The person responsible for the repository will:

Create a GitHub repo for these exercises, i.e.   
[https://github.com/<GitHub\_username>/gis4107-day09.git](https://github.com/%3cGitHub_username%3e/gis4107-day09.git)

Clone it as described by GIS4x07GitWorkflow.docx in <https://github.com/viljoed/gis4x07>

In the local repository, E:\acgis\gis4107\day09\lab, create the folder named with you and your partner’s name as provided by the instructor.

1. Copy the ListTupleDict.py file into you exercise folder. This script contains stubs (header and body with pass keyword) for the functions and skeleton test functions for each. Complete the functions and tests. Follow these steps for each function (i.e. one function at a time!):  
   1. Run the code
   2. Given the values under the Tests setup comment, set a value for expected in the test function
   3. Run the code to make sure if fails as expected
   4. Edit the function you are working on and run the code until its associated test passes
   5. Move on to the next function

The following functions will return a day name associated with dayNumber **(1 = Monday)** from the List, Tuple, or Dict

* + getDayNameFromList(dayNumber)
  + getDayNameFromTuple(dayNumber)
  + getDayNameFromDict(dayNumber)

The following functions will return a list of day names between and including firstDayNumber and lastDayNumber

* + getDayNamesFromList(firstDayNumber, lastDayNumber)
  + getDayNamesFromTuple(firstDayNumber, lastDayNumber)
  + getDayNamesFromDict(firstDayNumber, lastDayNumber)

Things to think about:

* Which type(s) is (are) better for getting a specific day names by number?
* Which type(s) is (are) better for getting a range of day names given first and last day numbers?

1. Create a DictListUtils.py module by making a copy of ExerciseTemplate.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. This function returns a list of missing keys or an empty list if no keys are missing. Be sure to create a test\_getMissingKeys early in the process so you can test your progress.   
   e.g. If dictRef = {1:1, 2:2, 3:3} and dictToCompare = {2:2} then   
   getMissingKeys(dictRef, dictToCompare) returns [1, 3]  
     
   HINT: Use the has\_key method.
2. 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. An example output would be (2, [1, 3]). Again, be sure to create a test for this function to help develop this function.  
   NOTE: You can return more than one value using return with comma-separated values,   
   e.g. return value1, value2
3. 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 that allows you to quickly assess whether or not getUnique is working or not.
4. 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