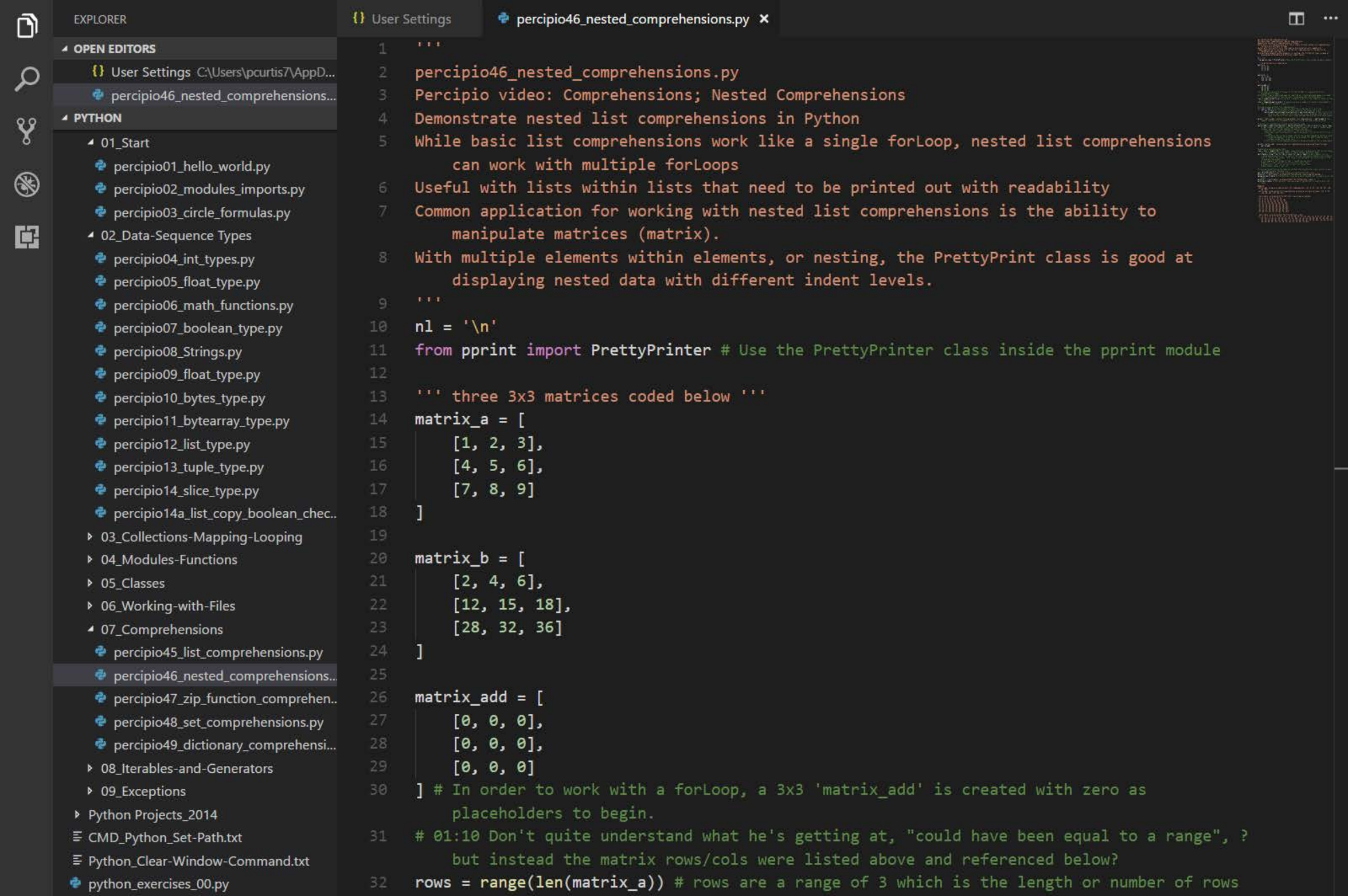
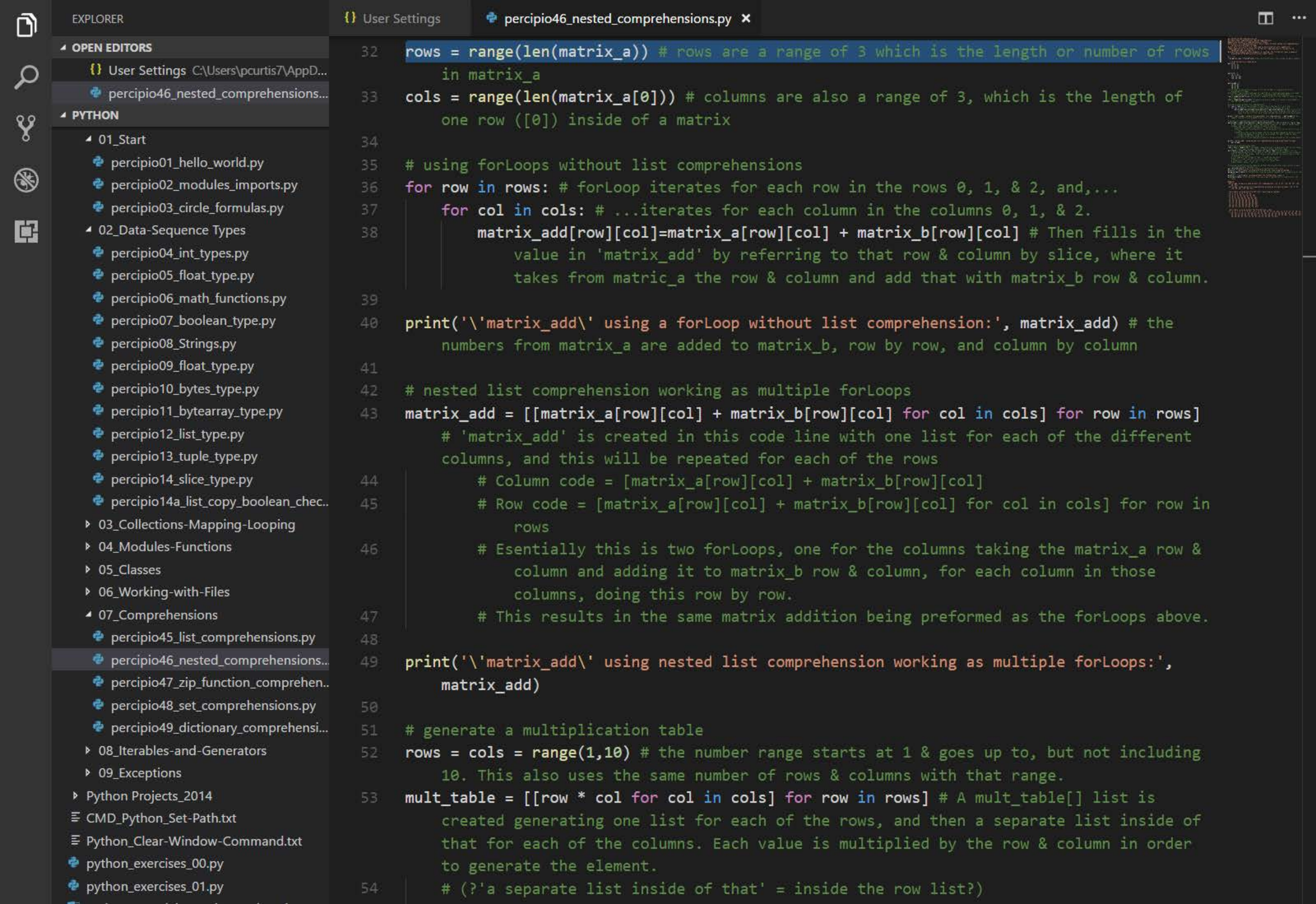
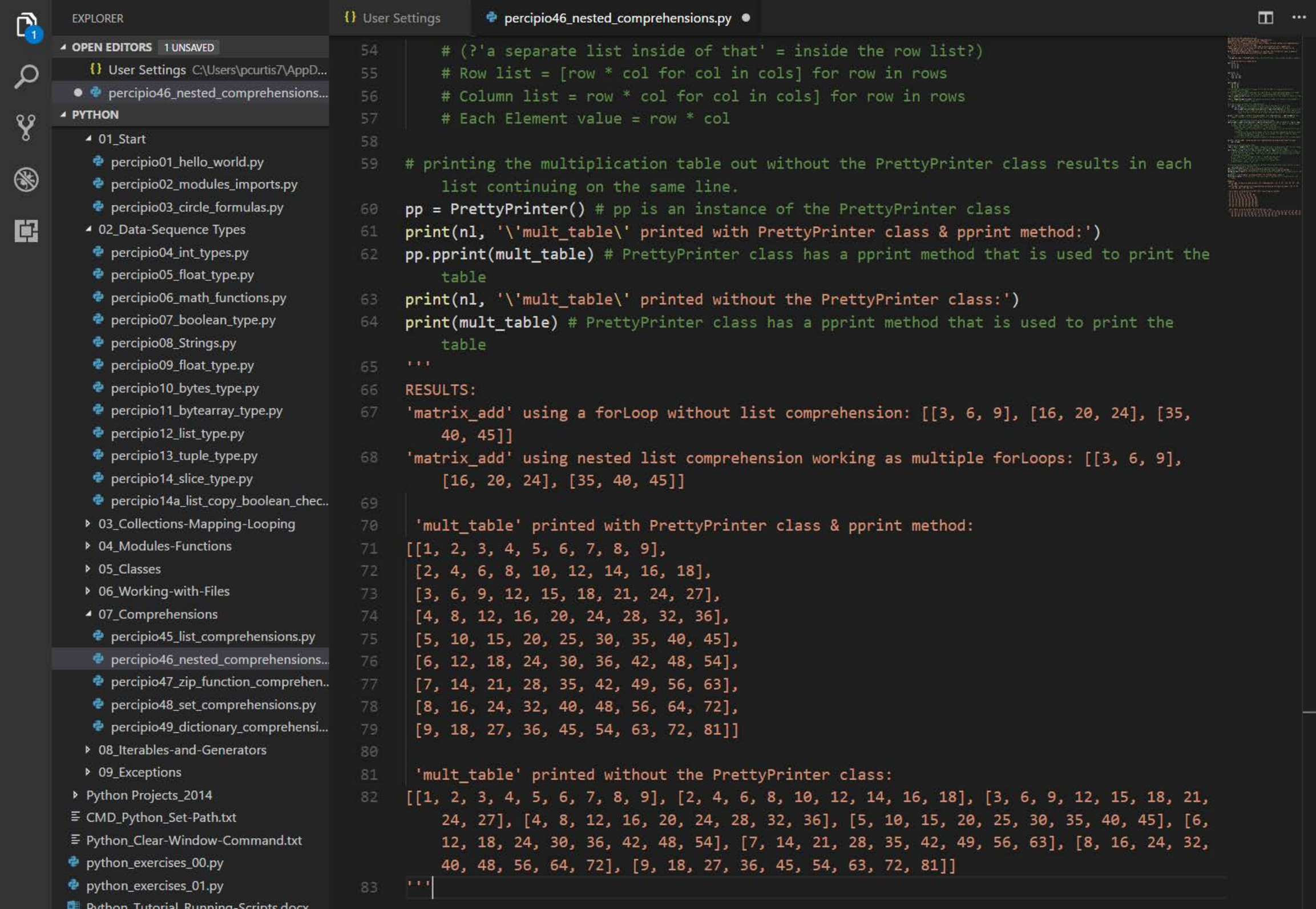

EXPLORER
User Settings
percipio46_nested_comprehensions.py
OPEN EDITORS
User Settings C:\Users\pcurtis7\AppD...
percipio46_nested_comprehensions...
PYTHON
01_Start
percipio01_hello_world.py
percipio02_modules_imports.py
percipio03_circle_formulas.py
02_Data-Sequence Types
percipio04_int_types.py
percipio05_float_type.py
percipio06_math_functions.py
percipio07_boolean_type.py
percipio08_Strings.py
percipio09_float_type.py
percipio10_bytes_type.py
percipio11_bytearray_type.py
percipio12_list_type.py
percipio13_tuple_type.py
percipio14_slice_type.py
percipio14a_list_copy_boolean_chec...
03_Collections-Mapping-Looping
04_Modules-Functions
05_Classes
06_Working-with-Files
07_Comprehensions
percipio45_list_comprehensions.py
percipio46_nested_comprehensions...
percipio47_zip_function_comprehen...
percipio48_set_comprehensions.py
percipio49_dictionary_comprehensi...
08_Iterables-and-Generators
09_Exceptions
Python Projects_2014
CMD_Python_Set-Path.txt
Python_Clear-Window-Command.txt
python_exercises_00.py
'''
percipio46_nested_comprehensions.py
Percipio video: Comprehensions; Nested Comprehensions
Demonstrate nested list comprehensions in Python
While basic list comprehensions work like a single forLoop, nested list comprehensions can work with multiple forLoops
Useful with lists within lists that need to be printed out with readability
Common application for working with nested list comprehensions is the ability to manipulate matrices (matrix).
With multiple elements within elements, or nesting, the PrettyPrint class is good at displaying nested data with different indent levels.
'''
nl = '\n'
from pprint import PrettyPrinter # Use the PrettyPrinter class inside the pprint module
''' three 3x3 matrices coded below '''
matrix_a = [
[1, 2, 3],
[4, 5, 6],
[7, 8, 9]
]
matrix_b = [
[2, 4, 6],
[12, 15, 18],
[28, 32, 36]
]
matrix_add = [
[0, 0, 0],
[0, 0, 0],
[0, 0, 0]
] # In order to work with a forLoop, a 3x3 'matrix_add' is created with zero as placeholders to begin.
# 01:10 Don't quite understand what he's getting at, "could have been equal to a range", ? but instead the matrix rows/cols were listed above and referenced below?
rows = range(len(matrix_a)) # rows are a range of 3 which is the length or number of rows


EXPLORER
User Settings
percipio46_nested_comprehensions.py
OPEN EDITORS
User Settings C:\Users\pcurtis7\AppD...
percipio46_nested_comprehensions...
PYTHON
01_Start
percipio01_hello_world.py
percipio02_modules_imports.py
percipio03_circle_formulas.py
02_Data-Sequence Types
percipio04_int_types.py
percipio05_float_type.py
percipio06_math_functions.py
percipio07_boolean_type.py
percipio08_Strings.py
percipio09_float_type.py
percipio10_bytes_type.py
percipio11_bytearray_type.py
percipio12_list_type.py
percipio13_tuple_type.py
percipio14_slice_type.py
percipio14a_list_copy_boolean_chec...
03_Collections-Mapping-Looping
04_Modules-Functions
05_Classes
06_Working-with-Files
07_Comprehensions
percipio45_list_comprehensions.py
percipio46_nested_comprehensions...
percipio47_zip_function_comprehen...
percipio48_set_comprehensions.py
percipio49_dictionary_comprehensi...
08_Iterables-and-Generators
09_Exceptions
Python Projects_2014
CMD_Python_Set-Path.txt
Python_Clear-Window-Command.txt
python_exercises_00.py
python_exercises_01.py
32 rows = range(len(matrix_a)) # rows are a range of 3 which is the length or number of rows in matrix_a
33 cols = range(len(matrix_a[0])) # columns are also a range of 3, which is the length of one row ([0]) inside of a matrix
34
35 # using forLoops without list comprehensions
36 for row in rows: # forLoop iterates for each row in the rows 0, 1, & 2, and,...
37 for col in cols: # ...iterates for each column in the columns 0, 1, & 2.
38 matrix_add[row][col]=matrix_a[row][col] + matrix_b[row][col] # Then fills in the value in 'matrix_add' by referring to that row & column by slice, where it takes from matric_a the row & column and add that with matrix_b row & column.
39
40 print('\'matrix_add\' using a forLoop without list comprehension:', matrix_add) # the numbers from matrix_a are added to matrix_b, row by row, and column by column
41
42 # nested list comprehension working as multiple forLoops
43 matrix_add = [[matrix_a[row][col] + matrix_b[row][col] for col in cols] for row in rows]
'matrix_add' is created in this code line with one list for each of the different columns, and this will be repeated for each of the rows
44 # Column code = [matrix_a[row][col] + matrix_b[row][col]
45 # Row code = [matrix_a[row][col] + matrix_b[row][col] for col in cols] for row in rows
46 # Esentially this is two forLoops, one for the columns taking the matrix_a row & column and adding it to matrix_b row & column, for each column in those columns, doing this row by row.
47 # This results in the same matrix addition being preformed as the forLoops above.
48
49 print('\'matrix_add\' using nested list comprehension working as multiple forLoops:', matrix_add)
50
51 # generate a multiplication table
52 rows = cols = range(1,10) # the number range starts at 1 & goes up to, but not including 10. This also uses the same number of rows & columns with that range.
53 mult_table = [[row * col for col in cols] for row in rows] # A mult_table[] list is created generating one list for each of the rows, and then a separate list inside of that for each of the columns. Each value is multiplied by the row & column in order to generate the element.
54 # (?'a separate list inside of that' = inside the row list?)


EXPLORER
User Settings
percipio46_nested_comprehensions.py
OPEN EDITORS 1 UNSAVED
User Settings C:\Users\pcurtis7\AppD...
percipio46_nested_comprehensions...
PYTHON
01_Start
percipio01_hello_world.py
percipio02_modules_imports.py
percipio03_circle_formulas.py
02_Data-Sequence Types
percipio04_int_types.py
percipio05_float_type.py
percipio06_math_functions.py
percipio07_boolean_type.py
percipio08_Strings.py
percipio09_float_type.py
percipio10_bytes_type.py
percipio11_bytearray_type.py
percipio12_list_type.py
percipio13_tuple_type.py
percipio14_slice_type.py
percipio14a_list_copy_boolean_chec...
03_Collections-Mapping-Looping
04_Modules-Functions
05_Classes
06_Working-with-Files
07_Comprehensions
percipio45_list_comprehensions.py
percipio46_nested_comprehensions...
percipio47_zip_function_comprehen...
percipio48_set_comprehensions.py
percipio49_dictionary_comprehensi...
08_Iterables-and-Generators
09_Exceptions
Python Projects_2014
CMD_Python_Set-Path.txt
Python_Clear-Window-Command.txt
python_exercises_00.py
python_exercises_01.py
Python_Tutorial_Running-Scripts.docx
54 # (?'a separate list inside of that' = inside the row list?)
55 # Row list = [row * col for col in cols] for row in rows
56 # Column list = row * col for col in cols] for row in rows
57 # Each Element value = row * col
58
59 # printing the multiplication table out without the PrettyPrinter class results in each list continuing on the same line.
60 pp = PrettyPrinter() # pp is an instance of the PrettyPrinter class
61 print(nl, '\'mult_table\' printed with PrettyPrinter class & pprint method:')
62 pp.pprint(mult_table) # PrettyPrinter class has a pprint method that is used to print the table
63 print(nl, '\'mult_table\' printed without the PrettyPrinter class:')
64 print(mult_table) # PrettyPrinter class has a pprint method that is used to print the table
65 '''
66 RESULTS:
67 'matrix_add' using a forLoop without list comprehension: [[3, 6, 9], [16, 20, 24], [35, 40, 45]]
68 'matrix_add' using nested list comprehension working as multiple forLoops: [[3, 6, 9], [16, 20, 24], [35, 40, 45]]
69
70 'mult_table' printed with PrettyPrinter class & pprint method:
71 [[1, 2, 3, 4, 5, 6, 7, 8, 9],
72 [2, 4, 6, 8, 10, 12, 14, 16, 18],
73 [3, 6, 9, 12, 15, 18, 21, 24, 27],
74 [4, 8, 12, 16, 20, 24, 28, 32, 36],
75 [5, 10, 15, 20, 25, 30, 35, 40, 45],
76 [6, 12, 18, 24, 30, 36, 42, 48, 54],
77 [7, 14, 21, 28, 35, 42, 49, 56, 63],
78 [8, 16, 24, 32, 40, 48, 56, 64, 72],
79 [9, 18, 27, 36, 45, 54, 63, 72, 81]]
80
81 'mult_table' printed without the PrettyPrinter class:
82 [[1, 2, 3, 4, 5, 6, 7, 8, 9], [2, 4, 6, 8, 10, 12, 14, 16, 18], [3, 6, 9, 12, 15, 18, 21, 24, 27], [4, 8, 12, 16, 20, 24, 28, 32, 36], [5, 10, 15, 20, 25, 30, 35, 40, 45], [6, 12, 18, 24, 30, 36, 42, 48, 54], [7, 14, 21, 28, 35, 42, 49, 56, 63], [8, 16, 24, 32, 40, 48, 56, 64, 72], [9, 18, 27, 36, 45, 54, 63, 72, 81]]
83 '''