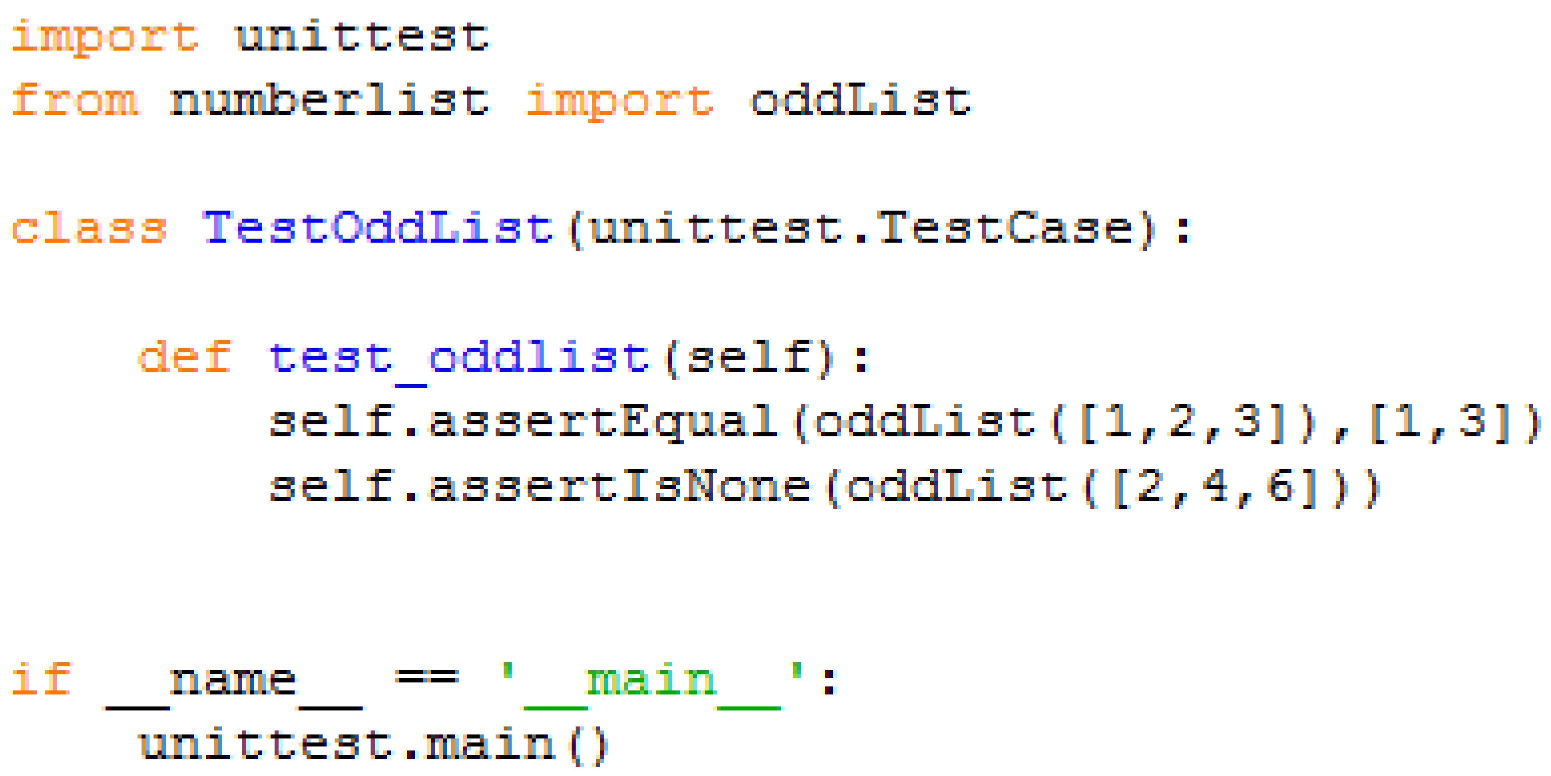


```
def oddList(numList):
    ans=[]
    for i in numList:
        if i%2==1:
            ans.append(i)

    if len(ans)==0:
        return None
    else:
        return ans
```

Exercise

- Write a test code.



```python
import unittest
from numberlist import oddList

class TestOddList(unittest.TestCase):

    def test_oddlist(self):
        self.assertEqual(oddList([1,2,3]),[1,3])
        self.assertIsNone(oddList([2,4,6]))


if __name__ == '__main__':
    unittest.main()
```

Exception Test

```
def div(x,y):
    return x/y
```

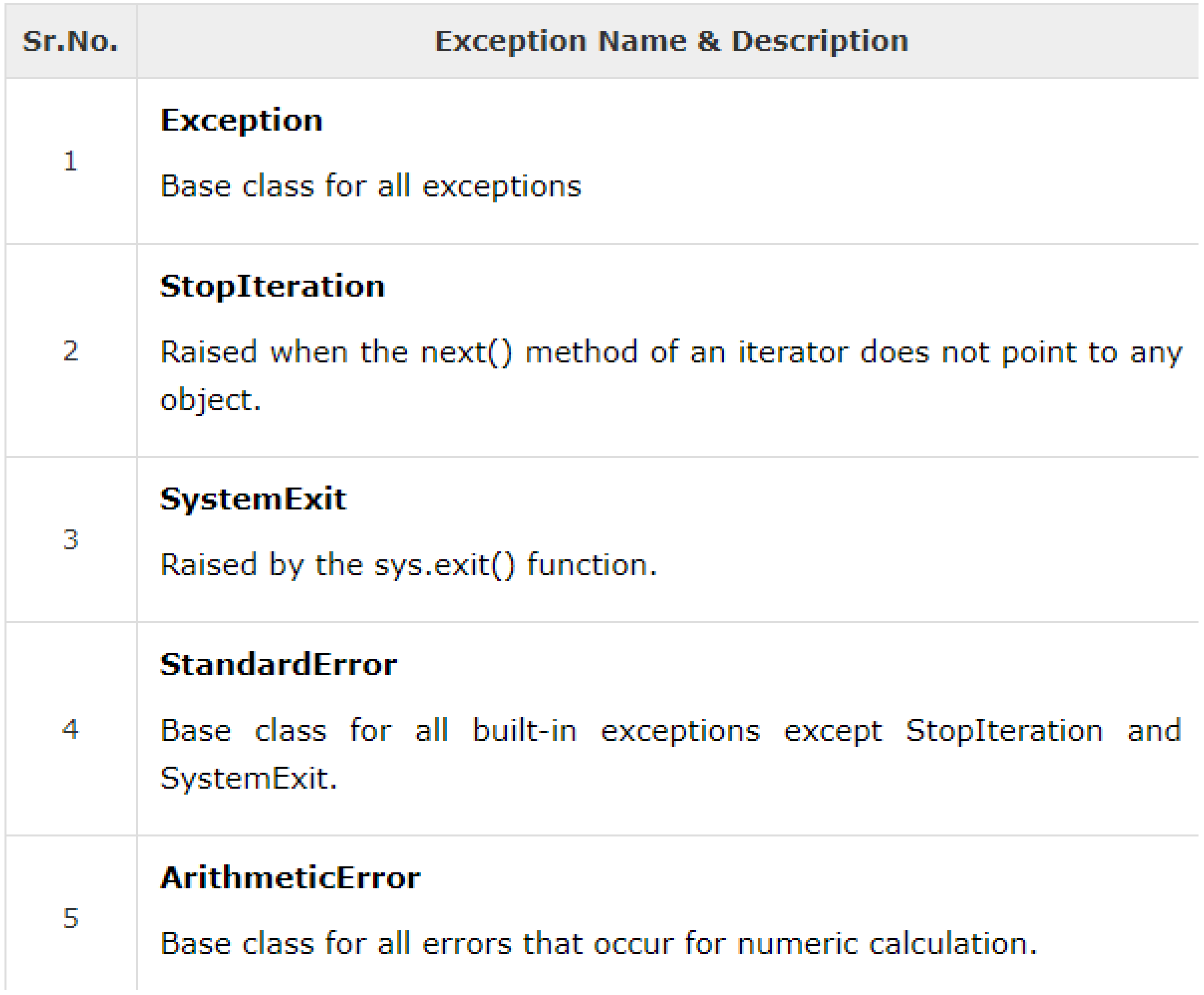
```
import unittest
from division import div

class TestDiv(unittest.TestCase):

    def test_div(self):
        self.assertRaises(ZeroDivisionError, div,3,2)
        self.assertRaises(ZeroDivisionError, div,3,0)


if __name__ == '__main__':
    unittest.main()
```

Exceptions



Sr.No.	Exception Name & Description
1	**Exception** Base class for all exceptions
2	**StopIteration** Raised when the next() method of an iterator does not point to any object.
3	**SystemExit** Raised by the sys.exit() function.
4	**StandardError** Base class for all built-in exceptions except StopIteration and SystemExit.
5	**ArithmeticError** Base class for all errors that occur for numeric calculation.

- References: https://docs.python.org/2/library/exceptions.html https://www.tutorialspoint.com/python/python_exceptions.htm

References

- https://pymbook.readthedocs.io/en/latest/testing.html
- https://docs.python.org/2/library/unittest.html

Exercise

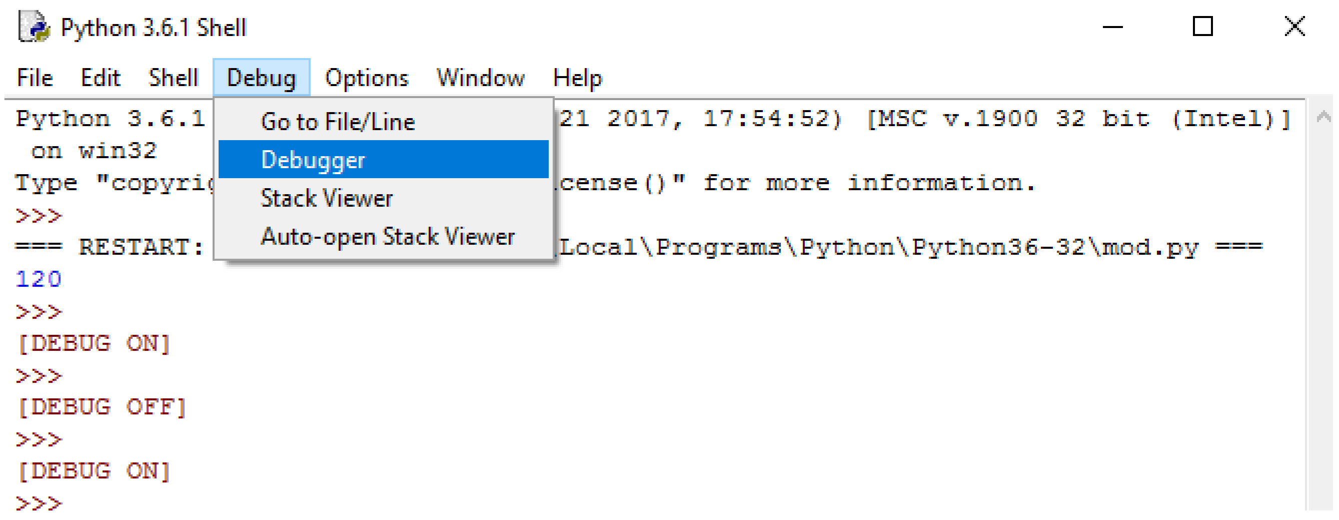
- เขียนฟังก์ชันที่รับอินพุตเลข 10 จำนวนแล้ว ทำการรีเทิร์นค่าผลรวมของตัวเลขทั้ง 10 จำนวน
- เขียน test code เพื่อทดสอบฟังก์ชันนี้ด้วย

- ตัวอย่าง

Input: 5 1 2 9 -3 7 8 2 0 -4

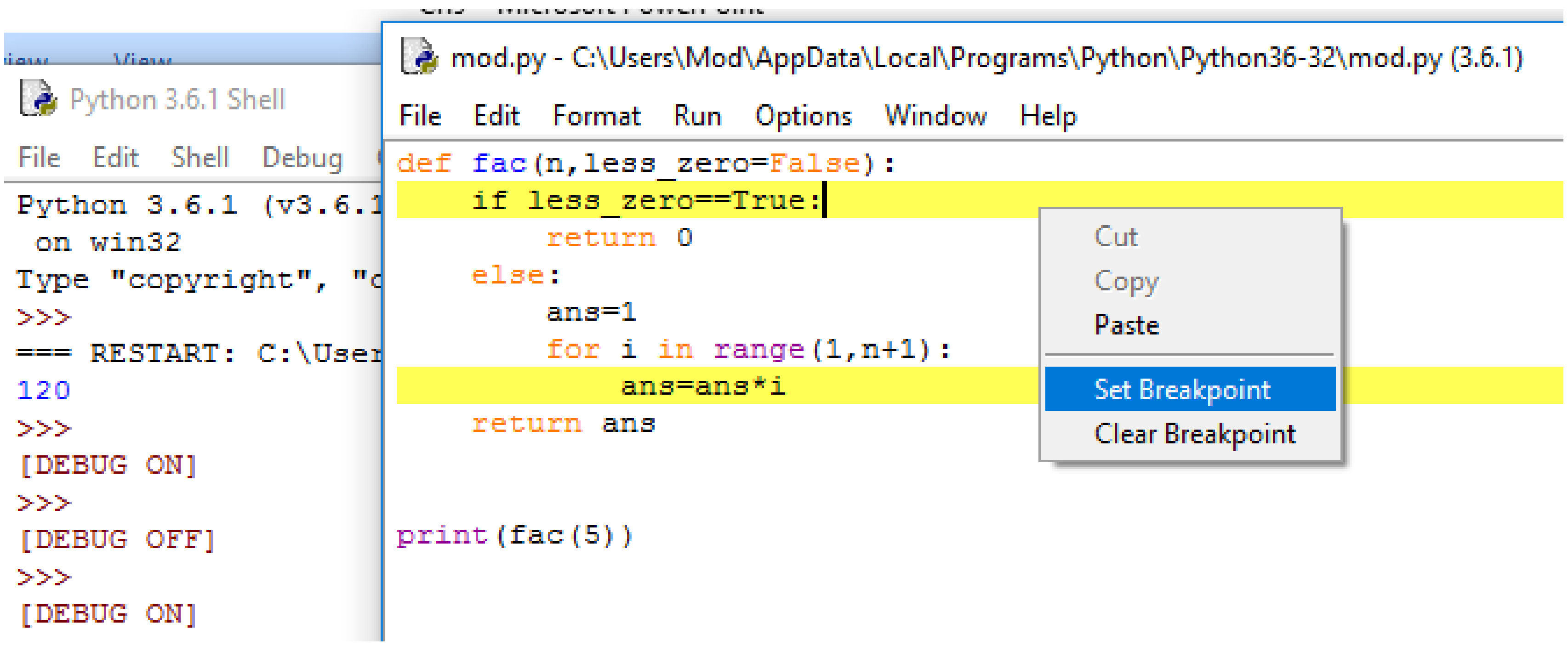
Output: 27

Debugging in Idle



```
Python 3.6.1                     21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyri                     cense()" for more information.
>>>
=== RESTART:                     Local\Programs\Python\Python36-32\mod.py ===
120
>>>
[DEBUG ON]
>>>
[DEBUG OFF]
>>>
[DEBUG ON]
>>>
```

Debugging in Idle



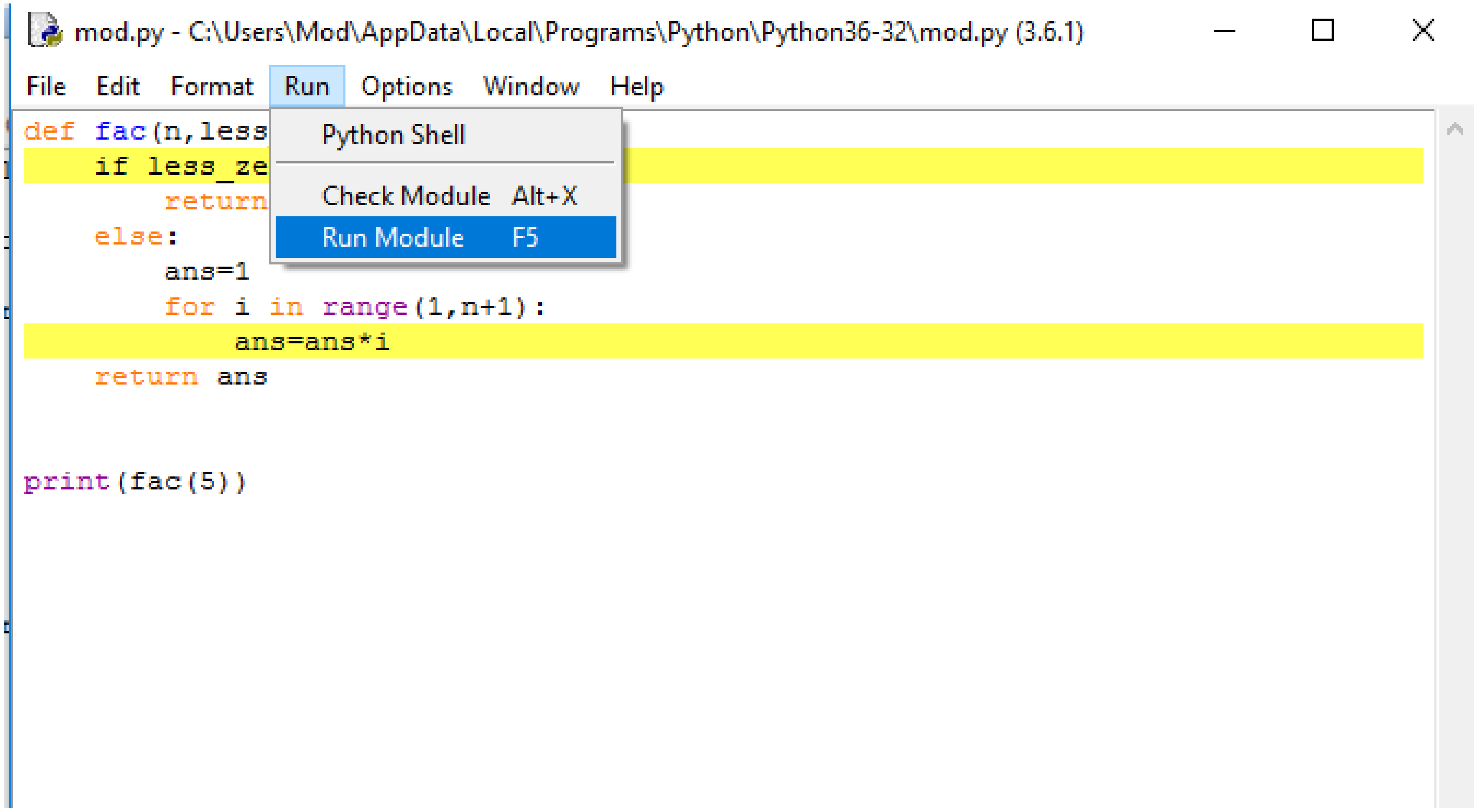
Python 3.6.1 Shell
File Edit Shell Debug
mod.py - C:\Users\Mod\AppData\Local\Programs\Python\Python36-32\mod.py (3.6.1)
File Edit Format Run Options Window Help
Cut
Copy
Paste
Set Breakpoint
Clear Breakpoint

```
Python 3.6.1 (v3.6.1
 on win32
Type "copyright", "c
>>>
=== RESTART: C:\User
120
>>>
[DEBUG ON]
>>>
[DEBUG OFF]
>>>
[DEBUG ON]
```

```
def fac(n,less_zero=False):
    if less_zero==True:
        return 0
    else:
        ans=1
        for i in range(1,n+1):
            ans=ans*i
    return ans


print(fac(5))
```

Debugging in Idle

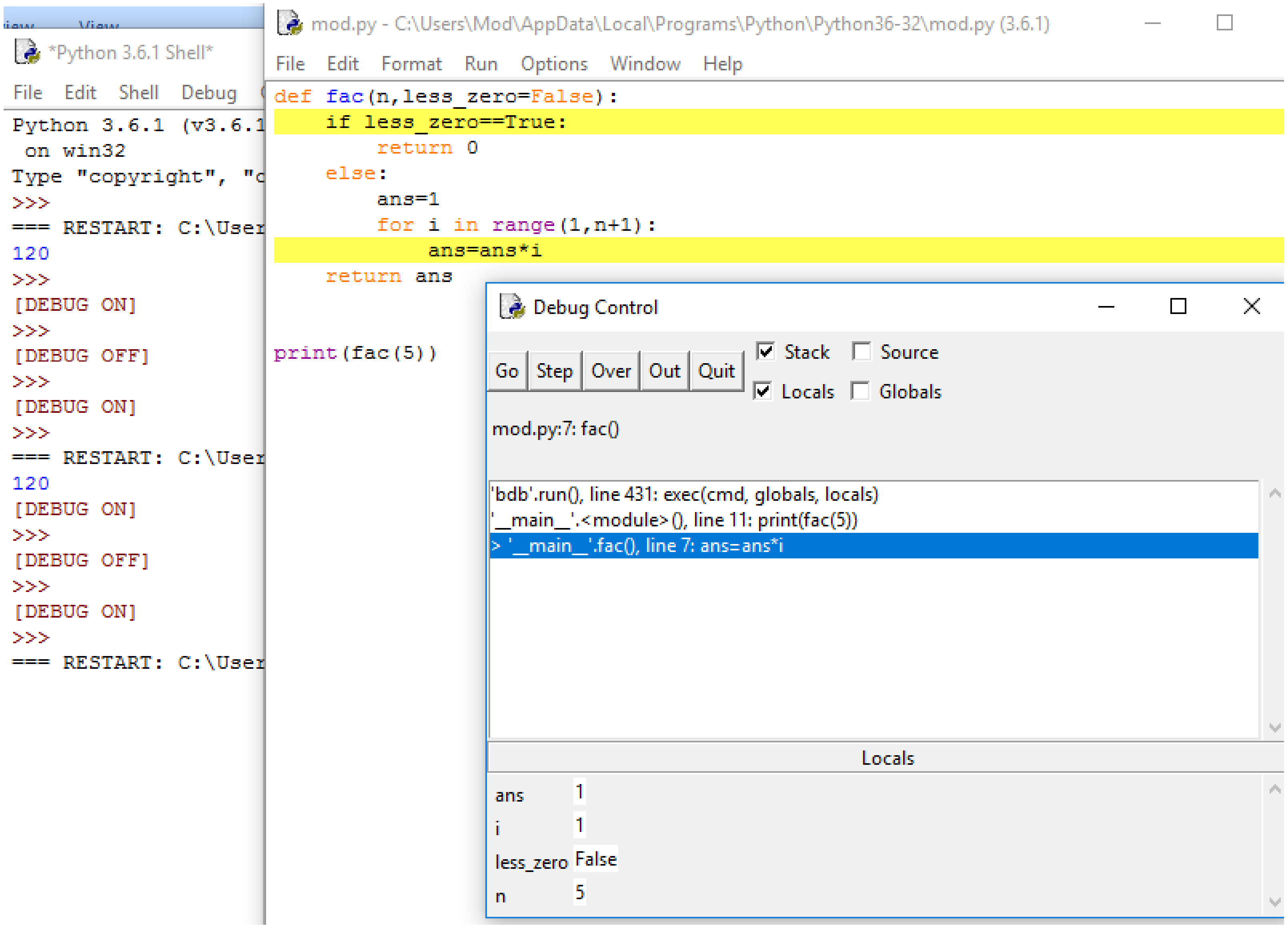


mod.py - C:\Users\Mod\AppData\Local\Programs\Python\Python36-32\mod.py (3.6.1)

```
def fac(n,less
    if less_ze
        return
    else:
        ans=1
        for i in range(1,n+1):
            ans=ans*i
    return ans


print(fac(5))
```

Debugging in Idle


Python 3.6.1 Shell
File Edit Shell Debug
mod.py - C:\Users\Mod\AppData\Local\Programs\Python\Python36-32\mod.py (3.6.1)
File Edit Format Run Options Window Help
Debug Control
Go Step Over Out Quit
Stack Source
Locals Globals
Locals

```
Python 3.6.1 (v3.6.1
 on win32
Type "copyright", "c
>>>
=== RESTART: C:\User
120
>>>
[DEBUG ON]
>>>
[DEBUG OFF]
>>>
[DEBUG ON]
>>>
=== RESTART: C:\User
120
[DEBUG ON]
>>>
[DEBUG OFF]
>>>
[DEBUG ON]
>>>
=== RESTART: C:\User
```

```python
def fac(n,less_zero=False):
    if less_zero==True:
        return 0
    else:
        ans=1
        for i in range(1,n+1):
            ans=ans*i
    return ans


print(fac(5))
```

[✓ Stack] Source
[✓ Locals] Globals

```
mod.py:7: fac()

'bdb'.run(), line 431: exec(cmd, globals, locals)
'__main__'.<module>(), line 11: print(fac(5))
> '__main__'.fac(), line 7: ans=ans*i
```

```
ans        1
i          1
less_zero  False
n          5
```