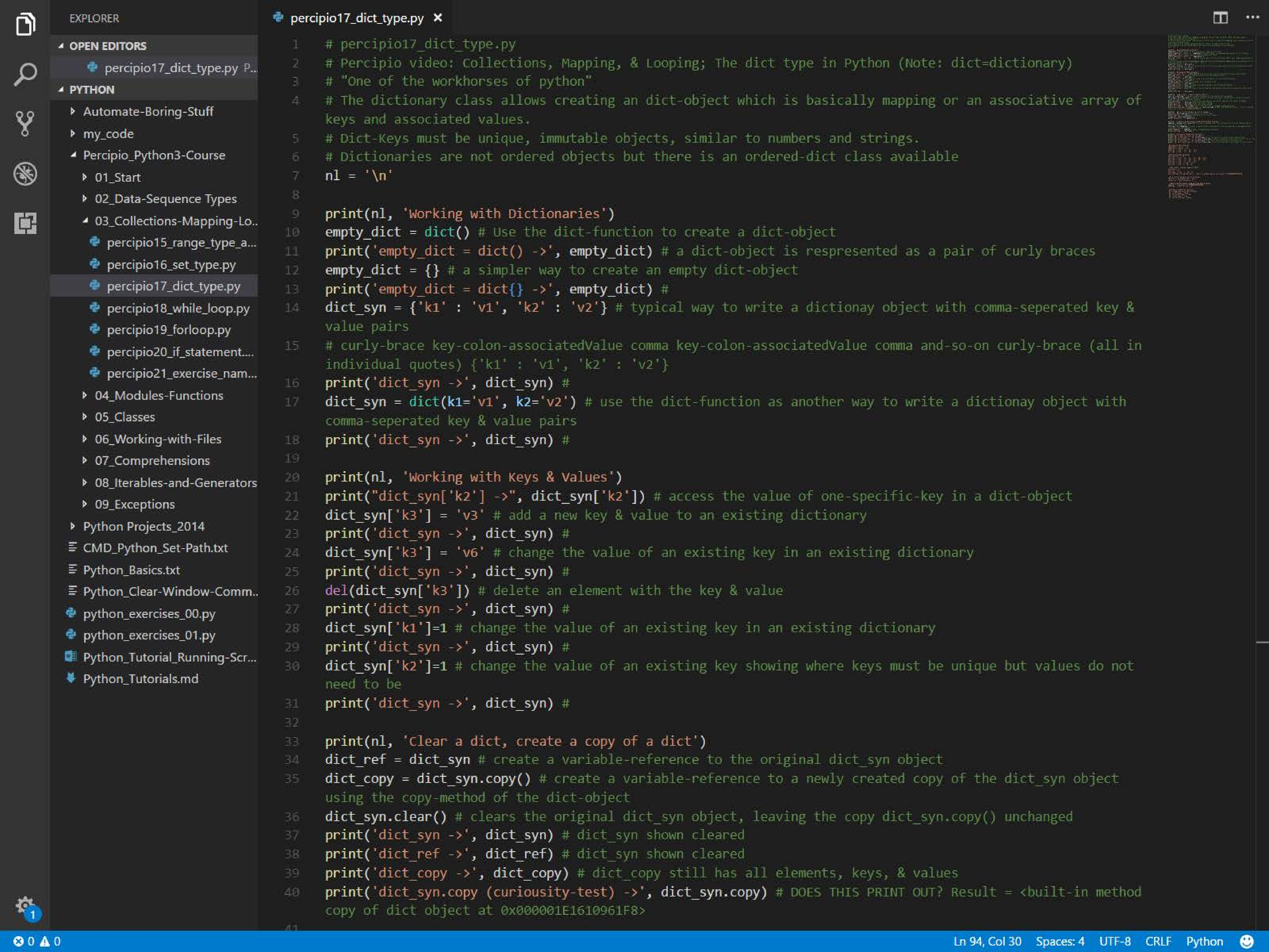
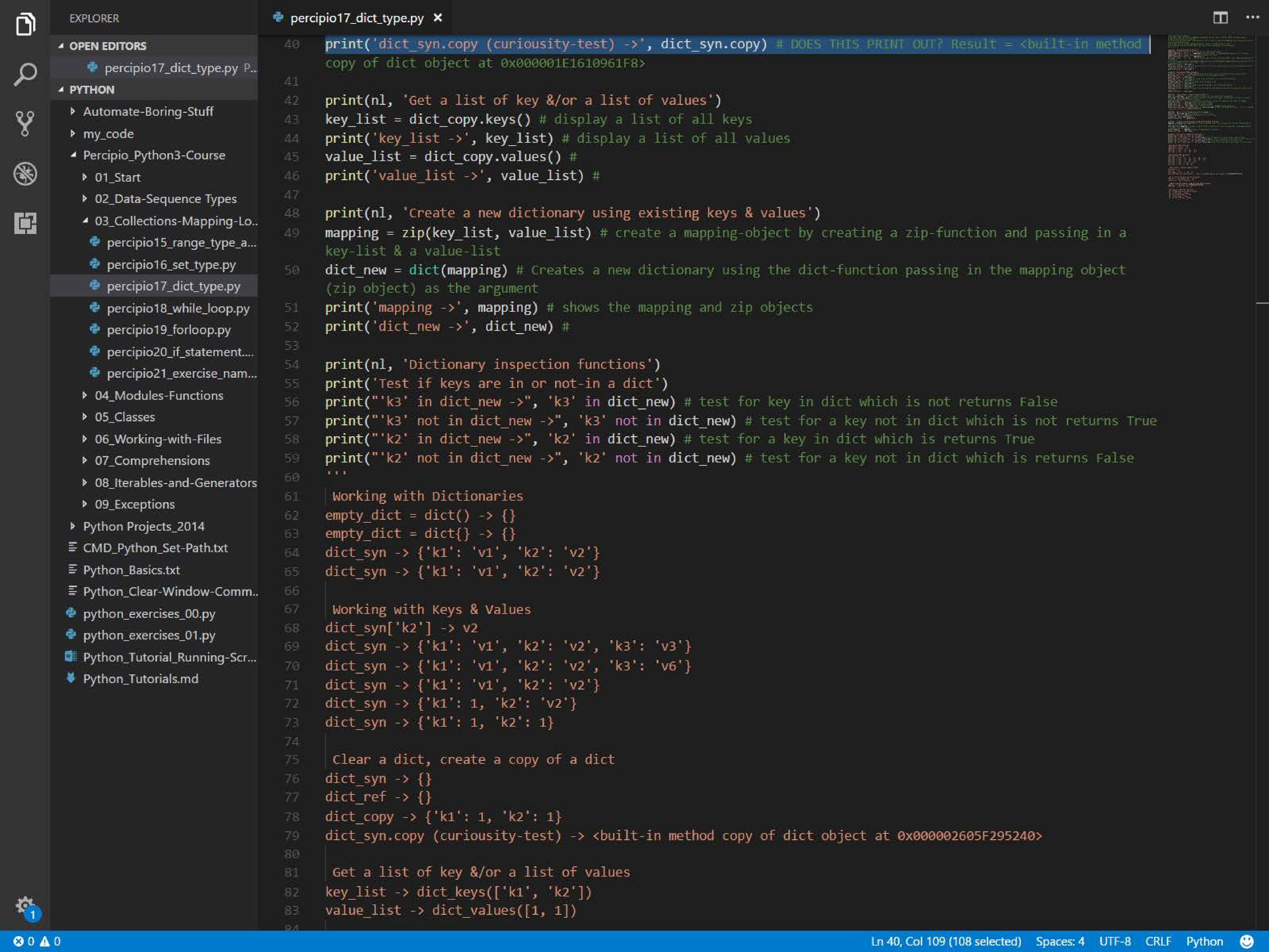
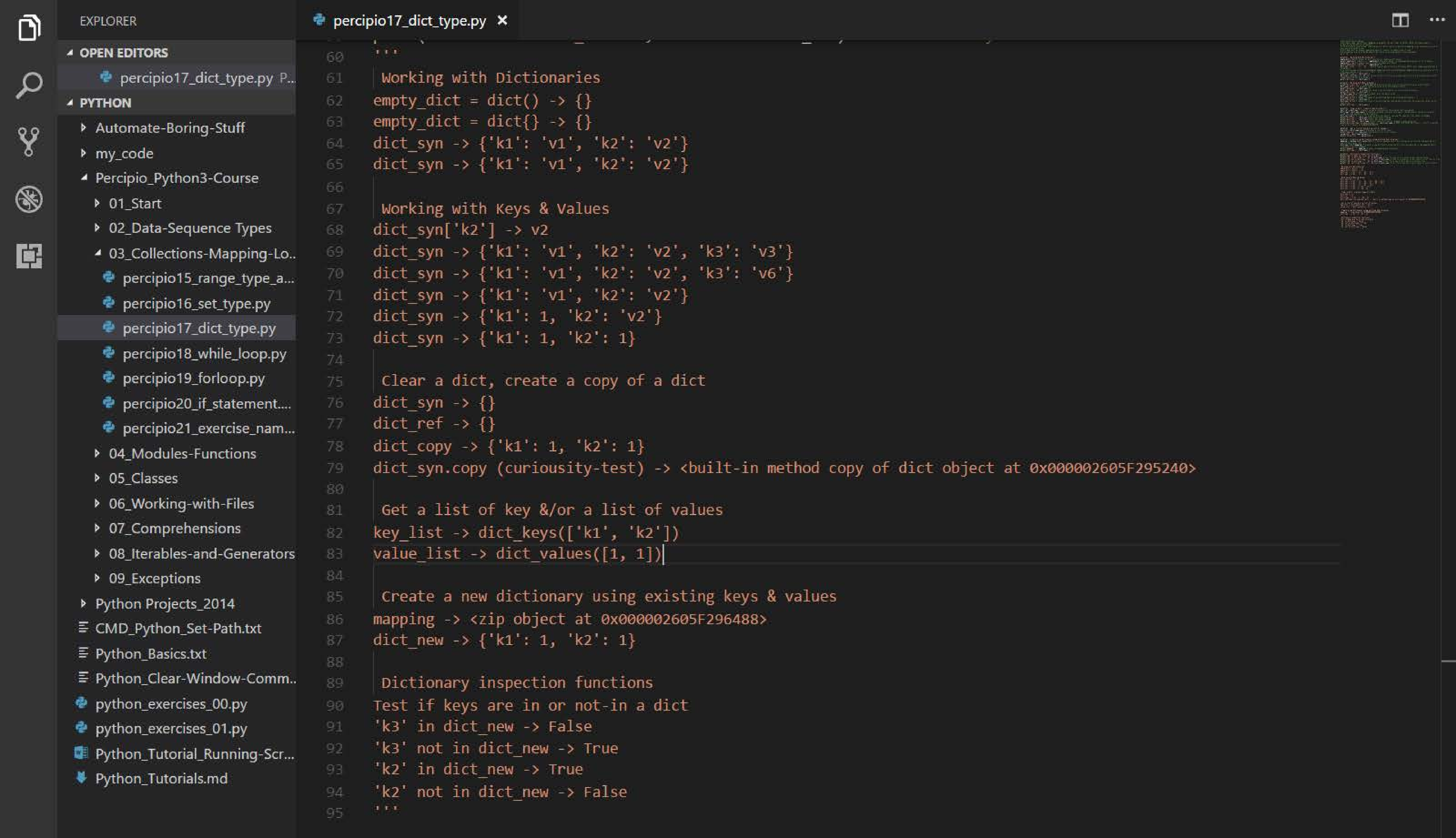


```python
# percipio17_dict_type.py
# Percipio video: Collections, Mapping, & Looping; The dict type in Python (Note: dict=dictionary)
# "One of the workhorses of python"
# The dictionary class allows creating an dict-object which is basically mapping or an associative array of keys and associated values.
# Dict-Keys must be unique, immutable objects, similar to numbers and strings.
# Dictionaries are not ordered objects but there is an ordered-dict class available
nl = '\n'

print(nl, 'Working with Dictionaries')
empty_dict = dict() # Use the dict-function to create a dict-object
print('empty_dict = dict() ->', empty_dict) # a dict-object is respresented as a pair of curly braces
empty_dict = {} # a simpler way to create an empty dict-object
print('empty_dict = dict{} ->', empty_dict) #
dict_syn = {'k1' : 'v1', 'k2' : 'v2'} # typical way to write a dictionay object with comma-seperated key & value pairs
# curly-brace key-colon-associatedValue comma key-colon-associatedValue comma and-so-on curly-brace (all in individual quotes) {'k1' : 'v1', 'k2' : 'v2'}
print('dict_syn ->', dict_syn) #
dict_syn = dict(k1='v1', k2='v2') # use the dict-function as another way to write a dictionay object with comma-seperated key & value pairs
print('dict_syn ->', dict_syn) #

print(nl, 'Working with Keys & Values')
print("dict_syn['k2'] ->", dict_syn['k2']) # access the value of one-specific-key in a dict-object
dict_syn['k3'] = 'v3' # add a new key & value to an existing dictionary
print('dict_syn ->', dict_syn) #
dict_syn['k3'] = 'v6' # change the value of an existing key in an existing dictionary
print('dict_syn ->', dict_syn) #
del(dict_syn['k3']) # delete an element with the key & value
print('dict_syn ->', dict_syn) #
dict_syn['k1']=1 # change the value of an existing key in an existing dictionary
print('dict_syn ->', dict_syn) #
dict_syn['k2']=1 # change the value of an existing key showing where keys must be unique but values do not need to be
print('dict_syn ->', dict_syn) #

print(nl, 'Clear a dict, create a copy of a dict')
dict_ref = dict_syn # create a variable-reference to the original dict_syn object
dict_copy = dict_syn.copy() # create a variable-reference to a newly created copy of the dict_syn object using the copy-method of the dict-object
dict_syn.clear() # clears the original dict_syn object, leaving the copy dict_syn.copy() unchanged
print('dict_syn ->', dict_syn) # dict_syn shown cleared
print('dict_ref ->', dict_ref) # dict_syn shown cleared
print('dict_copy ->', dict_copy) # dict_copy still has all elements, keys, & values
print('dict_syn.copy (curiousity-test) ->', dict_syn.copy) # DOES THIS PRINT OUT? Result = <built-in method copy of dict object at 0x000001E1610961F8>
```



```
print('dict_syn.copy (curiousity-test) ->', dict_syn.copy) # DOES THIS PRINT OUT? Result = <built-in method copy of dict object at 0x000001E1610961F8>

print(nl, 'Get a list of key &/or a list of values')
key_list = dict_copy.keys() # display a list of all keys
print('key_list ->', key_list) # display a list of all values
value_list = dict_copy.values() #
print('value_list ->', value_list) #

print(nl, 'Create a new dictionary using existing keys & values')
mapping = zip(key_list, value_list) # create a mapping-object by creating a zip-function and passing in a key-list & a value-list
dict_new = dict(mapping) # Creates a new dictionary using the dict-function passing in the mapping object (zip object) as the argument
print('mapping ->', mapping) # shows the mapping and zip objects
print('dict_new ->', dict_new) #

print(nl, 'Dictionary inspection functions')
print('Test if keys are in or not-in a dict')
print("'k3' in dict_new ->", 'k3' in dict_new) # test for key in dict which is not returns False
print("'k3' not in dict_new ->", 'k3' not in dict_new) # test for a key not in dict which is not returns True
print("'k2' in dict_new ->", 'k2' in dict_new) # test for a key in dict which is returns True
print("'k2' not in dict_new ->", 'k2' not in dict_new) # test for a key not in dict which is returns False
'''
 Working with Dictionaries
empty_dict = dict() -> {}
empty_dict = dict{} -> {}
dict_syn -> {'k1': 'v1', 'k2': 'v2'}
dict_syn -> {'k1': 'v1', 'k2': 'v2'}

 Working with Keys & Values
dict_syn['k2'] -> v2
dict_syn -> {'k1': 'v1', 'k2': 'v2', 'k3': 'v3'}
dict_syn -> {'k1': 'v1', 'k2': 'v2', 'k3': 'v6'}
dict_syn -> {'k1': 'v1', 'k2': 'v2'}
dict_syn -> {'k1': 1, 'k2': 'v2'}
dict_syn -> {'k1': 1, 'k2': 1}

 Clear a dict, create a copy of a dict
dict_syn -> {}
dict_ref -> {}
dict_copy -> {'k1': 1, 'k2': 1}
dict_syn.copy (curiousity-test) -> <built-in method copy of dict object at 0x000002605F295240>

 Get a list of key &/or a list of values
key_list -> dict_keys(['k1', 'k2'])
value_list -> dict_values([1, 1])
```



```
'''
 Working with Dictionaries
empty_dict = dict() -> {}
empty_dict = dict{} -> {}
dict_syn -> {'k1': 'v1', 'k2': 'v2'}
dict_syn -> {'k1': 'v1', 'k2': 'v2'}

 Working with Keys & Values
dict_syn['k2'] -> v2
dict_syn -> {'k1': 'v1', 'k2': 'v2', 'k3': 'v3'}
dict_syn -> {'k1': 'v1', 'k2': 'v2', 'k3': 'v6'}
dict_syn -> {'k1': 'v1', 'k2': 'v2'}
dict_syn -> {'k1': 1, 'k2': 'v2'}
dict_syn -> {'k1': 1, 'k2': 1}

 Clear a dict, create a copy of a dict
dict_syn -> {}
dict_ref -> {}
dict_copy -> {'k1': 1, 'k2': 1}
dict_syn.copy (curiousity-test) -> <built-in method copy of dict object at 0x000002605F295240>

 Get a list of key &/or a list of values
key_list -> dict_keys(['k1', 'k2'])
value_list -> dict_values([1, 1])

 Create a new dictionary using existing keys & values
mapping -> <zip object at 0x000002605F296488>
dict_new -> {'k1': 1, 'k2': 1}

 Dictionary inspection functions
Test if keys are in or not-in a dict
'k3' in dict_new -> False
'k3' not in dict_new -> True
'k2' in dict_new -> True
'k2' not in dict_new -> False
'''
```