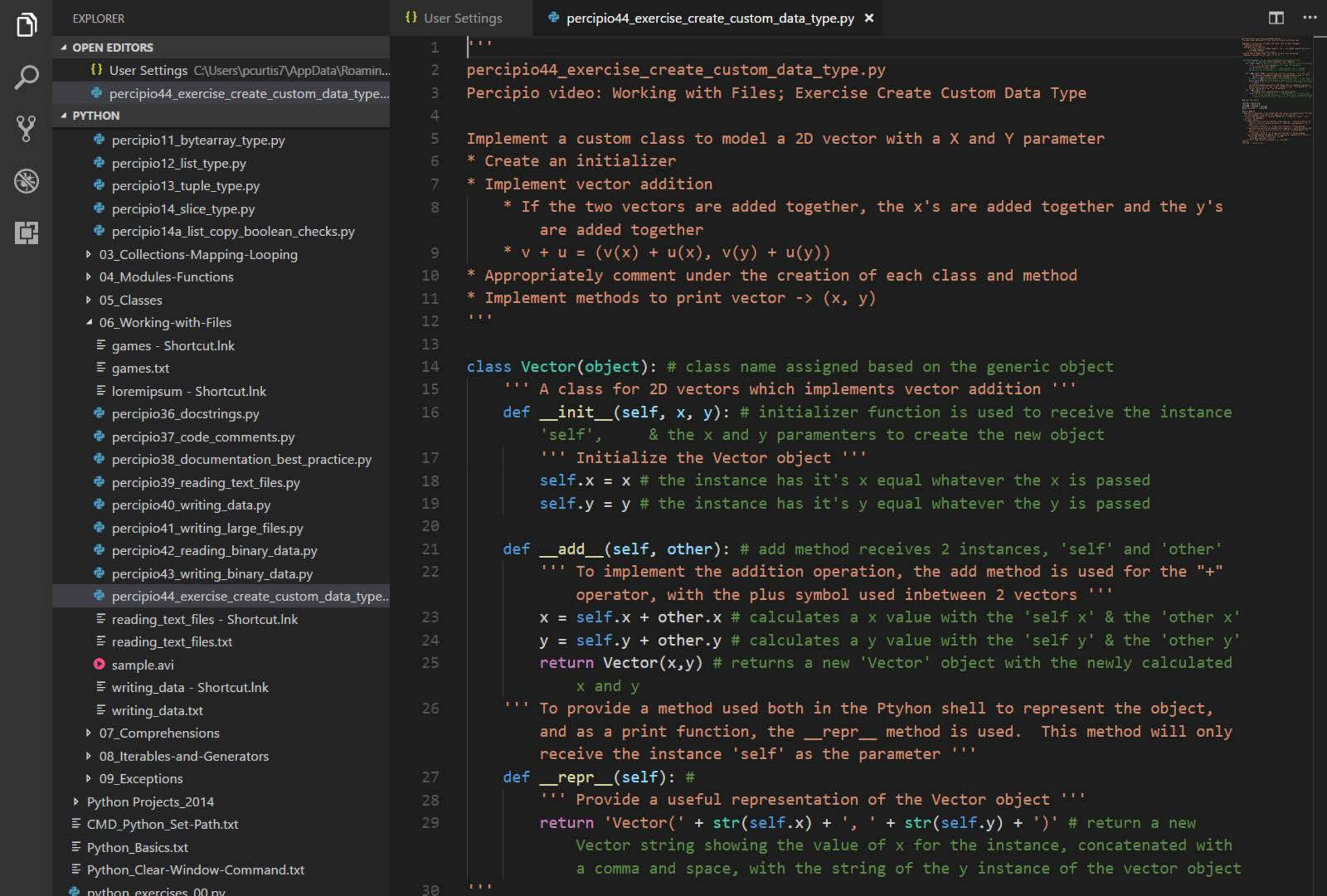
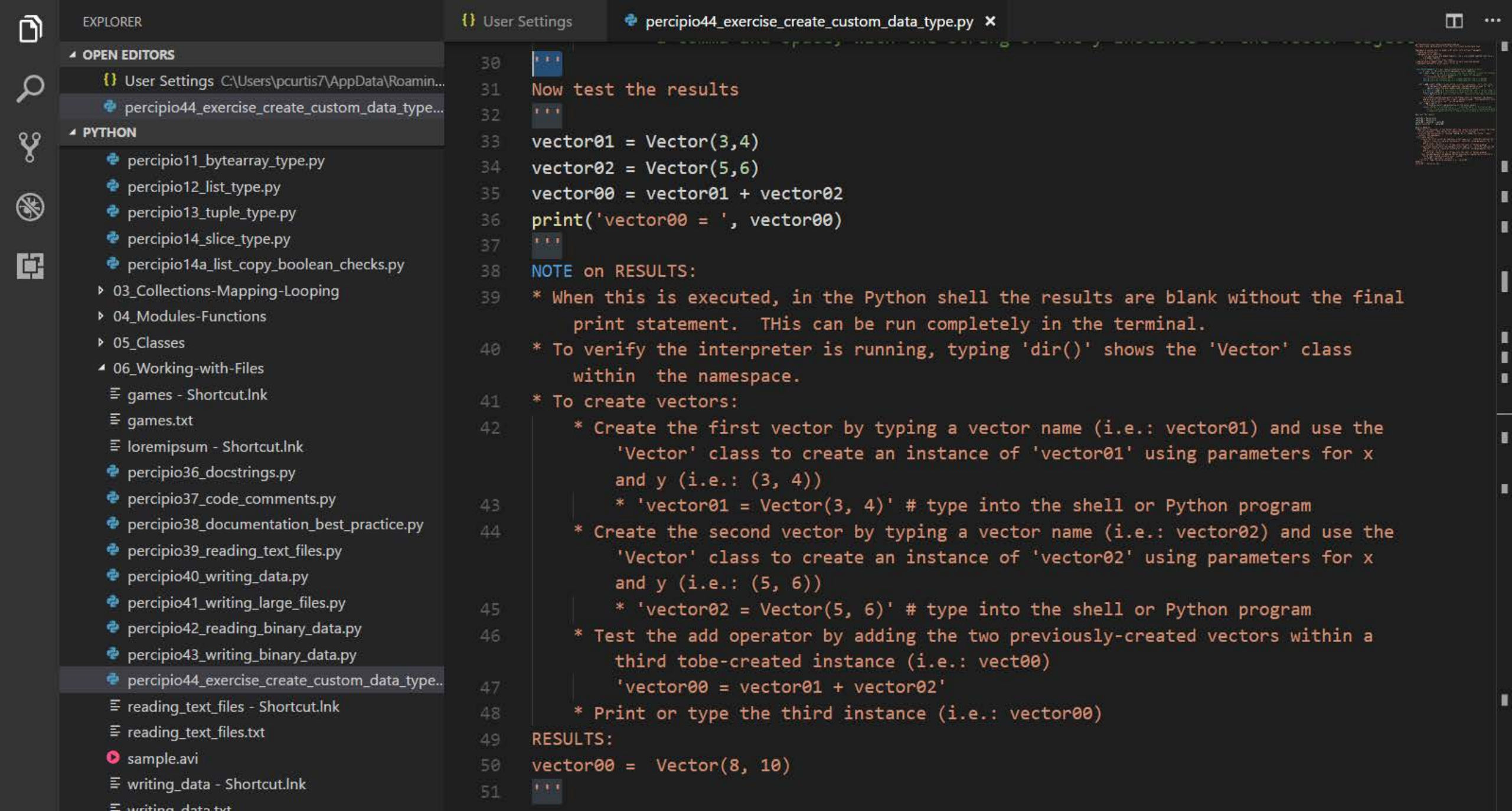

EXPLORER
OPEN EDITORS
User Settings C:\Users\pcurtis7\AppData\Roamin...
percipio44_exercise_create_custom_data_type...
PYTHON
percipio11_bytearray_type.py
percipio12_list_type.py
percipio13_tuple_type.py
percipio14_slice_type.py
percipio14a_list_copy_boolean_checks.py
03_Collections-Mapping-Looping
04_Modules-Functions
05_Classes
06_Working-with-Files
games - Shortcut.lnk
games.txt
loremipsum - Shortcut.lnk
percipio36_docstrings.py
percipio37_code_comments.py
percipio38_documentation_best_practice.py
percipio39_reading_text_files.py
percipio40_writing_data.py
percipio41_writing_large_files.py
percipio42_reading_binary_data.py
percipio43_writing_binary_data.py
percipio44_exercise_create_custom_data_type...
reading_text_files - Shortcut.lnk
reading_text_files.txt
sample.avi
writing_data - Shortcut.lnk
writing_data.txt
07_Comprehensions
08_Iterables-and-Generators
09_Exceptions
Python Projects_2014
CMD_Python_Set-Path.txt
Python_Basics.txt
Python_Clear-Window-Command.txt
python_exercises_00.py
User Settings
percipio44_exercise_create_custom_data_type.py
'''
percipio44_exercise_create_custom_data_type.py
Percipio video: Working with Files; Exercise Create Custom Data Type
Implement a custom class to model a 2D vector with a X and Y parameter
* Create an initializer
* Implement vector addition
* If the two vectors are added together, the x's are added together and the y's are added together
* v + u = (v(x) + u(x), v(y) + u(y))
* Appropriately comment under the creation of each class and method
* Implement methods to print vector -> (x, y)
'''
class Vector(object): # class name assigned based on the generic object
''' A class for 2D vectors which implements vector addition '''
def __init__(self, x, y): # initializer function is used to receive the instance 'self', & the x and y paramenters to create the new object
''' Initialize the Vector object '''
self.x = x # the instance has it's x equal whatever the x is passed
self.y = y # the instance has it's y equal whatever the y is passed
def __add__(self, other): # add method receives 2 instances, 'self' and 'other'
''' To implement the addition operation, the add method is used for the "+" operator, with the plus symbol used inbetween 2 vectors '''
x = self.x + other.x # calculates a x value with the 'self x' & the 'other x'
y = self.y + other.y # calculates a y value with the 'self y' & the 'other y'
return Vector(x,y) # returns a new 'Vector' object with the newly calculated x and y
''' To provide a method used both in the Ptyhon shell to represent the object, and as a print function, the __repr__ method is used. This method will only receive the instance 'self' as the parameter '''
def __repr__(self): #
''' Provide a useful representation of the Vector object '''
return 'Vector(' + str(self.x) + ', ' + str(self.y) + ')' # return a new Vector string showing the value of x for the instance, concatenated with a comma and space, with the string of the y instance of the vector object
'''


EXPLORER
OPEN EDITORS
User Settings C:\Users\pcurtis7\AppData\Roamin...
percipio44_exercise_create_custom_data_type...
PYTHON
percipio11_bytearray_type.py
percipio12_list_type.py
percipio13_tuple_type.py
percipio14_slice_type.py
percipio14a_list_copy_boolean_checks.py
03_Collections-Mapping-Looping
04_Modules-Functions
05_Classes
06_Working-with-Files
games - Shortcut.lnk
games.txt
loremipsum - Shortcut.lnk
percipio36_docstrings.py
percipio37_code_comments.py
percipio38_documentation_best_practice.py
percipio39_reading_text_files.py
percipio40_writing_data.py
percipio41_writing_large_files.py
percipio42_reading_binary_data.py
percipio43_writing_binary_data.py
percipio44_exercise_create_custom_data_type...
reading_text_files - Shortcut.lnk
reading_text_files.txt
sample.avi
writing_data - Shortcut.lnk
User Settings
percipio44_exercise_create_custom_data_type.py
'''
Now test the results
'''
vector01 = Vector(3,4)
vector02 = Vector(5,6)
vector00 = vector01 + vector02
print('vector00 = ', vector00)
'''
NOTE on RESULTS:
* When this is executed, in the Python shell the results are blank without the final
print statement. THis can be run completely in the terminal.
* To verify the interpreter is running, typing 'dir()' shows the 'Vector' class
within the namespace.
* To create vectors:
* Create the first vector by typing a vector name (i.e.: vector01) and use the
'Vector' class to create an instance of 'vector01' using parameters for x
and y (i.e.: (3, 4))
* 'vector01 = Vector(3, 4)' # type into the shell or Python program
* Create the second vector by typing a vector name (i.e.: vector02) and use the
'Vector' class to create an instance of 'vector02' using parameters for x
and y (i.e.: (5, 6))
* 'vector02 = Vector(5, 6)' # type into the shell or Python program
* Test the add operator by adding the two previously-created vectors within a
third tobe-created instance (i.e.: vect00)
'vector00 = vector01 + vector02'
* Print or type the third instance (i.e.: vector00)
RESULTS:
vector00 = Vector(8, 10)
'''