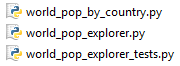
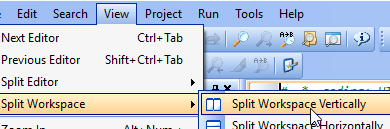
The person responsible for version control will follow the instructions in <https://github.com/viljoed/gis4x07/raw/master/GIS4107_ExerciseSetup.docx> to create a private GitHub repository called gis4107-day09 and clone this repository to acgis\gis4107\_Intro2Prog\day09\lab.

.

1. Copy the following scripts to your exercise folder:  
   
2. To facilitate your work, in PyScripter select   
     
     
     
   This will allow you to have a test module and module under test open at the same time.
3. Open world\_pop\_by\_country.py and world\_pop\_explorer.py in one workspace and the world\_pop\_explorer\_tests.py in the other.
4. The world\_pop\_by\_country.py is a script that contains population for all countries in the world for 2016 and 2017. The value of world\_pop\_by\_country.data is imported into world\_pop\_explorer.py as country\_populations. The world\_pop\_explorer.py module contains functions that process data in country\_populations. The world\_pop\_explorer\_tests.py module contains functions that test the functions in world\_pop\_explorer.py. To complete this part of the exercise, you will replace the pass statement in these modules with the appropriate code by following these steps:  
   1. Replace the pass statement in the test\_get\_country\_count() the last 3 lines of the template\_for\_test\_functions function
   2. Modify these three lines (and perhaps add others) to create a reasonable test.
   3. Run world\_pop\_explorer\_tests.py. The test should fail.
   4. Modify world\_pop\_explorer.get\_country\_count() and re-run tests until the test passes
   5. Repeat steps a-d for the other functions. Do not proceed to next function until the test for the current function passes.
5. Save exercise\_template.py as DictListUtils.py and save exercise\_template\_tests.py as DicListUntils\_tests.py. Create a function called getMissingKeys that has two Dictionary parameters: dictRef and dictToCompare where dictToCompare is the Dictionary that may have missing keys and dictRef is the Dictionary you want to compare it to. In exercise\_template\_tests.py, create a test\_getMissingKeys function and write a test with the following:  
    dictRef = {1:1, 2:2, 3:3}  
    dictToCompare = {2:2}  
    expected = [1, 3]  
   Run test\_getMissingKeys, modify getMissingKeys, and repeat until test\_getMissingKeys passes.   
   The getMissingKeys function returns a list of missing keys or an empty list if no keys are missing. Make sure you add a test for the case where there are no keys missing.   
   HINT: Use the has\_key method.
6. In DictListUtils.py, create another function called getMissingKeysWithCount that has two parameters like the previous function and will return a tuple containing the number of missing keys and the list of missing keys. This function can call getMissingKeys to get the list so you don’t duplicate its code.   
   In DictListUtils\_tests.py, create a test\_getMissingKeysWithCount function and use the same dictRef and dictToCompare as with getMissingKeys. The expected result would be (2, [1, 3]).   
   Use the same “run test, modify function under test, repeat” process as you have with all of these exercises.  
   NOTE: You can return more than one value using return with comma-separated values,   
   e.g. return value1, value2
7. In DictListUtils.py, create a function called getUnique that has one parameter named inList that may have duplicate values and returns a list of only unique values. For example, if the list is [1, 2, 2, 3], getUnique will return [1, 2, 3]. As before, make sure you create a test in DictListUtils\_tests.py that allows you to quickly assess whether or not getUnique is working or not.
8. In DictListUtils.py, create a function called flattenList that has one parameter named inList that refers to a list that may contain other lists and/or tuples. This function will return a list that contains the items of the list that are not lists or tuples as well as the items of the list(s) or tuples(s). The lists and tuples of inList will be removed. For example, if inList = [1, (2,3), [4,5]], the returned list would be   
   [1, 2, 3, 4, 5]  
   HINT: You can test for a list using code like   
   if type(someList) == list or   
   if type(someTuple) == tuple