EXPLORER

{} User Settings | percipio66_assertions.py ×

OPEN EDITORS
- {} User Settings C:\Users...
- percipio66_assertions....

PYTHON
- writefile.txt
- Percipio_Python3-Course
  - 01_Start
  - 02_Data-Sequence Types
  - 03_Collections-Mappin...
  - 04_Modules-Functions
  - 05_Classes
  - 06_Working-with-Files
  - 07_Comprehensions
  - 08_Iterables-and-Gener...
  - 09_Exceptions
    - percipio60_catching_...
    - percipio61_catching_...
    - percipio62_the_excep...
    - percipio62b_all_pytho...
    - percipio63_exception...
    - percipio64_creating_n...
    - percipio65_traceback...
    - percipio66_assertions...
    - percipio67_chaining_...
  - 10_Automation Progra...
- Python Projects_2014
- CMD_Python_Set-Path.txt
- excel_code_.py
- excel_code_summary_mas...
- excel_code_summary_mas...
- excel_code_summary_mas...
- python_all_objects_and_s...
- Python_Clear-Window-Co...
- python_debug_logging_c...
- python_exercises_00.py
- python_exercises_01.py
- Python_Tutorial_Running-...
- Python_Tutorials.md
- Scripts - Shortcut.lnk
- start_code_for_python_m...

```
'''
percipio66_assertions.py
Percipio video: Exceptions; Assertions

* Demonstrate using assertions
* An assertion in Python is a statement that should be true and if not raises an AssertionError
    along with a 2nd parameter or argument that is a string message.
* Assertion statements work by a special value, the variable __debug__, set by the python
    interpreter with a default of True
* Floor value always rounds down
* This video goes over optimization, the python interpreter, bytecode and was way over my head.
* Python Bytecode (.pyo files) is useful because it loads faster, but does not execute any faster.
    * The first time a Python program is loaded, it produces Python bytecode which is than
        interpreted.
    * The first time a Python program is run, this code be helpful having the bytecode prepared.
    * If no longer developing the program, have the assert statements or 'if __debug__' statements
        eliminated from the code by creating Python bytecode (.pyo) files, resulting in smaller,
        more efficient code.
'''
nl = '\n'
from math import floor # import the floor module from the math module which is used in a function
    to make assertions
import py_compile # allows compiling modules not optimized to not evaluate assertions

assert 1 == 1.0, '1 = 1 & the integer does equal the float' #  1=1.0 is True so no AssertionError
    and no message printed

def round_down(num): # in this function, a number is expected to be passed returning the 'floor'
    value of the number.
    return floor(num) #

assert 6 == round_down(6.999), '6 = 6 & round_down should always round a float down' # True
    because passing 6.999 into the round_down() function performs the floor() function on that
    number which rounds down to 6, so 6=6

try:
    assert -5 == round_down(-5.999), '-5 != -6 & round_down should always round a float down' #
        False because rounding down -5.999 equals -6
except AssertionError as exception: # AssertionError is handled with an exception block or except
    statement as an object named exception
```

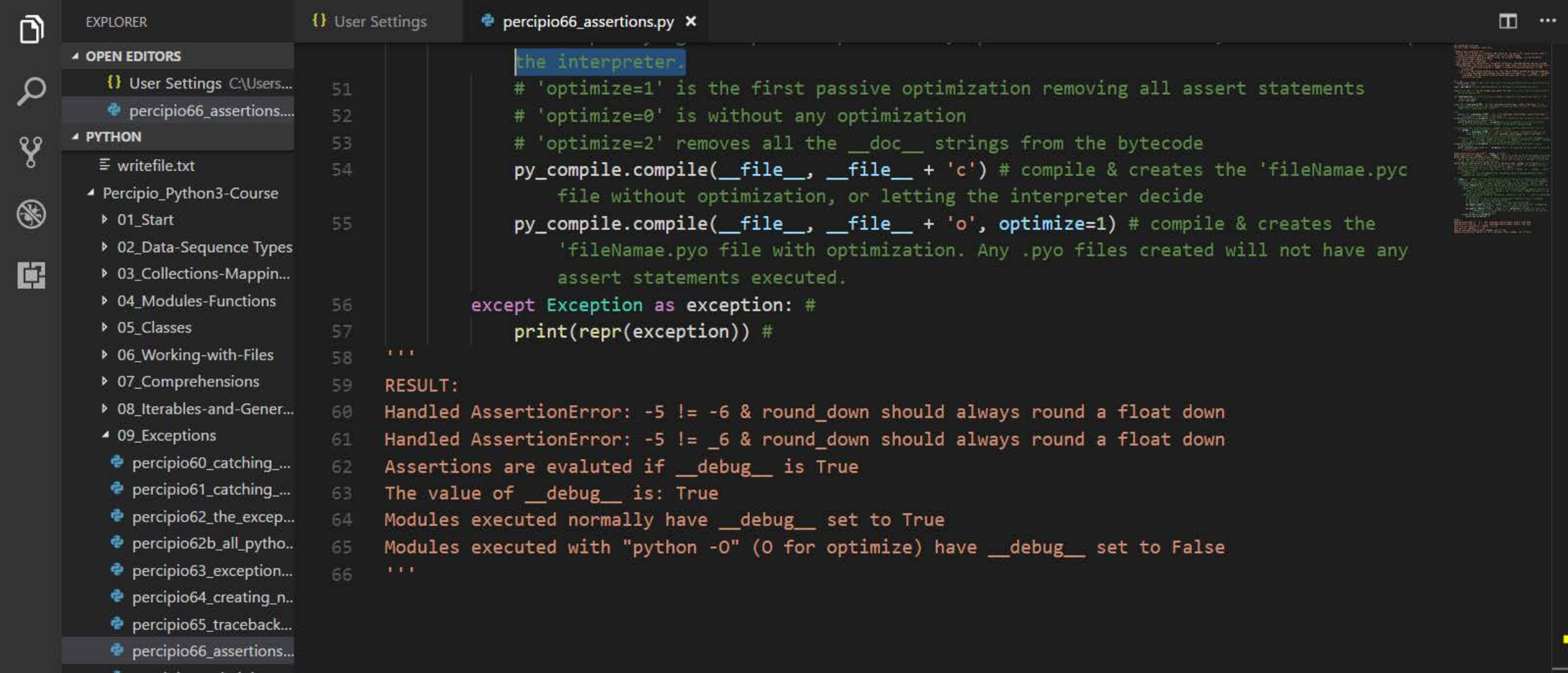
EXPLORER

{} User Settings | percipio66_assertions.py ×

OPEN EDITORS
- {} User Settings C:\Users...
- percipio66_assertions....

PYTHON
- writefile.txt
- Percipio_Python3-Course
  - 01_Start
  - 02_Data-Sequence Types
  - 03_Collections-Mappin...
  - 04_Modules-Functions
  - 05_Classes
  - 06_Working-with-Files
  - 07_Comprehensions
  - 08_Iterables-and-Gener...
  - 09_Exceptions
    - percipio60_catching_...
    - percipio61_catching_...
    - percipio62_the_excep...
    - percipio62b_all_pytho...
    - percipio63_exception...
    - percipio64_creating_n...
    - percipio65_traceback...
    - percipio66_assertions...
    - percipio67_chaining_...
  - 10_Automation Progra...
- Python Projects_2014
- CMD_Python_Set-Path.txt
- excel_code_.py
- excel_code_summary_mas...
- excel_code_summary_mas...
- excel_code_summary_mas...
- python_all_objects_and_s...
- Python_Clear-Window-Co...
- python_debug_logging_c...
- python_exercises_00.py
- python_exercises_01.py
- Python_Tutorial_Running-...
- Python_Tutorials.md
- Scripts - Shortcut.lnk
- start_code_for_python_m...

```
    statement as an object named exception
29      print('Handled AssertionError:', exception) # print the handled error with the exception
            object also printing the assert statement 2nd-paramenter passed message.
30
31  try: # with python -O optimization on, __debug__ is false, and this code block will not run
32      if __debug__: # normally defaults to True, so if True,...
33          if -5 != round_down(-5.999): # ...and if -5 does-not-equal round_down(-5.999) (or -6),...
34              raise AssertionError('-5 != _6 & round_down should always round a float down') #
                    ...and the condition is True, raise an AssertionError
35      # Essentially this previous code block constructs the equivilent of an assert statement (i.e.:
            assert -5 == round_down(-5.999)) by using an 'if __debug__ conditional statement' combined
            with a logical test that when True, raises an AssertionError
36  except AssertionError as exception: # an except block to handle this assertionError as an
        exception block
37      print ('Handled AssertionError:', exception) # print out handling the AssertionError with the
            exception object
38
39  print('Assertions are evaluted if __debug__ is True')
40  print('The value of __debug__ is:', __debug__) # when run inside the Python development environment
41  print('Modules executed normally have __debug__ set to True') # useful as True when developing a
        module. add assert statments to assert code reflecting the program is working correctly, and
        if not, the code raises an assertionError
42  print('Modules executed with "python -O" (O for optimize) have __debug__ set to False') # To run
        the python interpreter with the optimize option, from the command line, navigate into the
        directory containing the file, type 'python -O fileName'
43      # When this is done, this print code line, ''The value of __debug__ is:', __debug__', should
            change to False.
44      # for 'python -O', see (my webpage)[http://pcengineering.biz/programming/python/tables/],
            Command Line Library table
45
46  if __name__ == '__main__': # this code block only runs if this file is running in that '__main__'
        namespace,  run directly, and not started as a module.  If so, it's True that the namespace
        (represented by the special __name__ variable) is equal to that special '__main__' name.
47      if __file__.endswith('py'): # only compiles when running the source code file ending in 'py'
48          try: # potentially good be errors if the directory is not writable
49          # HELP: This is a bit confusing:  Now will try to use a py_compile.compile() function to
                compile the source code file, ending in .py, to that same fileName ending with a .pyc.
                A .pyc file is Python bytecode
50          # Without specifying the option 'optimize=1', optimize is set to '-1', which is based upon
                the interpreter.
```


EXPLORER
User Settings
percipio66_assertions.py
OPEN EDITORS
User Settings C:\Users...
percipio66_assertions....
PYTHON
writefile.txt
Percipio_Python3-Course
01_Start
02_Data-Sequence Types
03_Collections-Mappin...
04_Modules-Functions
05_Classes
06_Working-with-Files
07_Comprehensions
08_Iterables-and-Gener...
09_Exceptions
percipio60_catching_...
percipio61_catching_...
percipio62_the_excep...
percipio62b_all_pytho...
percipio63_exception...
percipio64_creating_n...
percipio65_traceback...
percipio66_assertions...
the interpreter.
'optimize=1' is the first passive optimization removing all assert statements
'optimize=0' is without any optimization
'optimize=2' removes all the __doc__ strings from the bytecode
py_compile.compile(__file__, __file__ + 'c') # compile & creates the 'fileNamae.pyc
file without optimization, or letting the interpreter decide
py_compile.compile(__file__, __file__ + 'o', optimize=1) # compile & creates the
'fileNamae.pyo file with optimization. Any .pyo files created will not have any
assert statements executed.
except Exception as exception: #
print(repr(exception)) #
'''
RESULT:
Handled AssertionError: -5 != -6 & round_down should always round a float down
Handled AssertionError: -5 != _6 & round_down should always round a float down
Assertions are evaluted if __debug__ is True
The value of __debug__ is: True
Modules executed normally have __debug__ set to True
Modules executed with "python -O" (O for optimize) have __debug__ set to False
'''