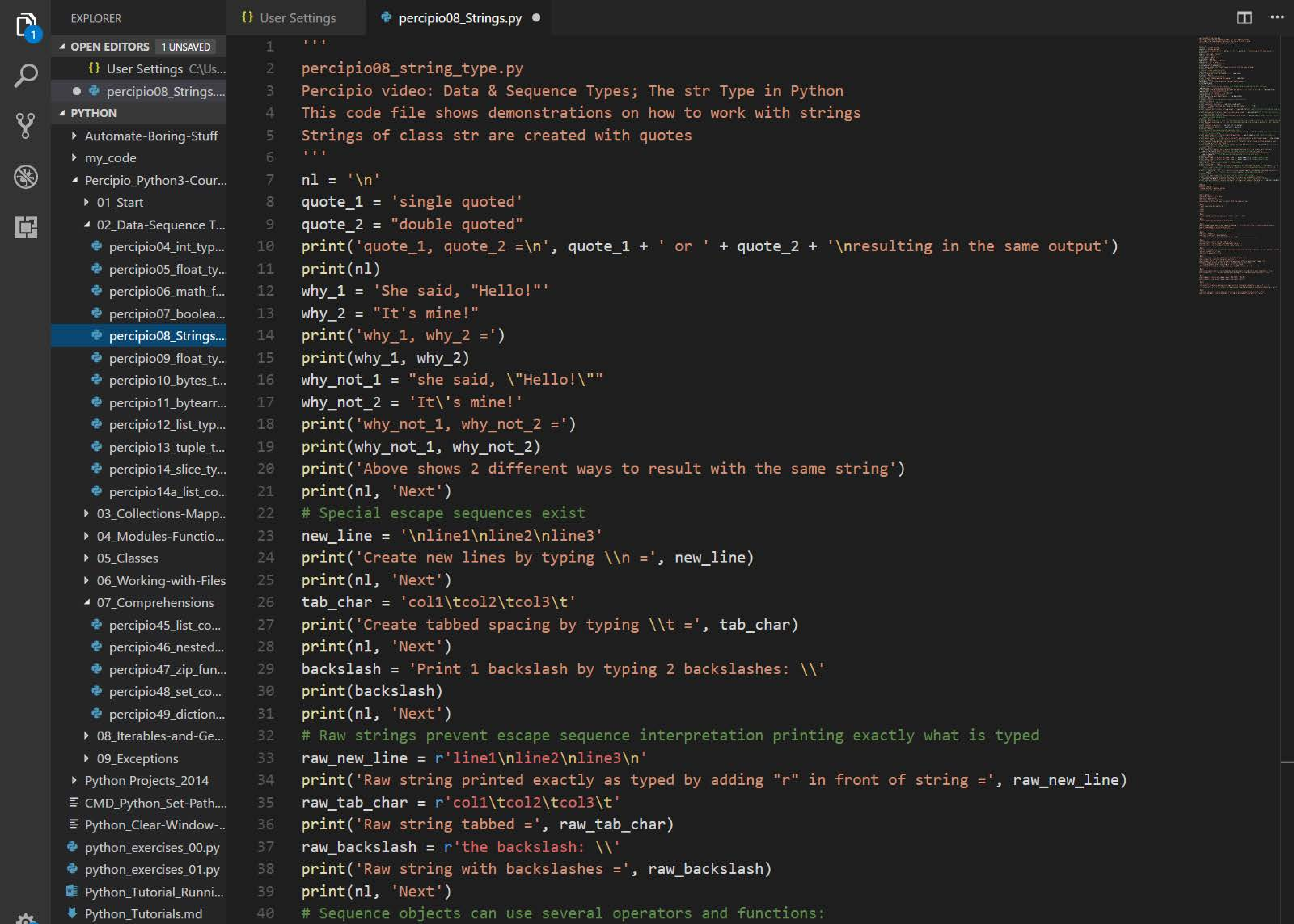
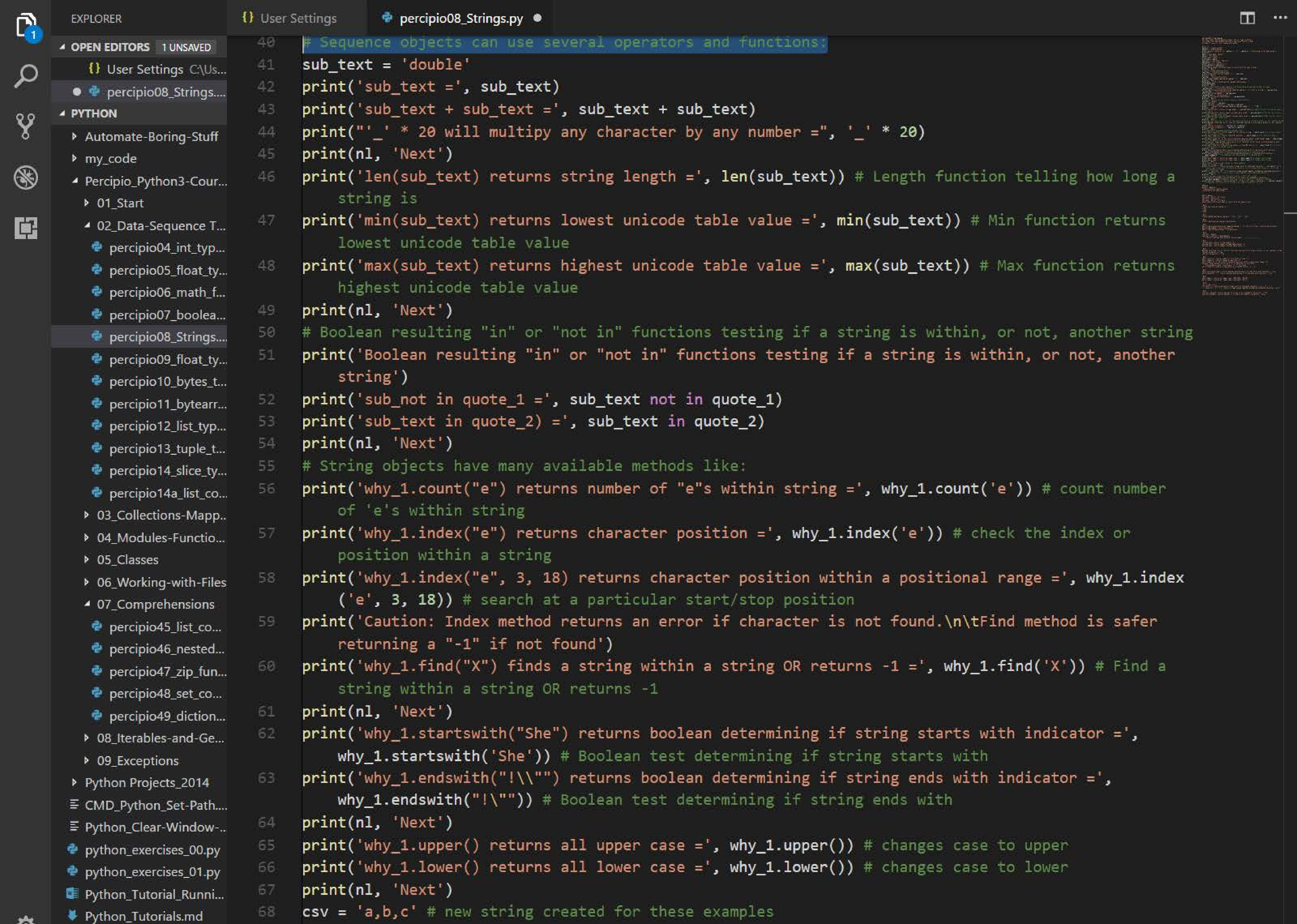
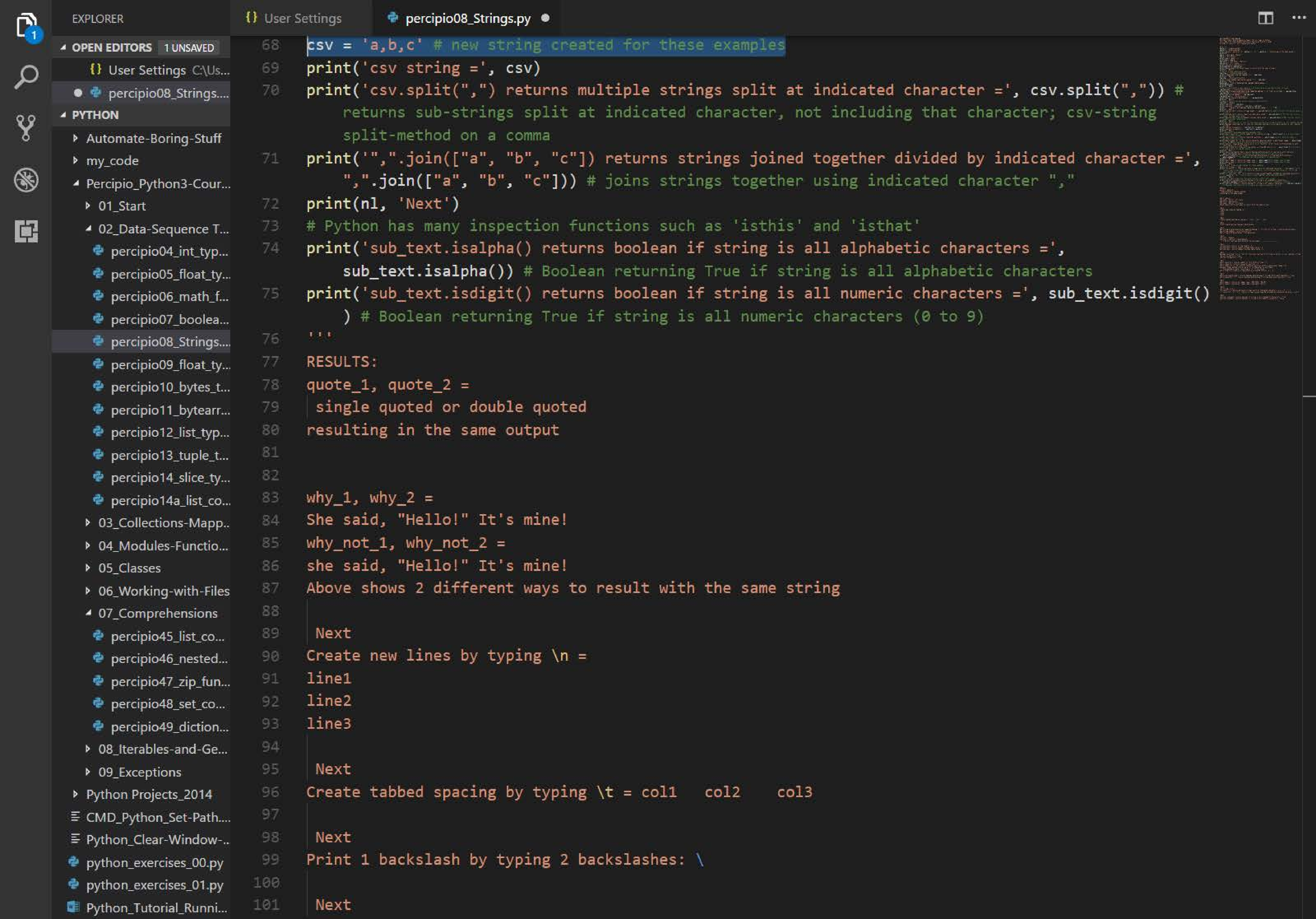


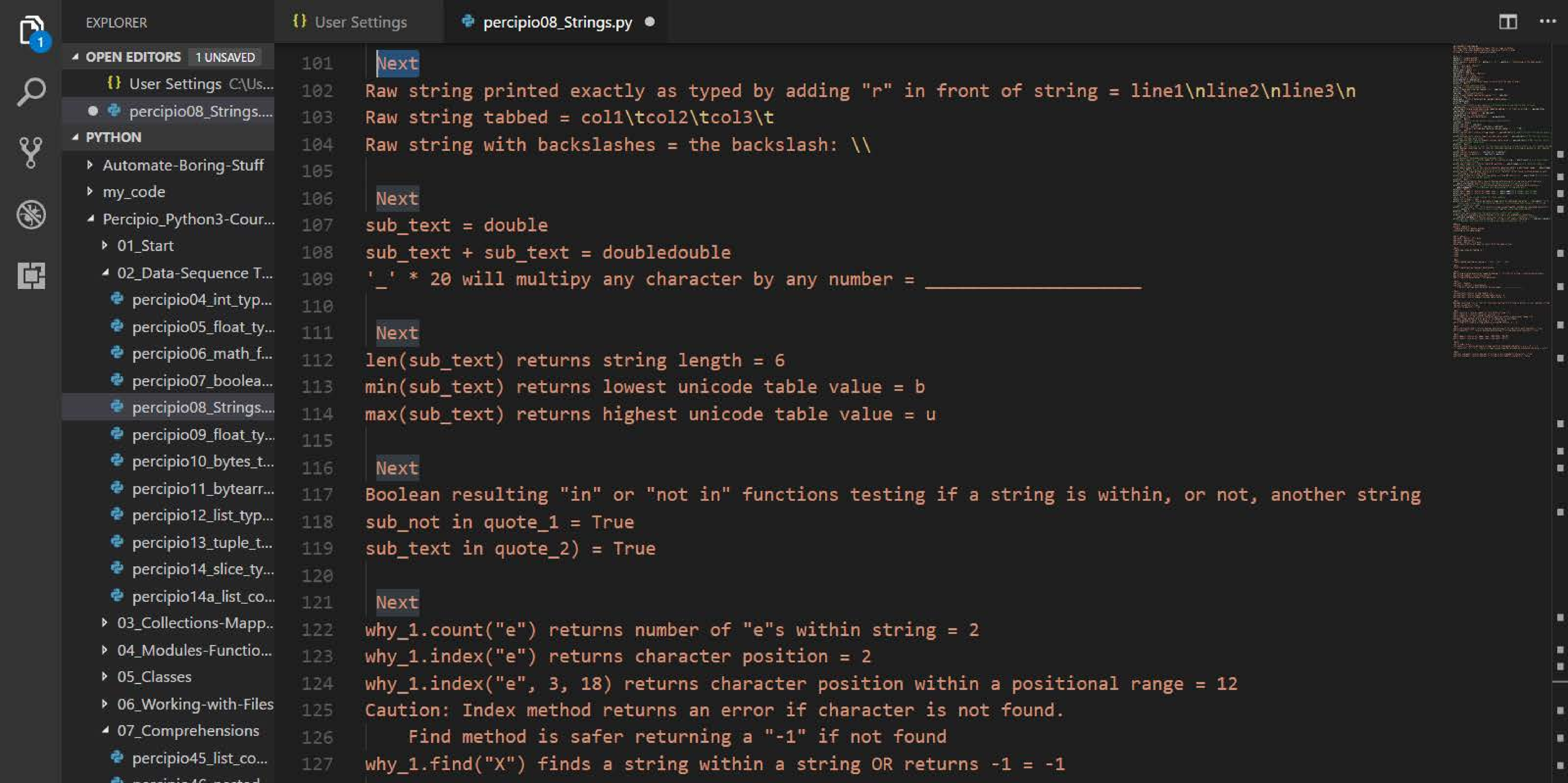
```python
'''
percipio08_string_type.py
Percipio video: Data & Sequence Types; The str Type in Python
This code file shows demonstrations on how to work with strings
Strings of class str are created with quotes
'''
nl = '\n'
quote_1 = 'single quoted'
quote_2 = "double quoted"
print('quote_1, quote_2 =\n', quote_1 + ' or ' + quote_2 + '\nresulting in the same output')
print(nl)
why_1 = 'She said, "Hello!"'
why_2 = "It's mine!"
print('why_1, why_2 =')
print(why_1, why_2)
why_not_1 = "she said, \"Hello!\""
why_not_2 = 'It\'s mine!'
print('why_not_1, why_not_2 =')
print(why_not_1, why_not_2)
print('Above shows 2 different ways to result with the same string')
print(nl, 'Next')
# Special escape sequences exist
new_line = '\nline1\nline2\nline3'
print('Create new lines by typing \\n =', new_line)
print(nl, 'Next')
tab_char = 'col1\tcol2\tcol3\t'
print('Create tabbed spacing by typing \\t =', tab_char)
print(nl, 'Next')
backslash = 'Print 1 backslash by typing 2 backslashes: \\'
print(backslash)
print(nl, 'Next')
# Raw strings prevent escape sequence interpretation printing exactly what is typed
raw_new_line = r'line1\nline2\nline3\n'
print('Raw string printed exactly as typed by adding "r" in front of string =', raw_new_line)
raw_tab_char = r'col1\tcol2\tcol3\t'
print('Raw string tabbed =', raw_tab_char)
raw_backslash = r'the backslash: \\'
print('Raw string with backslashes =', raw_backslash)
print(nl, 'Next')
# Sequence objects can use several operators and functions:
```



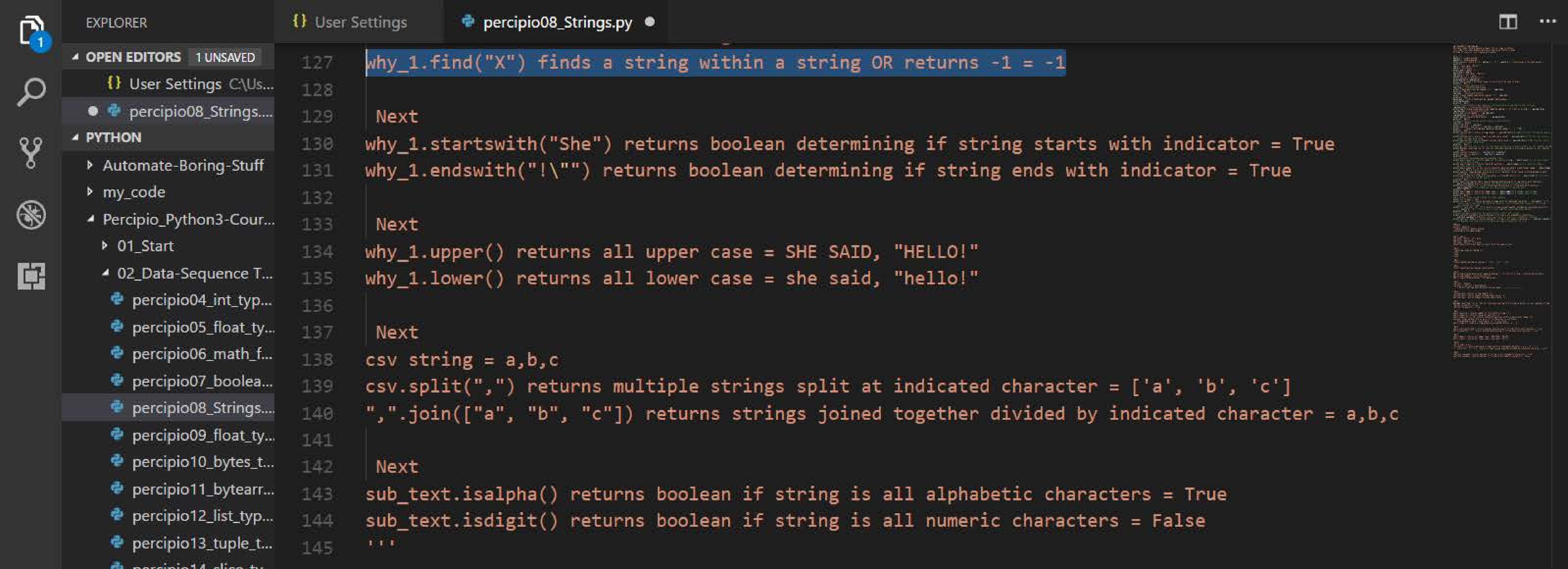
```python
# Sequence objects can use several operators and functions:
sub_text = 'double'
print('sub_text =', sub_text)
print('sub_text + sub_text =', sub_text + sub_text)
print("'_' * 20 will multipy any character by any number =", '_' * 20)
print(nl, 'Next')
print('len(sub_text) returns string length =', len(sub_text)) # Length function telling how long a string is
print('min(sub_text) returns lowest unicode table value =', min(sub_text)) # Min function returns lowest unicode table value
print('max(sub_text) returns highest unicode table value =', max(sub_text)) # Max function returns highest unicode table value
print(nl, 'Next')
# Boolean resulting "in" or "not in" functions testing if a string is within, or not, another string
print('Boolean resulting "in" or "not in" functions testing if a string is within, or not, another string')
print('sub_not in quote_1 =', sub_text not in quote_1)
print('sub_text in quote_2) =', sub_text in quote_2)
print(nl, 'Next')
# String objects have many available methods like:
print('why_1.count("e") returns number of "e"s within string =', why_1.count('e')) # count number of 'e's within string
print('why_1.index("e") returns character position =', why_1.index('e')) # check the index or position within a string
print('why_1.index("e", 3, 18) returns character position within a positional range =', why_1.index('e', 3, 18)) # search at a particular start/stop position
print('Caution: Index method returns an error if character is not found.\n\tFind method is safer returning a "-1" if not found')
print('why_1.find("X") finds a string within a string OR returns -1 =', why_1.find('X')) # Find a string within a string OR returns -1
print(nl, 'Next')
print('why_1.startswith("She") returns boolean determining if string starts with indicator =', why_1.startswith('She')) # Boolean test determining if string starts with
print('why_1.endswith("!\\"") returns boolean determining if string ends with indicator =', why_1.endswith("!\"")) # Boolean test determining if string ends with
print(nl, 'Next')
print('why_1.upper() returns all upper case =', why_1.upper()) # changes case to upper
print('why_1.lower() returns all lower case =', why_1.lower()) # changes case to lower
print(nl, 'Next')
csv = 'a,b,c' # new string created for these examples
```



```
csv = 'a,b,c' # new string created for these examples
print('csv string =', csv)
print('csv.split(",") returns multiple strings split at indicated character =', csv.split(",")) # returns sub-strings split at indicated character, not including that character; csv-string split-method on a comma
print('",".join(["a", "b", "c"]) returns strings joined together divided by indicated character =', ",".join(["a", "b", "c"])) # joins strings together using indicated character ","
print(nl, 'Next')
# Python has many inspection functions such as 'isthis' and 'isthat'
print('sub_text.isalpha() returns boolean if string is all alphabetic characters =', sub_text.isalpha()) # Boolean returning True if string is all alphabetic characters
print('sub_text.isdigit() returns boolean if string is all numeric characters =', sub_text.isdigit()) # Boolean returning True if string is all numeric characters (0 to 9)
'''
RESULTS:
quote_1, quote_2 =
 single quoted or double quoted
resulting in the same output


why_1, why_2 =
She said, "Hello!" It's mine!
why_not_1, why_not_2 =
she said, "Hello!" It's mine!
Above shows 2 different ways to result with the same string

 Next
Create new lines by typing \n =
line1
line2
line3

 Next
Create tabbed spacing by typing \t = col1    col2    col3

 Next
Print 1 backslash by typing 2 backslashes: \

 Next
```



```
 Next
Raw string printed exactly as typed by adding "r" in front of string = line1\nline2\nline3\n
Raw string tabbed = col1\tcol2\tcol3\t
Raw string with backslashes = the backslash: \\

 Next
sub_text = double
sub_text + sub_text = doubledouble
'_' * 20 will multipy any character by any number = ____________________

 Next
len(sub_text) returns string length = 6
min(sub_text) returns lowest unicode table value = b
max(sub_text) returns highest unicode table value = u

 Next
Boolean resulting "in" or "not in" functions testing if a string is within, or not, another string
sub_not in quote_1 = True
sub_text in quote_2) = True

 Next
why_1.count("e") returns number of "e"s within string = 2
why_1.index("e") returns character position = 2
why_1.index("e", 3, 18) returns character position within a positional range = 12
Caution: Index method returns an error if character is not found.
    Find method is safer returning a "-1" if not found
why_1.find("X") finds a string within a string OR returns -1 = -1
```



```
why_1.find("X") finds a string within a string OR returns -1 = -1

 Next
why_1.startswith("She") returns boolean determining if string starts with indicator = True
why_1.endswith("!\"") returns boolean determining if string ends with indicator = True

 Next
why_1.upper() returns all upper case = SHE SAID, "HELLO!"
why_1.lower() returns all lower case = she said, "hello!"

 Next
csv string = a,b,c
csv.split(",") returns multiple strings split at indicated character = ['a', 'b', 'c']
",".join(["a", "b", "c"]) returns strings joined together divided by indicated character = a,b,c

 Next
sub_text.isalpha() returns boolean if string is all alphabetic characters = True
sub_text.isdigit() returns boolean if string is all numeric characters = False
'''
```