



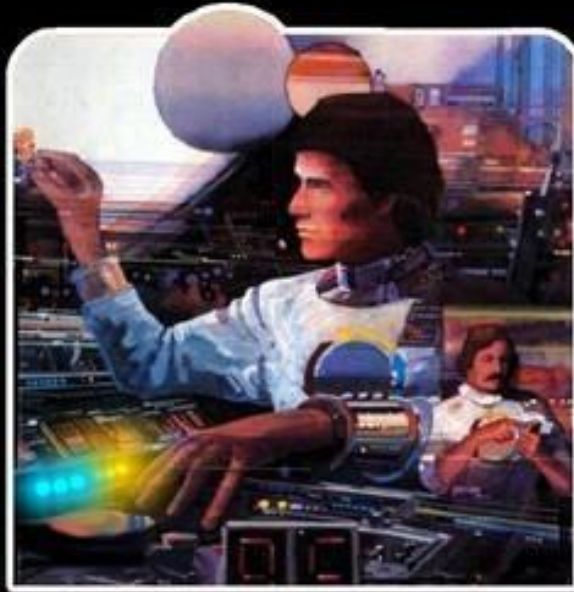
Helpful Tip



# Block Comment and Uncomment in VS Code

- Select what you want to comment and hit “Ctrl”+”K” then “Ctrl”+”C” to comment everything selected.
- “Ctrl”+”K” then “Ctrl”+”U” uncomments selected text.

# THE TWO STATES OF EVERY PROGRAMMER



**I AM A GOD.**



**I HAVE NO IDEA  
WHAT I'M DOING.**



# Documentation





**When I wrote this code,  
only God & I understood what it did.**



**Now...  
only God knows.**



# In-Line Comments

Use “#” to input lines the computer won’t read

```
# Calculate the area
length = 20
width = 40
area = length * width
statement = "The area is " + str(area) + "."
print(statement)
```



# Function (and Class) Documentation

```
def Add(number1, number2):  
    '''  
    Adds the two numbers.  
    '''  
    return number1 + number2
```





# Retrieving the Documentation

```
print(Add.__doc__)
```

Adds the two numbers.

That is two (2) “\_”s on both sides!



# Help

```
help(calculator)
```

---

Help on module calculator:

NAME

calculator

DESCRIPTION

Calculator Documentation

This class holds the calculator functions and memory variables.

-- More --



# PyDoc

## calculator

[index](#)  
[c:\users\work\dropbox \(personal\)\python class\teacher resources\lesson 4\calculator.py](c:\users\work\dropbox (personal)\python class\teacher resources\lesson 4\calculator.py)

Calculator Documentation

This class holds the calculator functions and memory variables.

### Functions

**Add**(number1, number2)

Adds the two numbers.

**ClearCurrentValue**()

Clears the current value.

**Divide**(numerator, denominator)

Divides the numerator by the denominator.

**Invert**(number)

Inverts the given number.

**MemoryClear**()

Clears the persistent memory.

**MemoryRecall**()

Retrieves the persistent memory value.

**MemoryStore**(number)

Stores the given number in the persistent memory.

**Multiply**(number1, number2)

Multiplies the two numbers.

**Power**(base, exponent)

Raises the base to the exponent.

**Subtract**(subtractFrom, subtractThis)

Subtracts subtractThis from subtractFrom.

### Data

**currentValue** = None

**memoryValue** = None



# Error Handling



## *Learning Python p.1127*

“In fact, realistic C programs often have as much code devoted to error detection as to doing actual work. But in Python, you don’t have to be so methodical (and neurotic!).”



# C++

```
try {  
    // code block to try  
}  
catch (invalid_argument& ex) {  
    cout << "An invalid-argument exception occurred: " << ex.what () << endl;  
}  
catch (...) {  
    cout << "Something besides an invalid-argument exception occurred" << endl;  
}
```



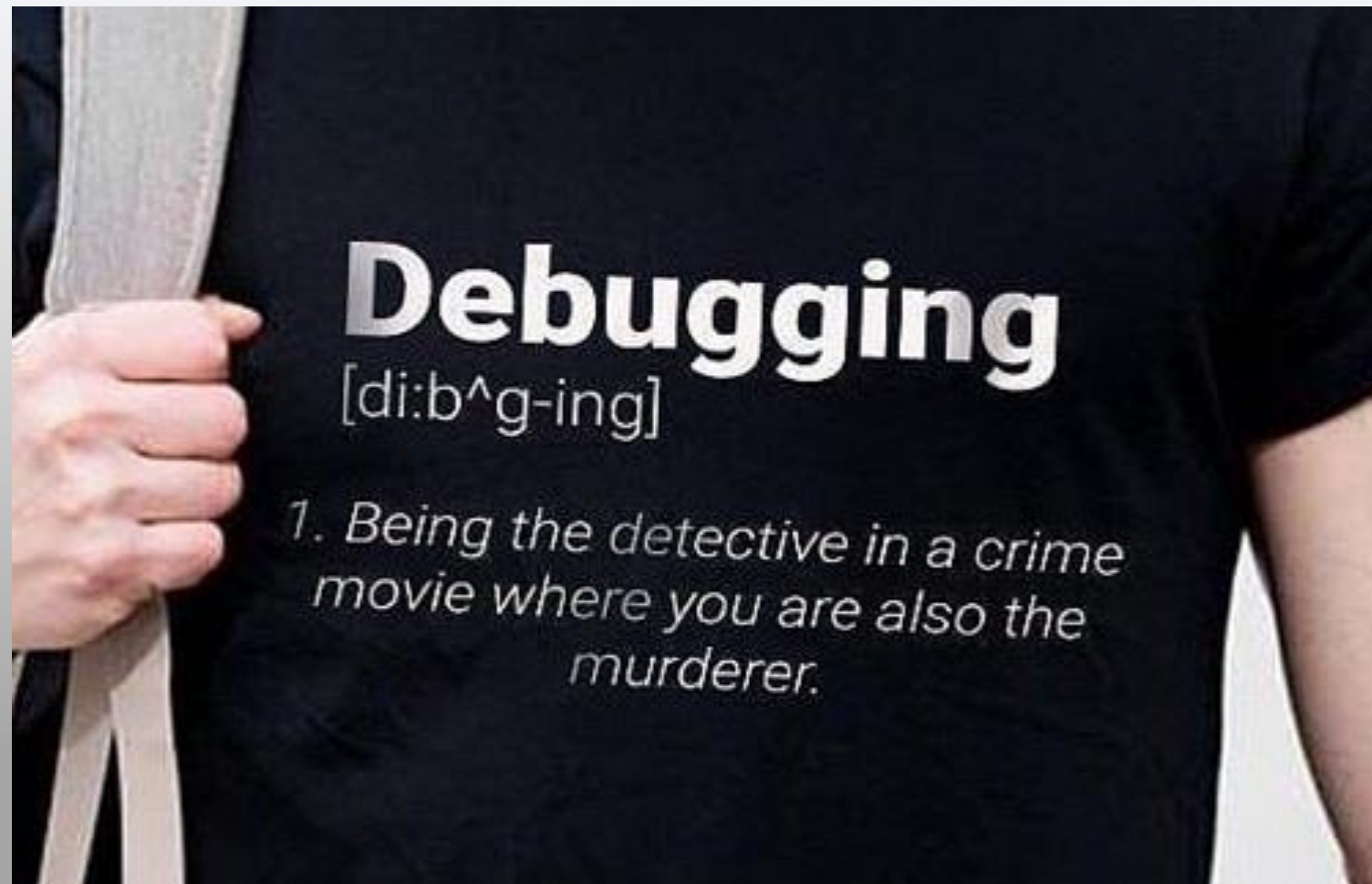
# Python

```
try:
    print(5/0)
except ArithmeticError:
    print("You had an arithmetic error.")
except:
    print("You cannot do that!")
else:
    print("No errors.")
```



# Debugging







# WHO WOULD WIN?

a computer program with  
millions of lines of code

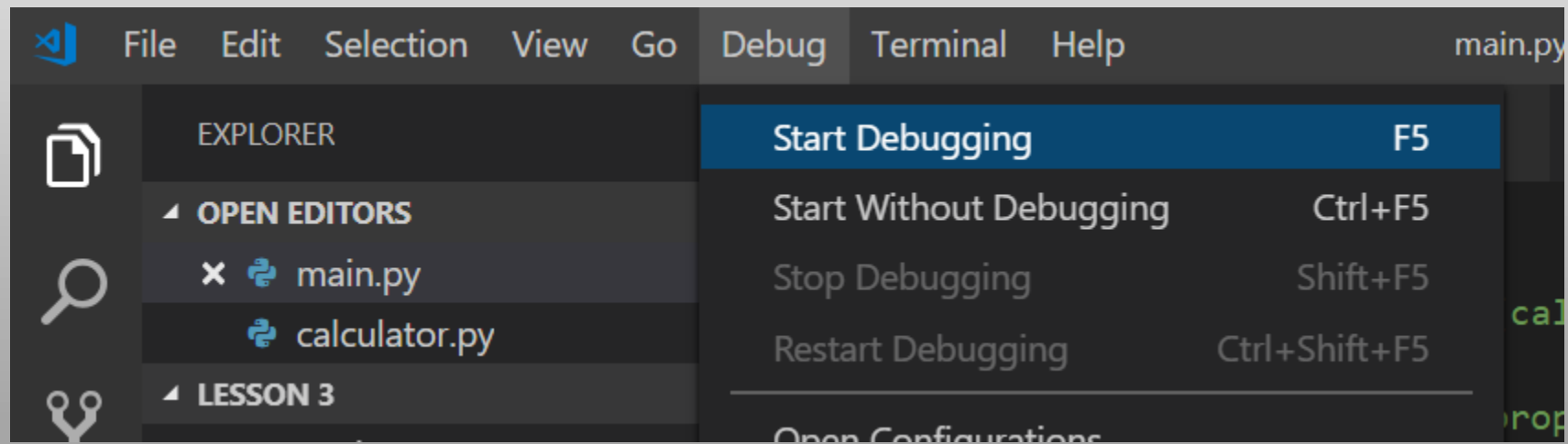


one C U R L Y B O Y  
with no friend





# Visual Studio Code Debugger





main.py - Lesson 3 - Visual Studio Code

DEBUG Python: Current File (Integrated Terminal) (Lesson 3)

main.py

**Break point**

**Current line of Code**

**Tool windows**

**VARIABLES**

- Locals
  - calculator: <module 'calculator' from 'calculator.py'>
  - firstNumber: '4'
  - operator: '\*'
  - \_\_builtins\_\_: {'ArithmeticError': <class 'ArithmeticError'>, 'AssertionError': <class 'AssertionError'>, 'AttributeError': <class 'AttributeError'>, 'BaseException': <class 'BaseException'>, 'BaseExceptionGroup': <class 'BaseExceptionGroup'>, 'BlockingIOError': <class 'BlockingIOError'>, 'BrokenPipeError': <class 'BrokenPipeError'>, 'BufferError': <class 'BufferError'>, 'BytesWarning': <class 'BytesWarning'>, 'ChildProcessError': <class 'ChildProcessError'>, 'ConnectionError': <class 'ConnectionError'>, 'ConnectionResetError': <class 'ConnectionResetError'>, 'DeprecationWarning': <class 'DeprecationWarning'>, 'EOFError': <class 'EOFError'>, 'Ellipsis': Ellipsis, 'EnvironmentError': <class 'OSError'>, 'Exception': <class 'Exception'>, 'FileNotFoundError': <class 'FileNotFoundError'>, 'FileExistsError': <class 'FileExistsError'>, 'FloatingPointError': <class 'FloatingPointError'>, 'FormatWarning': <class 'FormatWarning'>, 'FutureWarning': <class 'FutureWarning'>, 'GeneratorExit': <class 'GeneratorExit'>, 'GroupError': <class 'GroupError'>, 'ImportError': <class 'ImportError'>, 'ImportWarning': <class 'ImportWarning'>, 'IndexError': <class 'IndexError'>, 'IndentationError': <class 'IndentationError'>, 'InterruptedError': <class 'InterruptedError'>, 'IsADirectoryError': <class 'IsADirectoryError'>, 'KeyError': <class 'KeyError'>, 'KeyboardInterrupt': <class 'KeyboardInterrupt'>, 'LookupError': <class 'LookupError'>, 'MemoryError': <class 'MemoryError'>, 'ModuleNotFoundError': <class 'ModuleNotFoundError'>, 'NameError': <class 'NameError'>, 'NotADirectoryError': <class 'NotADirectoryError'>, 'NotImplementedError': <class 'NotImplementedError'>, 'OSError': <class 'OSError'>, 'OverflowError': <class 'OverflowError'>, 'PermissionError': <class 'PermissionError'>, 'ProcessLookupError': <class 'ProcessLookupError'>, 'RecursionError': <class 'RecursionError'>, 'ReferenceError': <class 'ReferenceError'>, 'RuntimeError': <class 'RuntimeError'>, 'RuntimeWarning': <class 'RuntimeWarning'>, 'StopAsyncIteration': <class 'StopAsyncIteration'>, 'StopIteration': <class 'StopIteration'>, 'SyntaxError': <class 'SyntaxError'>, 'SystemError': <class 'SystemError'>, 'SystemExit': <class 'SystemExit'>, 'TabError': <class 'TabError'>, 'TimeoutError': <class 'TimeoutError'>, 'TypeError': <class 'TypeError'>, 'UnboundLocalError': <class 'UnboundLocalError'>, 'UnicodeDecodeError': <class 'UnicodeDecodeError'>, 'UnicodeEncodeError': <class 'UnicodeEncodeError'>, 'UnicodeError': <class 'UnicodeError'>, 'UnicodeTranslateError': <class 'UnicodeTranslateError'>, 'UserWarning': <class 'UserWarning'>, 'ValueError': <class 'ValueError'>, 'Warning': <class 'Warning'>, 'ZeroDivisionError': <class 'ZeroDivisionError'>}
  - \_\_cached\_\_: None
  - \_\_doc\_\_: None

**WATCH**

**CALL STACK** PAUSED ON STEP

<module> main.py 31:1

**BREAKPOINTS**

- ☐ Raised Exceptions
- ☒ Uncaught Exceptions
- ☒ main.py 30

```
26 print("Input '\\' in the first number to use the last calculated value")
27 print("First number or help('\\H\\'): ")
28
29 firstNumber = input("First number or help('\\H\\'): ")
30 operator = input("Enter Operator: ")
31 secondNumber = input("Second number or leave blank: ")
32
33 # Use current value if first number is omitted.
34
35
36
37
38 if(operator == "+"):
39     calculator.currentValue = calculator.Add(float(firstNumber), float(secondNumber))
40
41 calculator.currentValue = calculator.Subtract(float(firstNumber), float(secondNumber))
42 print(calculator.currentValue)
43 elif(operator == "*"):
```

1: Python Debug Console

Microsoft Windows [Version 10.0.17134.590]  
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\worki\Dropbox (Personal)\Python Class\Teacher Resources\Lesson 3>cd "c:\Users\worki\Dropbox (Personal)\Python Class\Teacher Resources\Lesson 3" && cmd /C "set "PYTHONENCODING=UTF-8" && set "PYTHONUNBUFFERED=1" && C:\Users\worki\AppData\Local\Programs\Python\Python37-32\python.exe c:\Users\worki\.vscode\extensions\ms-python.python-2019.1.0\pythonFiles\ptvsd\_launcher.py --default --client --host localhost --port 57222 "c:\Users\worki\Dropbox (Personal)\Python Class\Teacher Resources\Lesson 3\main.py" "

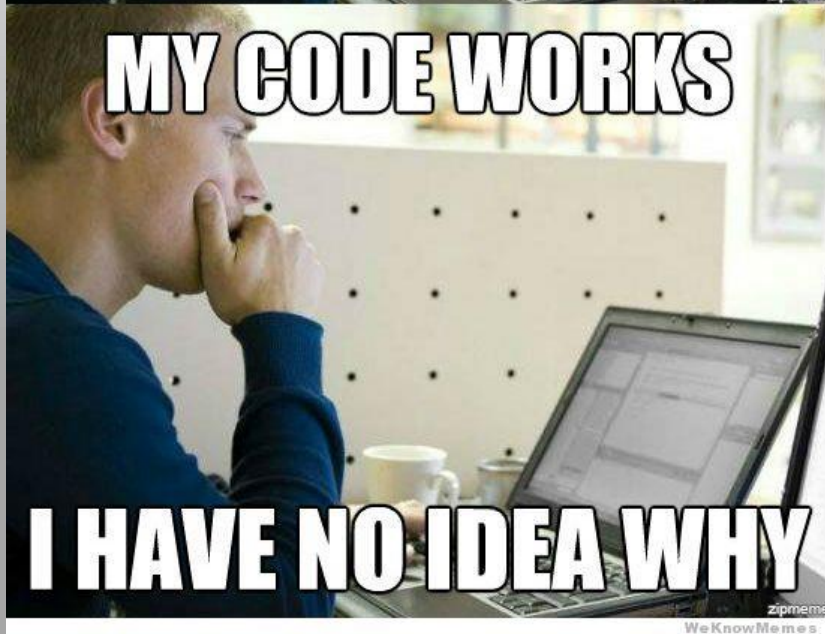
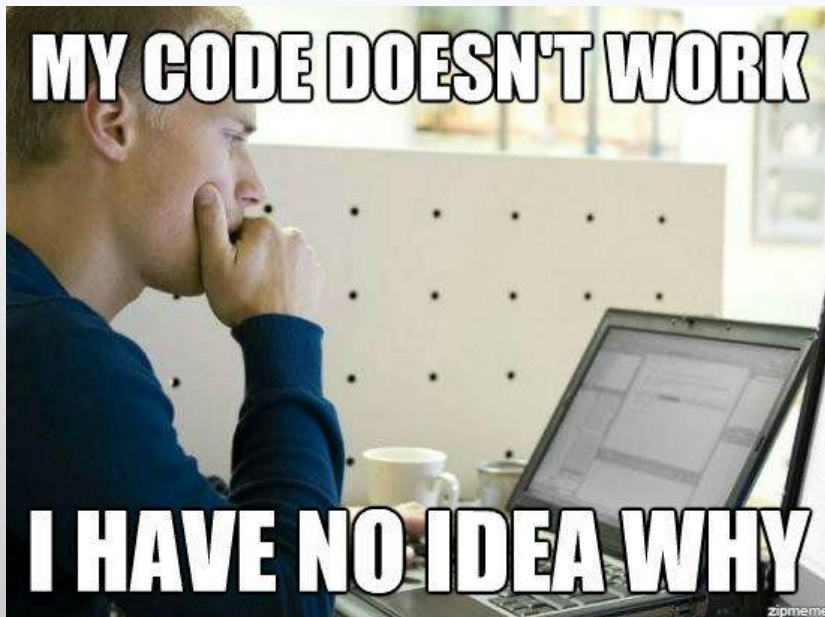
Welcome to the calculator. Input 'H' in the first number for the instructions  
First number or help('H'): 4  
Enter Operator: \*

Python 3.7.2 32-bit 0 1 Python: Current File (Integrated Terminal) (Lesson 3) Run Tests Ln 31, Col 1 Spaces: 4 UTF-8 CRLF MagicPython



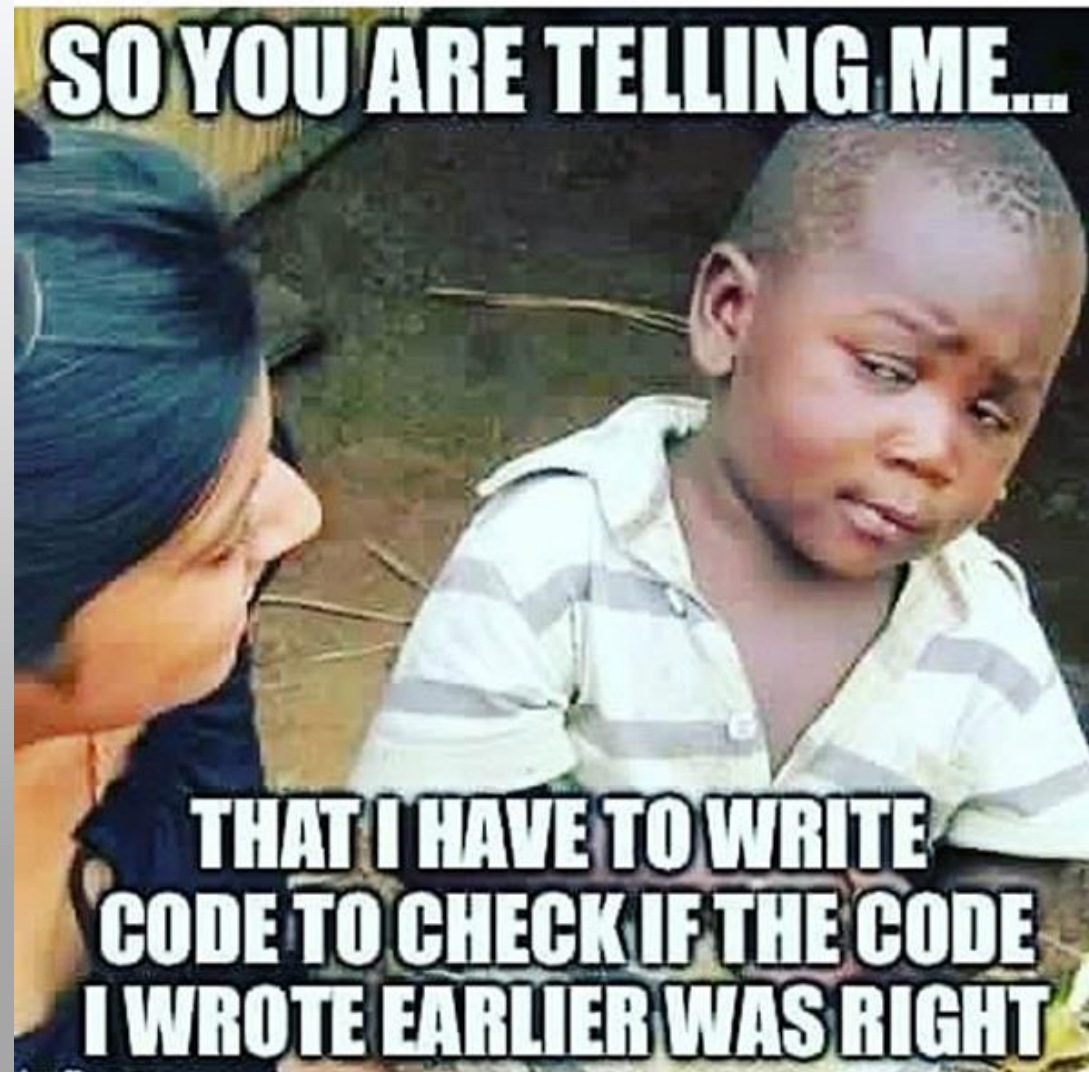
99 little bugs in the code.  
99 little bugs in the code.  
Take one down, patch it around.  
  
127 little bugs in the code...







# Unit Tests







**Bill Sempf**

@sempf

QA Engineer walks into a bar. Orders a beer. Orders 0 beers. Orders 9999999999 beers. Orders a lizard. Orders -1 beers. Orders a sfdeljknesv.



# unittest

```
import unittest
import calculator

class TestAddMethods(unittest.TestCase):

    def testPositives(self):
        self.assertEqual(calculator.Add(3, 6), 9)

    def testNegatives(self):
        self.assertEqual(calculator.Add(-3, -4), -7)

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