

The image shows the Visual Studio Code editor with a Python file named `percipio47_zip_function_list_comprehensions_matrix_multiply.py` open. The Explorer sidebar on the left shows a project structure with folders for data types, collections, modules, classes, file operations, comprehensions, iterables, and exceptions, along with Python projects and tutorials. The main editor area displays the following code:

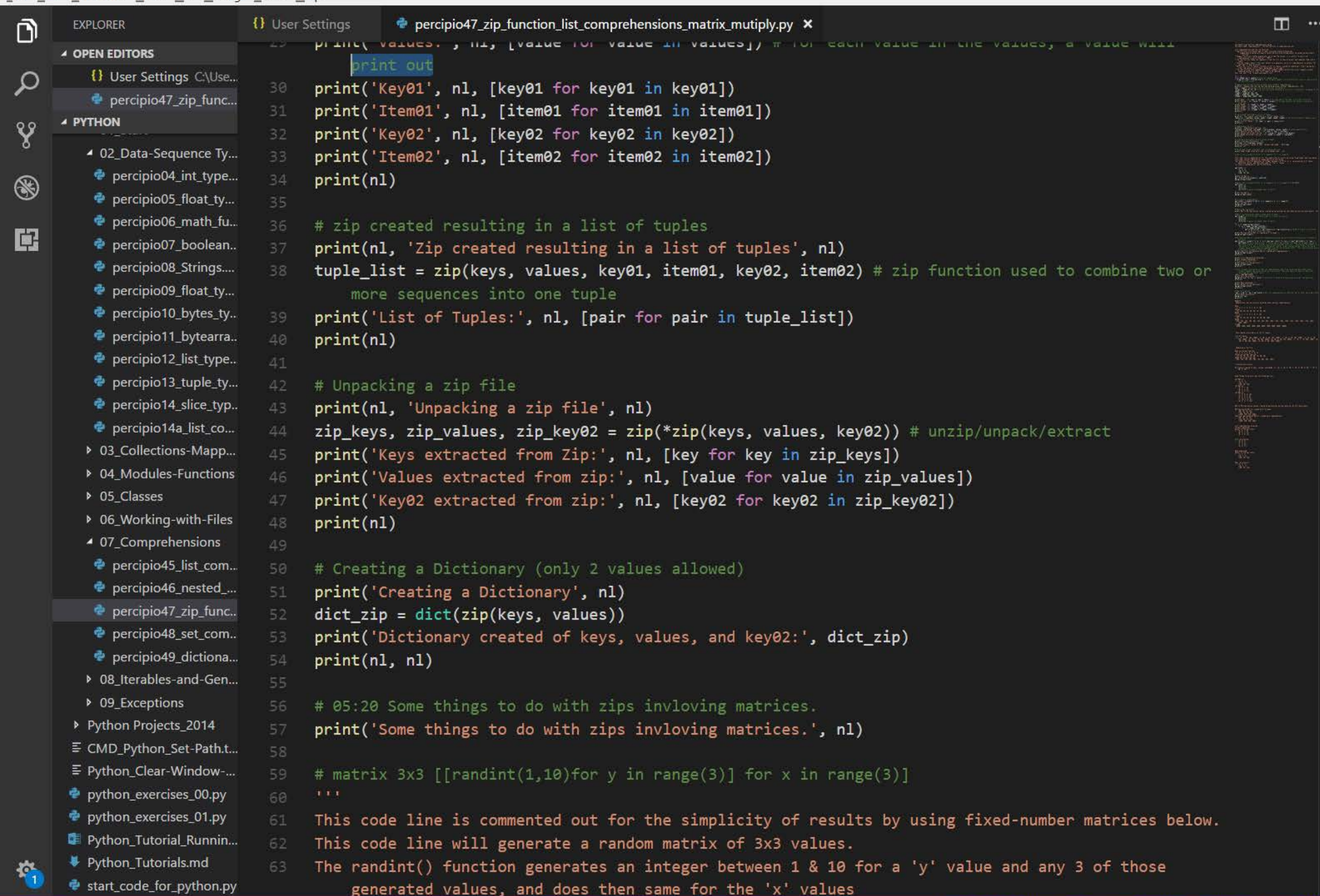
```
'''
percipio47_zip_function_comprehensions.py.py
Percipio video: Comprehensions; The Zip Function in Comprehensions.py

List comprehensions and the zip function
* Zip functions are a great way to take lists and create dictionaries
    * Zipped keys & values are easily converted into a dictionary object by using the dictionary
      function
* Range() functions creates a generator so to see the values, it's usefull to use a list
  comprehension to print them out. *WHY??
* The zip function takes two sequences (like the list of keys & values) and combines them into a
  tuple
* Like the range() object, the zip() object is a generator so a list comprehension is helpful for
  each pair of keys & values.
* Notice the 'List of Tuples' contains a list of tuples created by combining 1 item from key & 1
  item from values into 1 tuple, going down for all pairs.
* The zip function only combines equivalent element numbers within the lists, elements without
  numbered matches in other lists seem to be ignored
Use 'zip' and '*zip' to pack and unpack lists
'''

nl = '\n'
from random import randint # used for random values
from pprint import PrettyPrinter # from the pprint module, the PrettyPrinter class is used to print
    out matrices nicely

# Demonstrate the zip function working alone, without comprehension
print(nl, 'Demonstrate the zip function working alone, without comprehension', nl)
keys = range(1, 11) # keys from 1 up to but not including 11
values = range(10, 91, 10) # values starting at 10 going up to but not including 91, stepping by 10
key01 = range(1, 11, 1)
item01 = range(10, 101, 10)
key02 = range(201, 2002, 100)
item02 = range(1000, 10001, 1000)

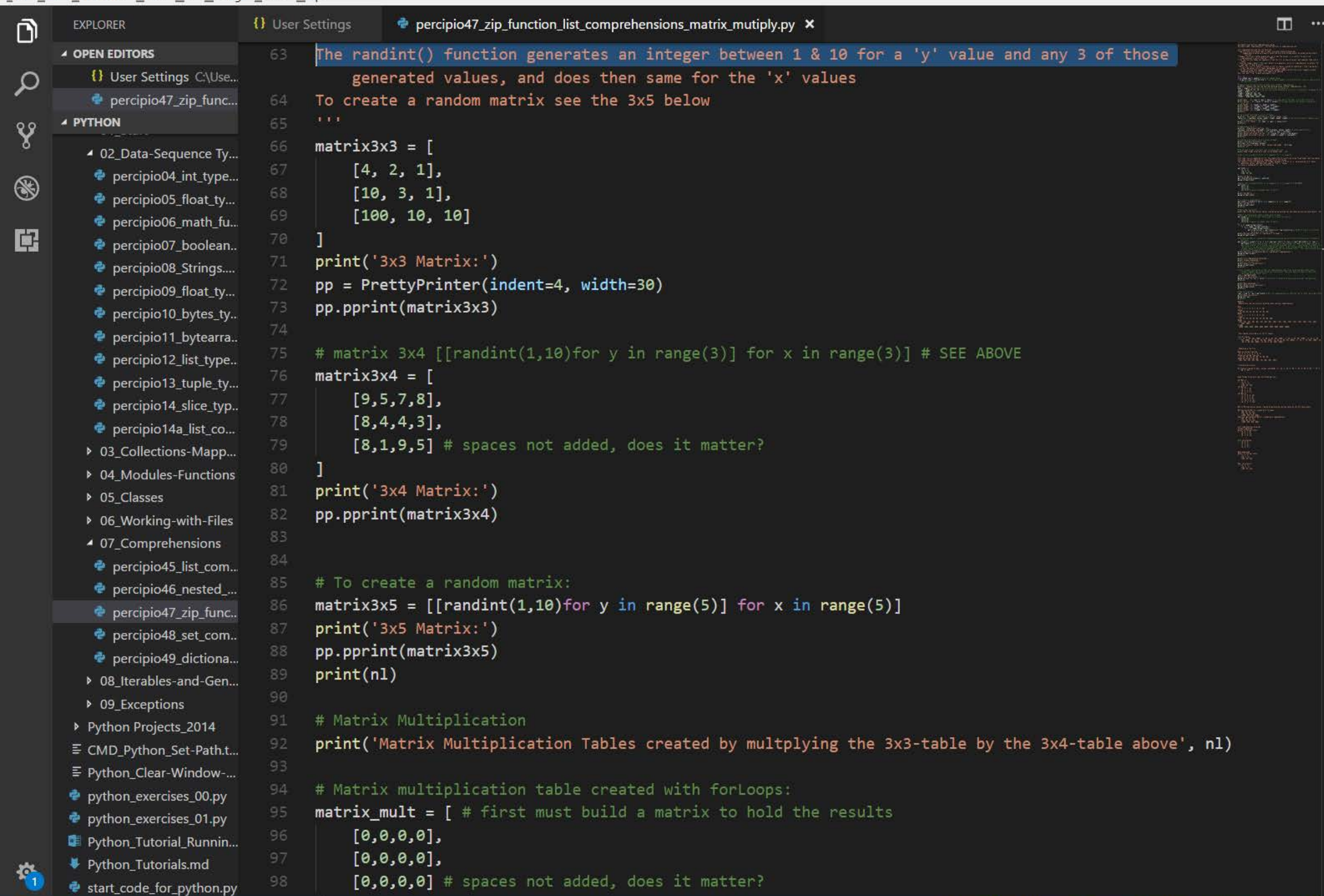
print('Keys:', nl, [key for key in keys]) # for each key in the keys, a list will print out.
print('Values:', nl, [value for value in values]) # for each value in the values, a value will
    print out
```

The image shows the Visual Studio Code editor with a Python file named `percipio47_zip_function_list_comprehensions_matrix_multiply.py` open. The left sidebar contains the Explorer and Python toolbars. The Explorer shows a project structure with folders like `02_Data-Sequence Ty...`, `03_Collections-Mapp...`, `04_Modules-Functions`, `05_Classes`, `06_Working-with-Files`, `07_Comprehensions`, `08_Iterables-and-Gen...`, and `09_Exceptions`. The Python toolbar shows various file types. The main editor area displays the following Python code:

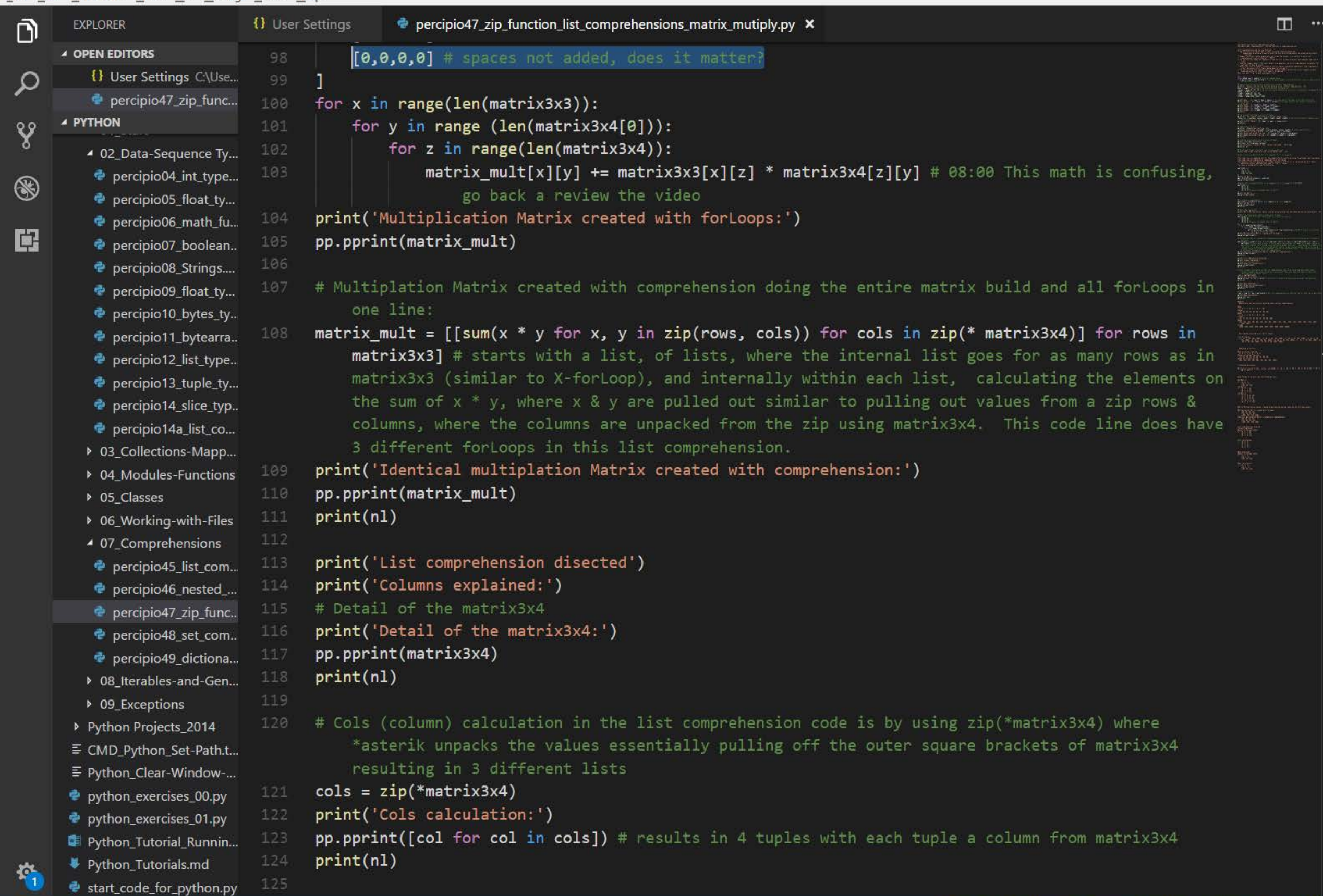
```
29 print(values, nl, [value for value in values]) # for each value in the values, a value will
    print out
30 print('Key01', nl, [key01 for key01 in key01])
31 print('Item01', nl, [item01 for item01 in item01])
32 print('Key02', nl, [key02 for key02 in key02])
33 print('Item02', nl, [item02 for item02 in item02])
34 print(nl)
35
36 # zip created resulting in a list of tuples
37 print(nl, 'Zip created resulting in a list of tuples', nl)
38 tuple_list = zip(keys, values, key01, item01, key02, item02) # zip function used to combine two or
    more sequences into one tuple
39 print('List of Tuples:', nl, [pair for pair in tuple_list])
40 print(nl)
41
42 # Unpacking a zip file
43 print(nl, 'Unpacking a zip file', nl)
44 zip_keys, zip_values, zip_key02 = zip(*zip(keys, values, key02)) # unzip/unpack/extract
45 print('Keys extracted from Zip:', nl, [key for key in zip_keys])
46 print('Values extracted from zip:', nl, [value for value in zip_values])
47 print('Key02 extracted from zip:', nl, [key02 for key02 in zip_key02])
48 print(nl)
49
50 # Creating a Dictionary (only 2 values allowed)
51 print('Creating a Dictionary', nl)
52 dict_zip = dict(zip(keys, values))
53 print('Dictionary created of keys, values, and key02:', dict_zip)
54 print(nl, nl)
55
56 # 05:20 Some things to do with zips involving matrices.
57 print('Some things to do with zips involving matrices.', nl)
58
59 # matrix 3x3 [[randint(1,10)for y in range(3)] for x in range(3)]
60 '''
61 This code line is commented out for the simplicity of results by using fixed-number matrices below.
62 This code line will generate a random matrix of 3x3 values.
63 The randint() function generates an integer between 1 & 10 for a 'y' value and any 3 of those
    generated values, and does then same for the 'x' values
```

The status bar at the bottom indicates: Ln 29, Col 96 (9 selected) Spaces: 4 UTF-8 CRLF Python



The image shows the Visual Studio Code editor with a Python file open. The Explorer sidebar on the left shows a project structure with folders for '02_Data-Sequence Types', '03_Collections-Mapping', '04_Modules-Functions', '05_Classes', '06_Working-with-Files', '07_Comprehensions', '08_Iterables-and-Generators', and '09_Exceptions'. The file 'percipio47_zip_function_list_comprehensions_matrix_multiply.py' is selected under the '07_Comprehensions' folder. The main editor area shows the code for this file, which includes comments and code for generating 3x3, 3x4, and 3x5 matrices using the `randint()` function, and then performing matrix multiplication. The code is as follows:

```
63 The randint() function generates an integer between 1 & 10 for a 'y' value and any 3 of those
64 generated values, and does then same for the 'x' values
65 To create a random matrix see the 3x5 below
66 '''
67 matrix3x3 = [
68     [4, 2, 1],
69     [10, 3, 1],
70     [100, 10, 10]
71 ]
72 print('3x3 Matrix:')
73 pp = PrettyPrinter(indent=4, width=30)
74 pp.pprint(matrix3x3)
75 # matrix 3x4 [[randint(1,10)for y in range(3)] for x in range(3)] # SEE ABOVE
76 matrix3x4 = [
77     [9,5,7,8],
78     [8,4,4,3],
79     [8,1,9,5] # spaces not added, does it matter?
80 ]
81 print('3x4 Matrix:')
82 pp.pprint(matrix3x4)
83
84
85 # To create a random matrix:
86 matrix3x5 = [[randint(1,10)for y in range(5)] for x in range(5)]
87 print('3x5 Matrix:')
88 pp.pprint(matrix3x5)
89 print(nl)
90
91 # Matrix Multiplication
92 print('Matrix Multiplication Tables created by multiplying the 3x3-table by the 3x4-table above', nl)
93
94 # Matrix multiplication table created with forLoops:
95 matrix_mult = [ # first must build a matrix to hold the results
96     [0,0,0,0],
97     [0,0,0,0],
98     [0,0,0,0] # spaces not added, does it matter?
```

EXPLORER

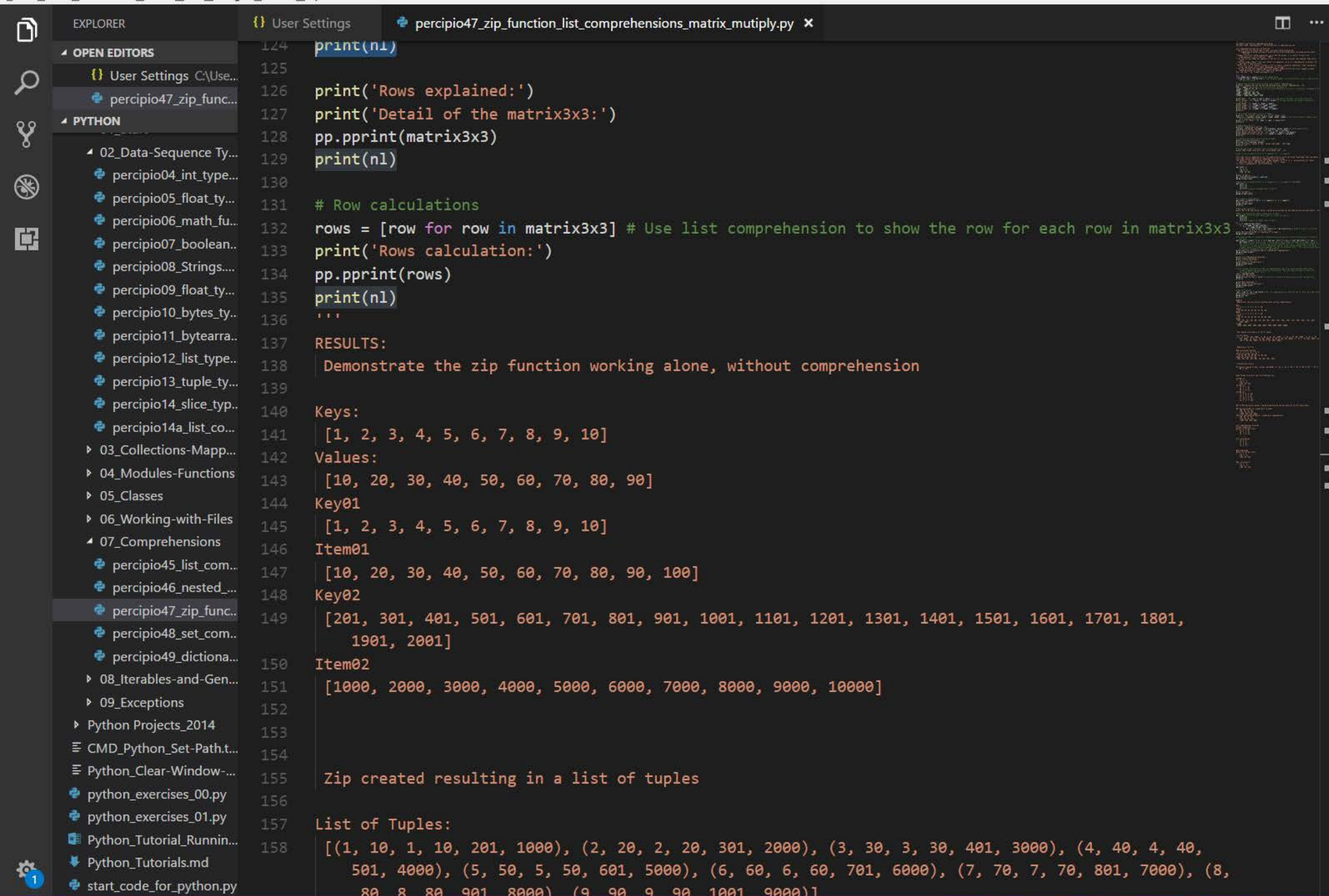
OPEN EDITORS

- User Settings C:\Use...
- percipio47_zip_func...

PYTHON

- 02_Data-Sequence Ty...
- percipio04_int_type...
- percipio05_float_ty...
- percipio06_math_fu...
- percipio07_boolean...
- percipio08_Strings...
- percipio09_float_ty...
- percipio10_bytes_ty...
- percipio11_bytearra...
- percipio12_list_type...
- percipio13_tuple_ty...
- percipio14a_list_co...
- 03_Collections-Mapp...
- 04_Modules-Functions
- 05_Classes
- 06_Working-with-Files
- 07_Comprehensions
 - percipio45_list_com...
 - percipio46_nested_...
 - percipio47_zip_func...
 - percipio48_set_com...
 - percipio49_dictiona...
- 08_Iterables-and-Gen...
- 09_Exceptions
- Python Projects_2014
- CMD_Python_Set-Path.t...
- Python_Clear-Window-...
- python_exercises_00.py
- python_exercises_01.py
- Python_Tutorial_Runnin...
- Python_Tutorials.md
- start_code_for_python.py

```
108 [0,0,0,0] # spaces not added, does it matter?
109 ]
110 for x in range(len(matrix3x3)):
111     for y in range (len(matrix3x4[0])):
112         for z in range(len(matrix3x4)):
113             matrix_mult[x][y] += matrix3x3[x][z] * matrix3x4[z][y] # 08:00 This math is confusing,
114                                     go back a review the video
115 print('Multiplication Matrix created with forLoops:')
116 pp.pprint(matrix_mult)
117
118 # Multiplation Matrix created with comprehension doing the entire matrix build and all forLoops in
119 one line:
120 matrix_mult = [[sum(x * y for x, y in zip(rows, cols)) for cols in zip(* matrix3x4)] for rows in
121 matrix3x3] # starts with a list, of lists, where the internal list goes for as many rows as in
122 matrix3x3 (similar to X-forLoop), and internally within each list, calculating the elements on
123 the sum of x * y, where x & y are pulled out similar to pulling out values from a zip rows &
124 columns, where the columns are unpacked from the zip using matrix3x4. This code line does have
125 3 different forLoops in this list comprehension.
126
127 print('Identical multiplation Matrix created with comprehension:')
128 pp.pprint(matrix_mult)
129 print(nl)
130
131 print('List comprehension dissected')
132 print('Columns explained:')
133 # Detail of the matrix3x4
134 print('Detail of the matrix3x4:')
135 pp.pprint(matrix3x4)
136 print(nl)
137
138 # Cols (column) calculation in the list comprehension code is by using zip(*matrix3x4) where
139 *asterik unpacks the values essentially pulling off the outer square brackets of matrix3x4
140 resulting in 3 different lists
141 cols = zip(*matrix3x4)
142 print('Cols calculation:')
143 pp.pprint([col for col in cols]) # results in 4 tuples with each tuple a column from matrix3x4
144 print(nl)
```

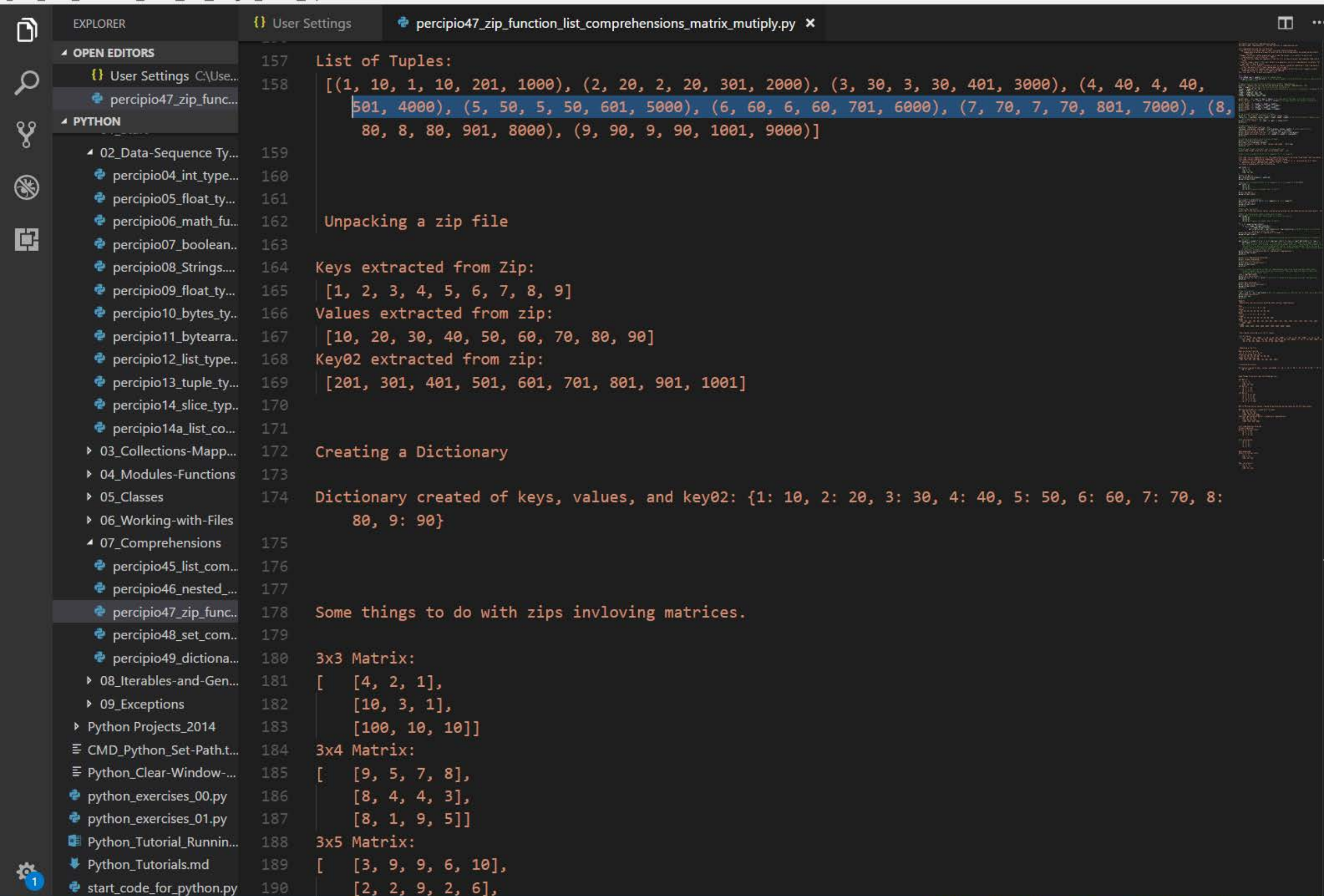
The image shows the Visual Studio Code editor with a Python file named `percipio47_zip_function_list_comprehensions_matrix_multiply.py` open. The left sidebar shows the Explorer view with a tree structure of files and folders. The main editor area displays the code and its output.

EXPLORER

- OPEN EDITORS
 - User Settings C:\Use...
 - percipio47_zip_func...
- PYTHON
 - 02_Data-Sequence Ty...
 - percipio04_int_type...
 - percipio05_float_ty...
 - percipio06_math_fu...
 - percipio07_boolean...
 - percipio08.Strings...
 - percipio09_float_ty...
 - percipio10_bytes_ty...
 - percipio11_bytearra...
 - percipio12_list_type...
 - percipio13_tuple_ty...
 - percipio14.slice_typ...
 - percipio14a_list_co...
 - 03_Collections-Mapp...
 - 04_Modules-Functions
 - 05_Classes
 - 06_Working-with-Files
 - 07_Comprehensions
 - percipio45_list_com...
 - percipio46_nested_...
 - percipio47_zip_func...
 - percipio48_set_com...
 - percipio49_dictiona...
 - 08_iterables-and-Gen...
 - 09_Exceptions
 - Python Projects_2014
 - CMD_Python_Set-Path.t...
 - Python_Clear-Window-...
 - python_exercises_00.py
 - python_exercises_01.py
 - Python_Tutorial_Runnin...
 - Python_Tutorials.md
 - start_code_for_python.py

Code:

```
124 print(n1)
125
126 print('Rows explained:')
127 print('Detail of the matrix3x3:')
128 pp.pprint(matrix3x3)
129 print(n1)
130
131 # Row calculations
132 rows = [row for row in matrix3x3] # Use list comprehension to show the row for each row in matrix3x3
133 print('Rows calculation:')
134 pp.pprint(rows)
135 print(n1)
136 '''
137 RESULTS:
138     Demonstrate the zip function working alone, without comprehension
139
140 Keys:
141     [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
142 Values:
143     [10, 20, 30, 40, 50, 60, 70, 80, 90]
144 Key01
145     [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
146 Item01
147     [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
148 Key02
149     [201, 301, 401, 501, 601, 701, 801, 901, 1001, 1101, 1201, 1301, 1401, 1501, 1601, 1701, 1801,
150         1901, 2001]
151 Item02
152     [1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000]
153
154
155 Zip created resulting in a list of tuples
156
157 List of Tuples:
158     [(1, 10, 1, 10, 201, 1000), (2, 20, 2, 20, 301, 2000), (3, 30, 3, 30, 401, 3000), (4, 40, 4, 40,
159         501, 4000), (5, 50, 5, 50, 601, 5000), (6, 60, 6, 60, 701, 6000), (7, 70, 7, 70, 801, 7000), (8,
160         80, 8, 80, 901, 8000), (9, 90, 9, 90, 1001, 9000), (10, 100, 10, 100, 1101, 11000), (11, 110, 11, 110, 1201, 12000), (12, 120, 12, 120, 1301, 13000), (13, 130, 13, 130, 1401, 14000), (14, 140, 14, 140, 1501, 15000), (15, 150, 15, 150, 1601, 16000), (16, 160, 16, 160, 1701, 17000), (17, 170, 17, 170, 1801, 18000), (18, 180, 18, 180, 1901, 19000), (19, 190, 19, 190, 2001, 20000), (20, 200, 20, 200, 2101, 21000)]
```

The image shows the Visual Studio Code editor with a Python file named `percipio47_zip_function_list_comprehensions_matrix_multiply.py` open. The Explorer sidebar on the left shows a project structure with folders for data sequences, collections, modules, classes, files, comprehensions, iterables, and exceptions, along with Python projects and tutorials. The main editor area displays the following code:

```
157 List of Tuples:
158 [(1, 10, 1, 10, 201, 1000), (2, 20, 2, 20, 301, 2000), (3, 30, 3, 30, 401, 3000), (4, 40, 4, 40,
501, 4000), (5, 50, 5, 50, 601, 5000), (6, 60, 6, 60, 701, 6000), (7, 70, 7, 70, 801, 7000), (8,
80, 8, 80, 901, 8000), (9, 90, 9, 90, 1001, 9000)]

Unpacking a zip file

Keys extracted from Zip:
[1, 2, 3, 4, 5, 6, 7, 8, 9]
Values extracted from zip:
[10, 20, 30, 40, 50, 60, 70, 80, 90]
Key02 extracted from zip:
[201, 301, 401, 501, 601, 701, 801, 901, 1001]

Creating a Dictionary

Dictionary created of keys, values, and key02: {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60, 7: 70, 8:
80, 9: 90}

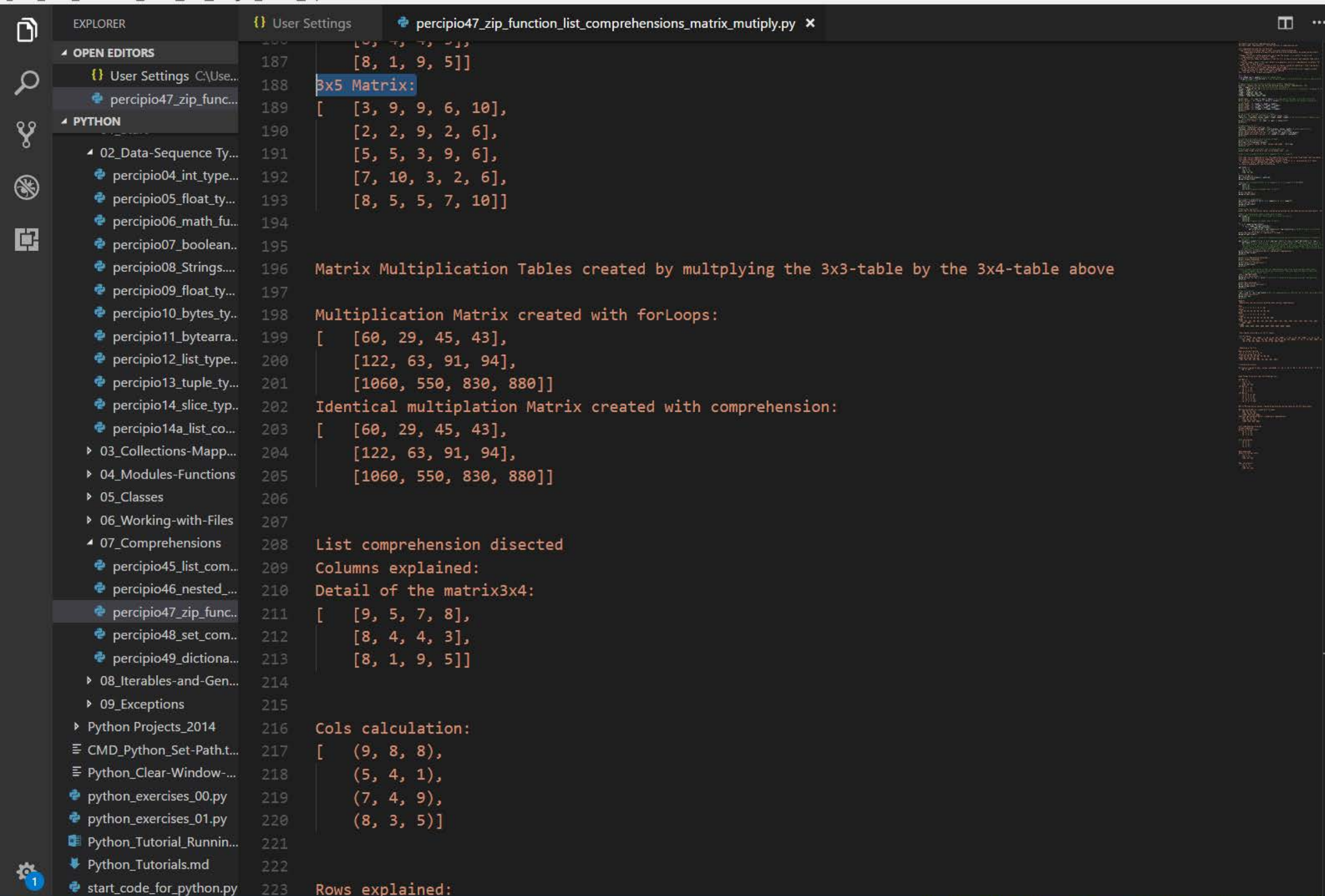
Some things to do with zips involving matrices.

3x3 Matrix:
[ [4, 2, 1],
  [10, 3, 1],
  [100, 10, 10]]

3x4 Matrix:
[ [9, 5, 7, 8],
  [8, 4, 4, 3],
  [8, 1, 9, 5]]

3x5 Matrix:
[ [3, 9, 9, 6, 10],
  [2, 2, 9, 2, 6],
```

The status bar at the bottom indicates the cursor is at line 158, column 99 (96 selected), with 4 spaces, UTF-8 encoding, CRLF line endings, and Python syntax.



The image shows the Visual Studio Code editor with a Python file named `percipio47_zip_function_list_comprehensions_matrix_multiply.py` open. The file explorer on the left shows a project structure with folders for data types, collections, modules, classes, working with files, comprehensions, iterables, and exceptions, as well as Python projects and tutorials.

```
187 [8, 1, 9, 5]]
188 3x5 Matrix:
189 [ [3, 9, 9, 6, 10],
190   [2, 2, 9, 2, 6],
191   [5, 5, 3, 9, 6],
192   [7, 10, 3, 2, 6],
193   [8, 5, 5, 7, 10]]
194
195
196 Matrix Multiplication Tables created by multiplying the 3x3-table by the 3x4-table above
197
198 Multiplication Matrix created with forLoops:
199 [ [60, 29, 45, 43],
200   [122, 63, 91, 94],
201   [1060, 550, 830, 880]]
202 Identical multiplication Matrix created with comprehension:
203 [ [60, 29, 45, 43],
204   [122, 63, 91, 94],
205   [1060, 550, 830, 880]]
206
207
208 List comprehension dissected
209 Columns explained:
210 Detail of the matrix3x4:
211 [ [9, 5, 7, 8],
212   [8, 4, 4, 3],
213   [8, 1, 9, 5]]
214
215
216 Cols calculation:
217 [ (9, 8, 8),
218   (5, 4, 1),
219   (7, 4, 9),
220   (8, 3, 5)]
221
222
223 Rows explained:
```




percipio47_zip_function_list_comprehensions_matrix_multiply.py

► 05 Classes

```
230 Rows calculation:
231 [ [4, 2, 1],
232   [10, 3, 1],
233   [100, 10, 10]]
234 '''
```