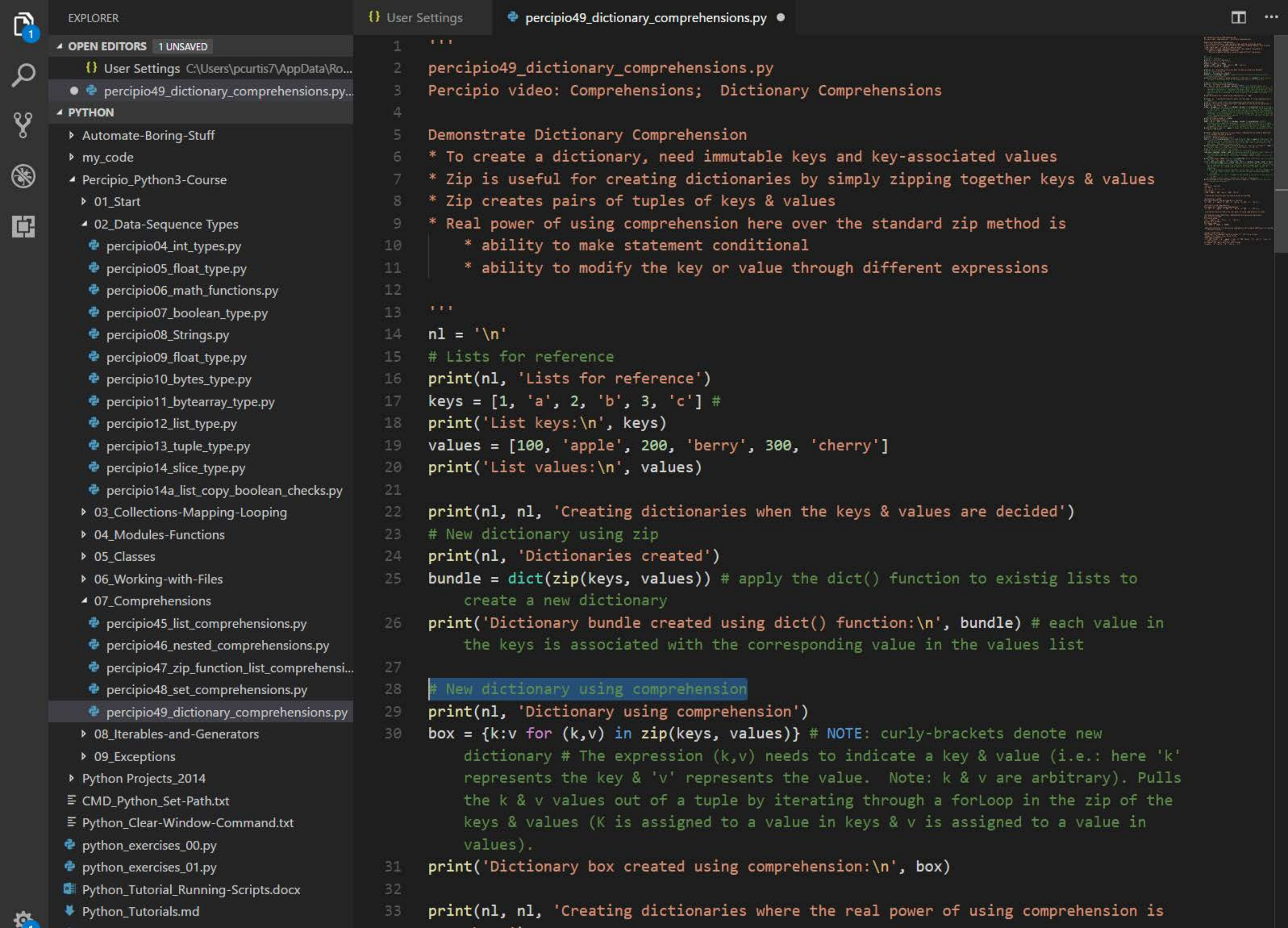
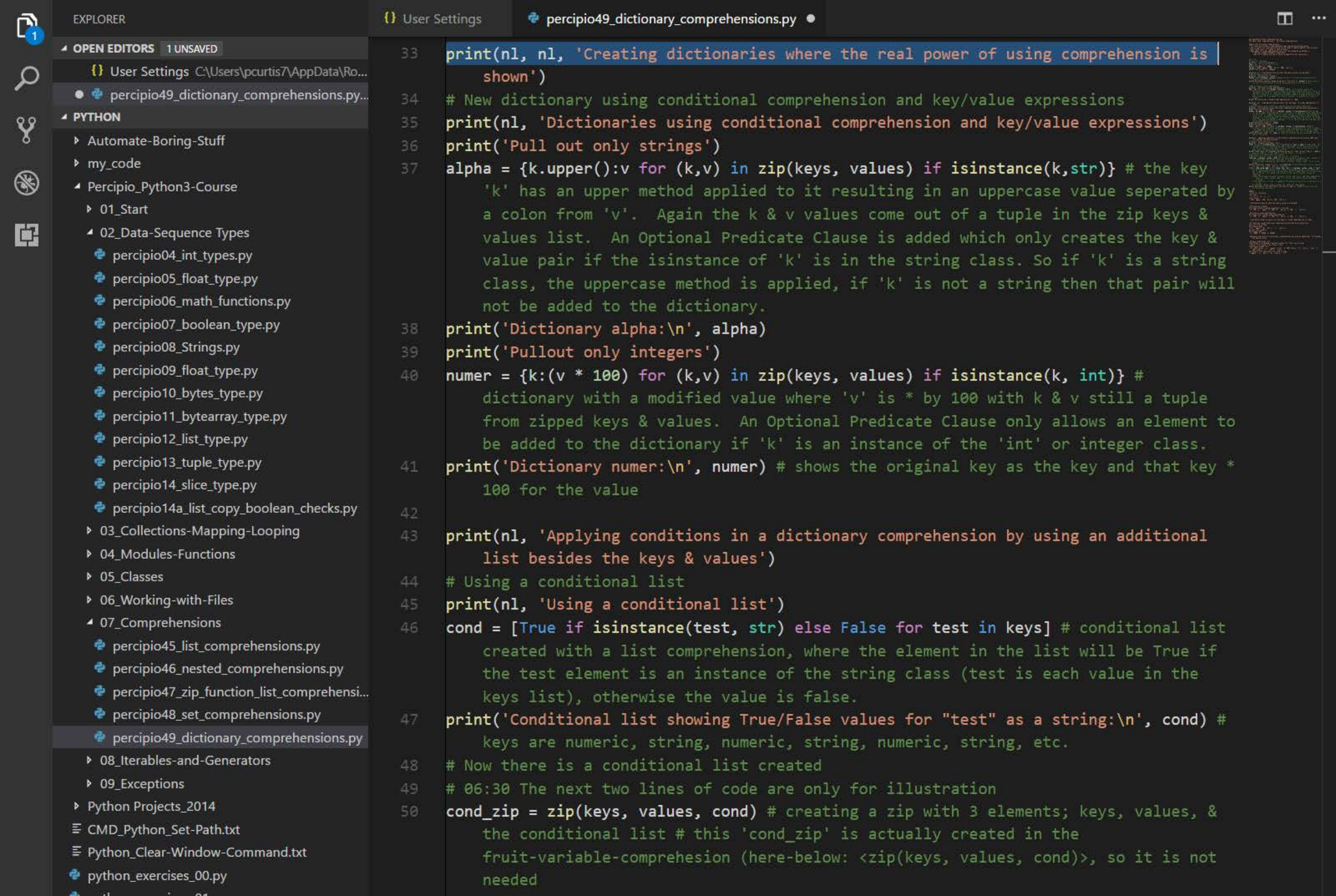
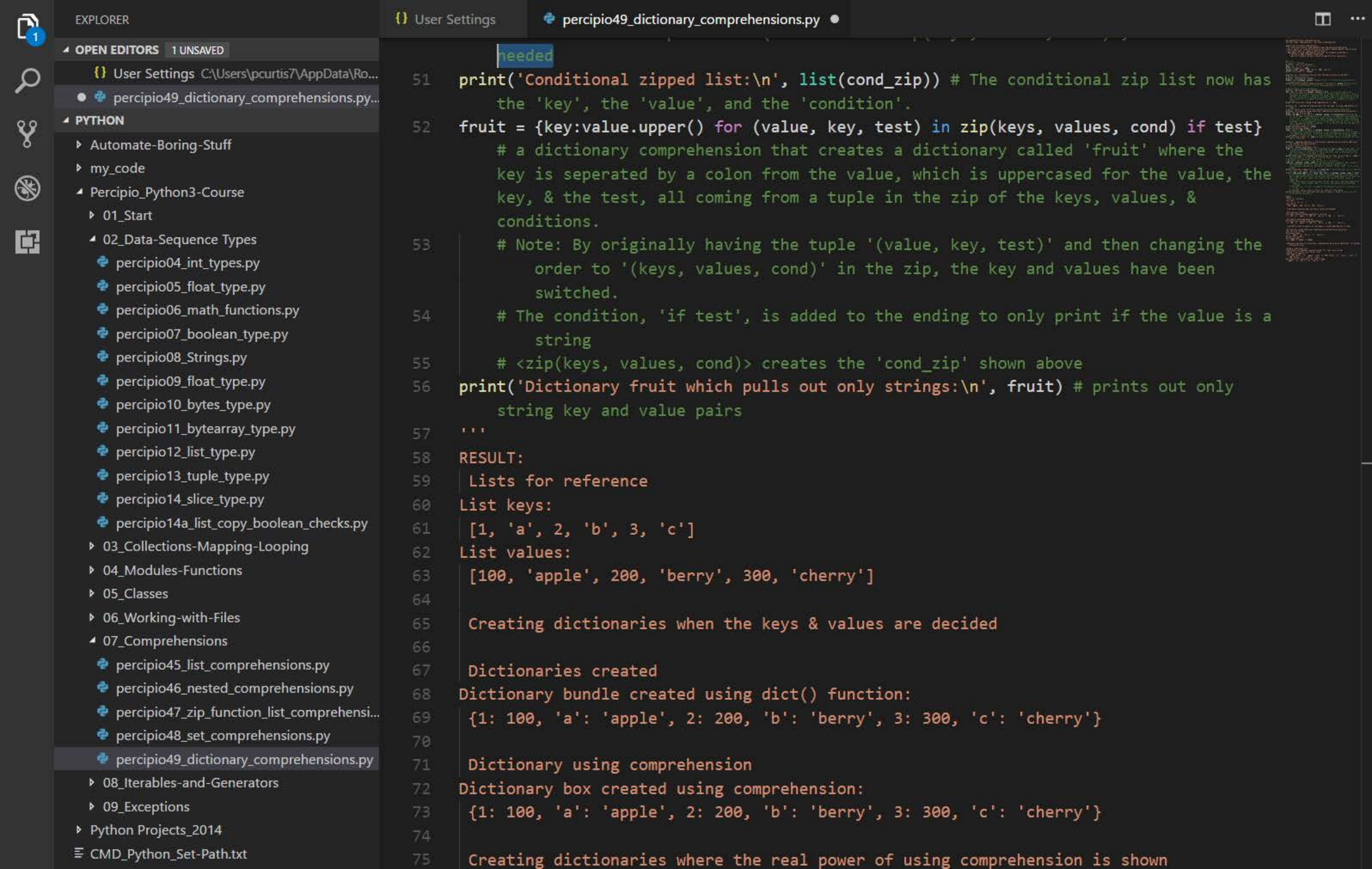
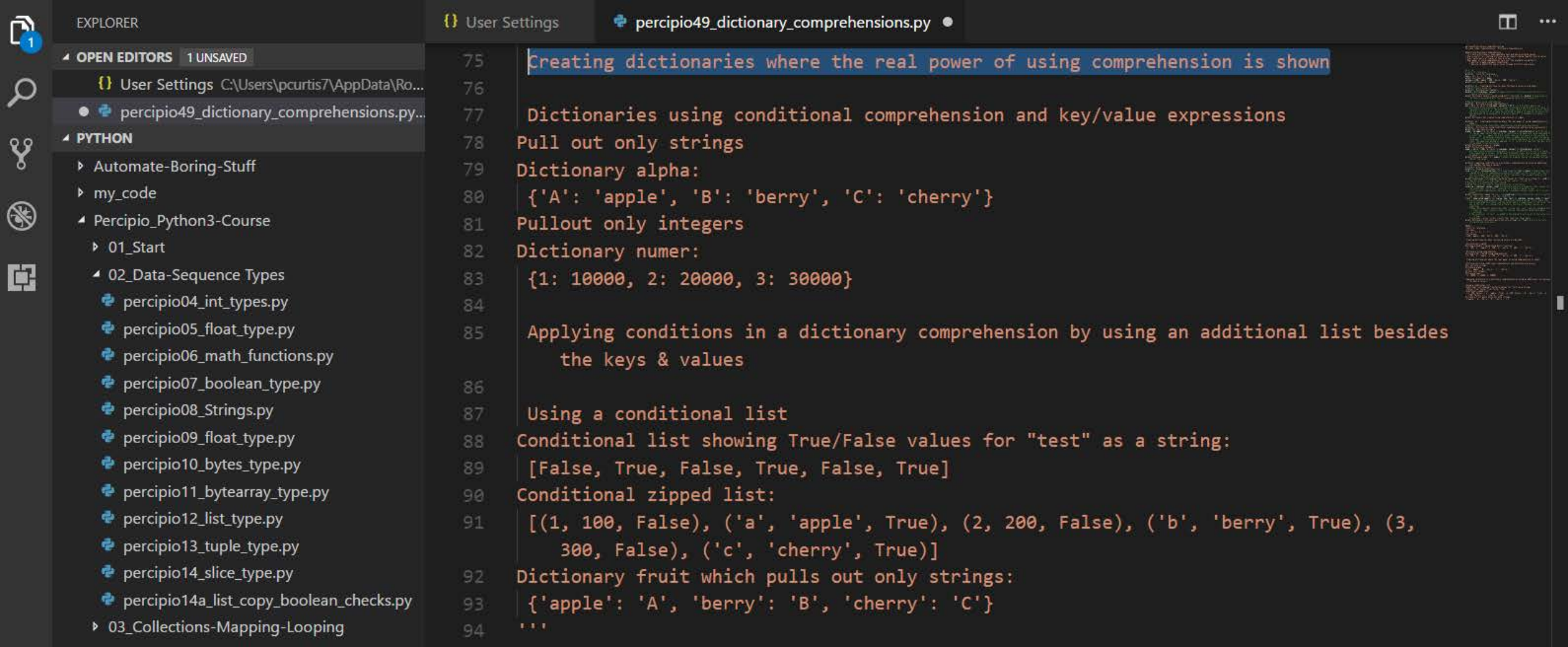

EXPLORER
{} User Settings
percipio49_dictionary_comprehensions.py
OPEN EDITORS 1 UNSAVED
{} User Settings C:\Users\pcurtis7\AppData\Ro...
percipio49_dictionary_comprehensions.py...
PYTHON
Automate-Boring-Stuff
my_code
Percipio_Python3-Course
01_Start
02_Data-Sequence Types
percipio04_int_types.py
percipio05_float_type.py
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_checks.py
03_Collections-Mapping-Looping
04_Modules-Functions
05_Classes
06_Working-with-Files
07_Comprehensions
percipio45_list_comprehensions.py
percipio46_nested_comprehensions.py
percipio47_zip_function_list_comprehensi...
percipio48_set_comprehensions.py
percipio49_dictionary_comprehensions.py
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
Python_Tutorials.md
'''
percipio49_dictionary_comprehensions.py
Percipio video: Comprehensions; Dictionary Comprehensions
Demonstrate Dictionary Comprehension
* To create a dictionary, need immutable keys and key-associated values
* Zip is useful for creating dictionaries by simply zipping together keys & values
* Zip creates pairs of tuples of keys & values
* Real power of using comprehension here over the standard zip method is
* ability to make statement conditional
* ability to modify the key or value through different expressions
'''
nl = '\n'
# Lists for reference
print(nl, 'Lists for reference')
keys = [1, 'a', 2, 'b', 3, 'c'] #
print('List keys:\n', keys)
values = [100, 'apple', 200, 'berry', 300, 'cherry']
print('List values:\n', values)
print(nl, nl, 'Creating dictionaries when the keys & values are decided')
# New dictionary using zip
print(nl, 'Dictionaries created')
bundle = dict(zip(keys, values)) # apply the dict() function to existig lists to create a new dictionary
print('Dictionary bundle created using dict() function:\n', bundle) # each value in the keys is associated with the corresponding value in the values list
# New dictionary using comprehension
print(nl, 'Dictionary using comprehension')
box = {k:v for (k,v) in zip(keys, values)} # NOTE: curly-brackets denote new dictionary # The expression (k,v) needs to indicate a key & value (i.e.: here 'k' represents the key & 'v' represents the value. Note: k & v are arbitrary). Pulls the k & v values out of a tuple by iterating through a forLoop in the zip of the keys & values (K is assigned to a value in keys & v is assigned to a value in values).
print('Dictionary box created using comprehension:\n', box)
print(nl, nl, 'Creating dictionaries where the real power of using comprehension is


EXPLORER
User Settings
percipio49_dictionary_comprehensions.py
OPEN EDITORS 1 UNSAVED
User Settings C:\Users\pcurtis7\AppData\Ro...
percipio49_dictionary_comprehensions.py...
PYTHON
Automate-Boring-Stuff
my_code
Percipio_Python3-Course
01_Start
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_checks.py
03_Collections-Mapping-Looping
04_Modules-Functions
05_Classes
06_Working-with-Files
07_Comprehensions
percipio45_list_comprehensions.py
percipio46_nested_comprehensions.py
percipio47_zip_function_list_comprehensi...
percipio48_set_comprehensions.py
percipio49_dictionary_comprehensions.py
08_Iterables-and-Generators
09_Exceptions
Python Projects_2014
CMD_Python_Set-Path.txt
Python_Clear-Window-Command.txt
python_exercises_00.py
33 print(nl, nl, 'Creating dictionaries where the real power of using comprehension is shown')
34 # New dictionary using conditional comprehension and key/value expressions
35 print(nl, 'Dictionaries using conditional comprehension and key/value expressions')
36 print('Pull out only strings')
37 alpha = {k.upper():v for (k,v) in zip(keys, values) if isinstance(k,str)} # the key 'k' has an upper method applied to it resulting in an uppercase value seperated by a colon from 'v'. Again the k & v values come out of a tuple in the zip keys & values list. An Optional Predicate Clause is added which only creates the key & value pair if the isinstance of 'k' is in the string class. So if 'k' is a string class, the uppercase method is applied, if 'k' is not a string then that pair will not be added to the dictionary.
38 print('Dictionary alpha:\n', alpha)
39 print('Pullout only integers')
40 numer = {k:(v * 100) for (k,v) in zip(keys, values) if isinstance(k, int)} # dictionary with a modified value where 'v' is * by 100 with k & v still a tuple from zipped keys & values. An Optional Predicate Clause only allows an element to be added to the dictionary if 'k' is an instance of the 'int' or integer class.
41 print('Dictionary numer:\n', numer) # shows the original key as the key and that key * 100 for the value
42
43 print(nl, 'Applying conditions in a dictionary comprehension by using an additional list besides the keys & values')
44 # Using a conditional list
45 print(nl, 'Using a conditional list')
46 cond = [True if isinstance(test, str) else False for test in keys] # conditional list created with a list comprehension, where the element in the list will be True if the test element is an instance of the string class (test is each value in the keys list), otherwise the value is false.
47 print('Conditional list showing True/False values for "test" as a string:\n', cond) # keys are numeric, string, numeric, string, numeric, string, etc.
48 # Now there is a conditional list created
49 # 06:30 The next two lines of code are only for illustration
50 cond_zip = zip(keys, values, cond) # creating a zip with 3 elements; keys, values, & the conditional list # this 'cond_zip' is actually created in the fruit-variable-comprehesion (here-below: <zip(keys, values, cond)>, so it is not needed


EXPLORER
User Settings
percipio49_dictionary_comprehensions.py
OPEN EDITORS 1 UNSAVED
User Settings C:\Users\pcurtis7\AppData\Ro...
percipio49_dictionary_comprehensions.py...
PYTHON
Automate-Boring-Stuff
my_code
Percipio_Python3-Course
01_Start
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_checks.py
03_Collections-Mapping-Looping
04_Modules-Functions
05_Classes
06_Working-with-Files
07_Comprehensions
percipio45_list_comprehensions.py
percipio46_nested_comprehensions.py
percipio47_zip_function_list_comprehensi...
percipio48_set_comprehensions.py
percipio49_dictionary_comprehensions.py
08_Iterables-and-Generators
09_Exceptions
Python Projects_2014
CMD_Python_Set-Path.txt
needed
print('Conditional zipped list:\n', list(cond_zip)) # The conditional zip list now has the 'key', the 'value', and the 'condition'.
fruit = {key:value.upper() for (value, key, test) in zip(keys, values, cond) if test}
# a dictionary comprehension that creates a dictionary called 'fruit' where the key is seperated by a colon from the value, which is uppercased for the value, the key, & the test, all coming from a tuple in the zip of the keys, values, & conditions.
# Note: By originally having the tuple '(value, key, test)' and then changing the order to '(keys, values, cond)' in the zip, the key and values have been switched.
# The condition, 'if test', is added to the ending to only print if the value is a string
# <zip(keys, values, cond)> creates the 'cond_zip' shown above
print('Dictionary fruit which pulls out only strings:\n', fruit) # prints out only string key and value pairs
'''
RESULT:
Lists for reference
List keys:
[1, 'a', 2, 'b', 3, 'c']
List values:
[100, 'apple', 200, 'berry', 300, 'cherry']
Creating dictionaries when the keys & values are decided
Dictionaries created
Dictionary bundle created using dict() function:
{1: 100, 'a': 'apple', 2: 200, 'b': 'berry', 3: 300, 'c': 'cherry'}
Dictionary using comprehension
Dictionary box created using comprehension:
{1: 100, 'a': 'apple', 2: 200, 'b': 'berry', 3: 300, 'c': 'cherry'}
Creating dictionaries where the real power of using comprehension is shown


EXPLORER
OPEN EDITORS 1 UNSAVED
User Settings C:\Users\pcurtis7\AppData\Ro...
percipio49_dictionary_comprehensions.py...
PYTHON
Automate-Boring-Stuff
my_code
Percipio_Python3-Course
01_Start
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_checks.py
03_Collections-Mapping-Looping
User Settings
percipio49_dictionary_comprehensions.py
Creating dictionaries where the real power of using comprehension is shown
Dictionaries using conditional comprehension and key/value expressions
Pull out only strings
Dictionary alpha:
{'A': 'apple', 'B': 'berry', 'C': 'cherry'}
Pullout only integers
Dictionary numer:
{1: 10000, 2: 20000, 3: 30000}
Applying conditions in a dictionary comprehension by using an additional list besides the keys & values
Using a conditional list
Conditional list showing True/False values for "test" as a string:
[False, True, False, True, False, True]
Conditional zipped list:
[(1, 100, False), ('a', 'apple', True), (2, 200, False), ('b', 'berry', True), (3, 300, False), ('c', 'cherry', True)]
Dictionary fruit which pulls out only strings:
{'apple': 'A', 'berry': 'B', 'cherry': 'C'}
'''