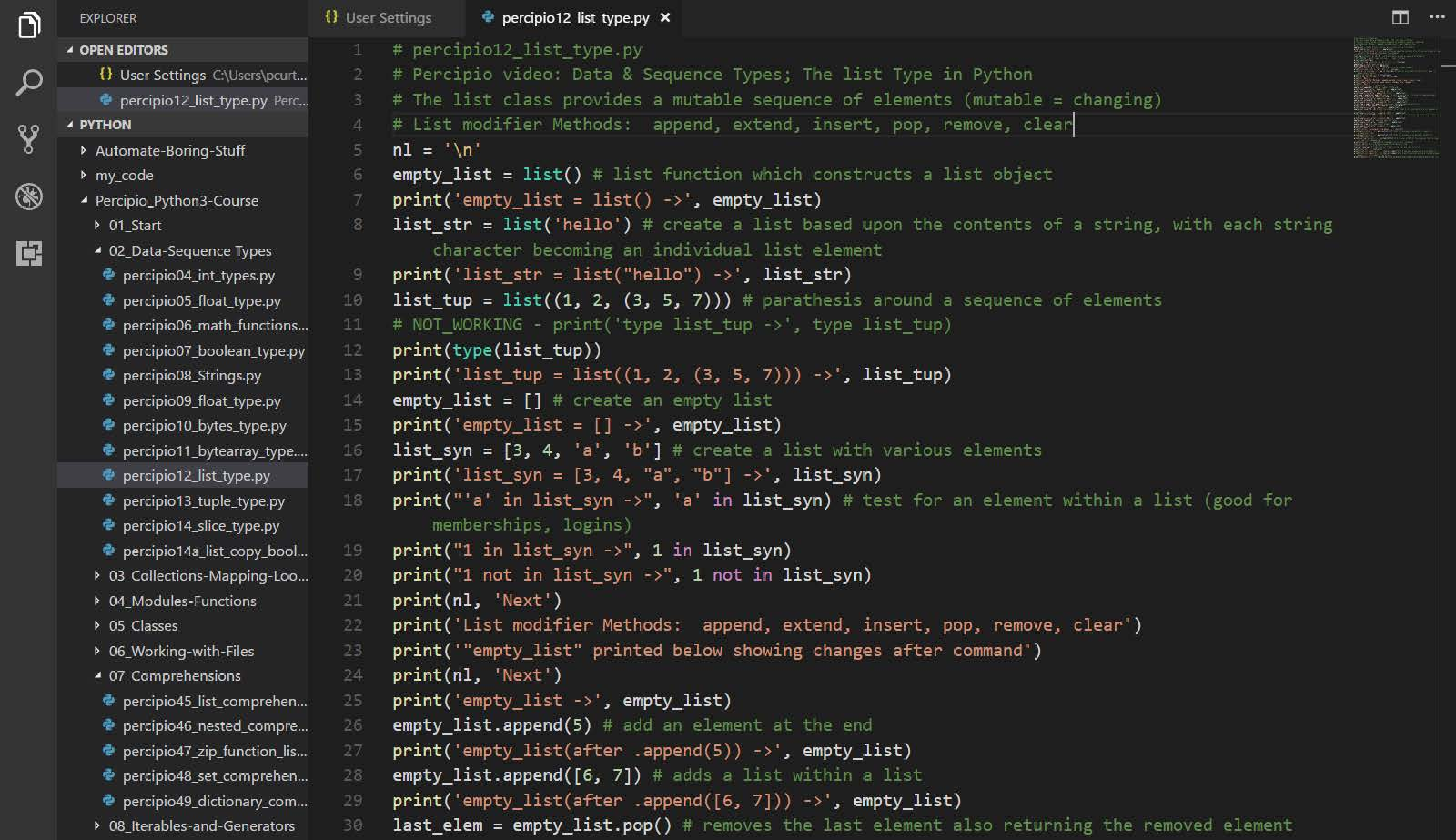
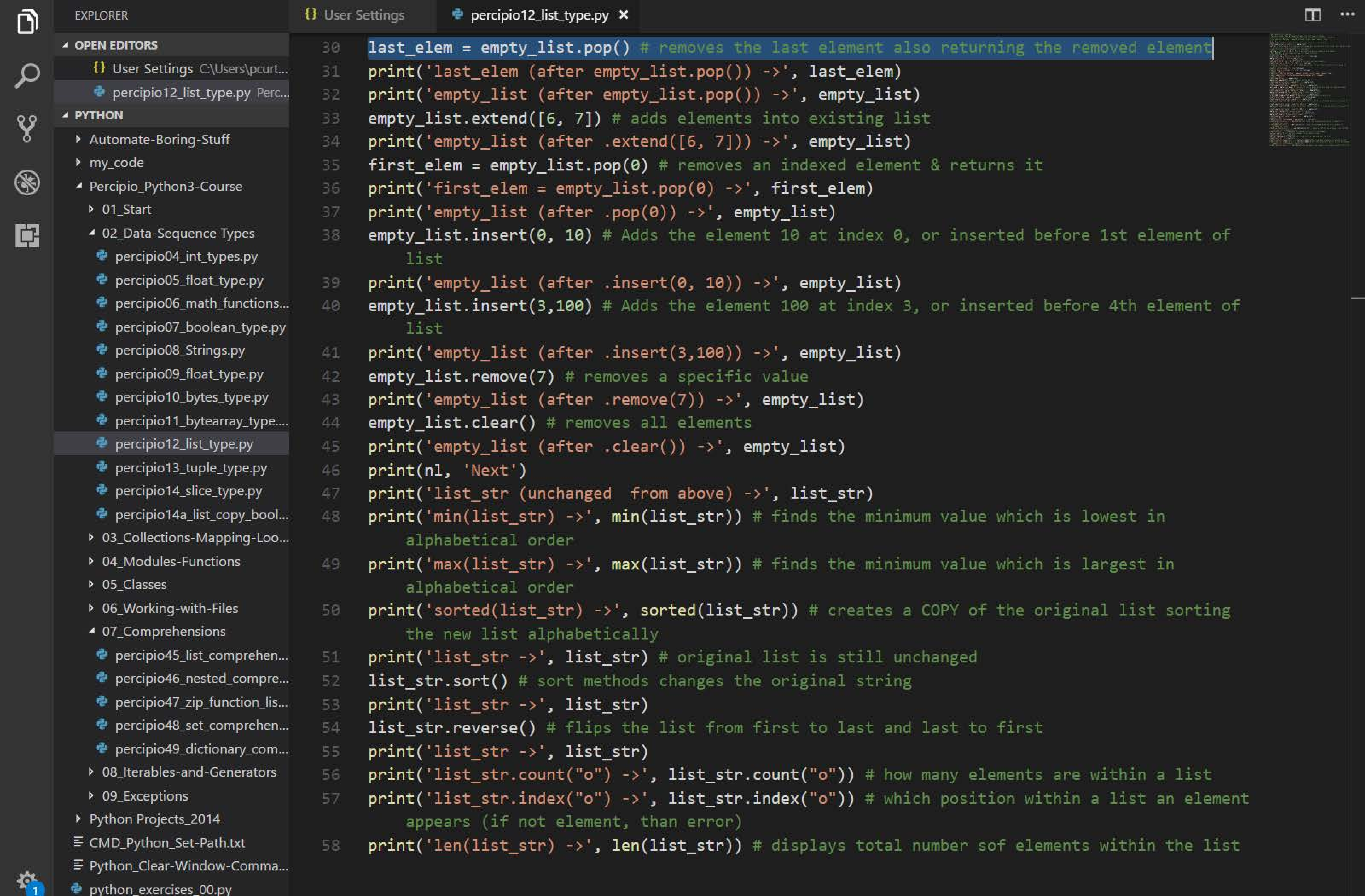

EXPLORER
{} User Settings
percipio12_list_type.py ×
OPEN EDITORS
{} User Settings C:\Users\pcurt...
percipio12_list_type.py Perc...
PYTHON
Automate-Boring-Stuff
my_code
Percipio_Python3-Course
01_Start
02_Data-Sequence Types
percipio04_int_types.py
percipio05_float_type.py
percipio06_math_functions...
percipio07_boolean_type.py
percipio08_Strings.py
percipio09_float_type.py
percipio10_bytes_type.py
percipio11_bytearray_type....
percipio12_list_type.py
percipio13_tuple_type.py
percipio14_slice_type.py
percipio14a_list_copy_bool...
03_Collections-Mapping-Loo...
04_Modules-Functions
05_Classes
06_Working-with-Files
07_Comprehensions
percipio45_list_comprehen...
percipio46_nested_compre...
percipio47_zip_function_lis...
percipio48_set_comprehen...
percipio49_dictionary_com...
08_Iterables-and-Generators

```python
# percipio12_list_type.py
# Percipio video: Data & Sequence Types; The list Type in Python
# The list class provides a mutable sequence of elements (mutable = changing)
# List modifier Methods:  append, extend, insert, pop, remove, clear
nl = '\n'
empty_list = list() # list function which constructs a list object
print('empty_list = list() ->', empty_list)
list_str = list('hello') # create a list based upon the contents of a string, with each string character becoming an individual list element
print('list_str = list("hello") ->', list_str)
list_tup = list((1, 2, (3, 5, 7))) # parathesis around a sequence of elements
# NOT_WORKING - print('type list_tup ->', type list_tup)
print(type(list_tup))
print('list_tup = list((1, 2, (3, 5, 7))) ->', list_tup)
empty_list = [] # create an empty list
print('empty_list = [] ->', empty_list)
list_syn = [3, 4, 'a', 'b'] # create a list with various elements
print('list_syn = [3, 4, "a", "b"] ->', list_syn)
print("'a' in list_syn ->", 'a' in list_syn) # test for an element within a list (good for memberships, logins)
print("1 in list_syn ->", 1 in list_syn)
print("1 not in list_syn ->", 1 not in list_syn)
print(nl, 'Next')
print('List modifier Methods:  append, extend, insert, pop, remove, clear')
print('"empty_list" printed below showing changes after command')
print(nl, 'Next')
print('empty_list ->', empty_list)
empty_list.append(5) # add an element at the end
print('empty_list(after .append(5)) ->', empty_list)
empty_list.append([6, 7]) # adds a list within a list
print('empty_list(after .append([6, 7])) ->', empty_list)
last_elem = empty_list.pop() # removes the last element also returning the removed element
```



```python
last_elem = empty_list.pop() # removes the last element also returning the removed element
print('last_elem (after empty_list.pop()) ->', last_elem)
print('empty_list (after empty_list.pop()) ->', empty_list)
empty_list.extend([6, 7]) # adds elements into existing list
print('empty_list (after .extend([6, 7])) ->', empty_list)
first_elem = empty_list.pop(0) # removes an indexed element & returns it
print('first_elem = empty_list.pop(0) ->', first_elem)
print('empty_list (after .pop(0)) ->', empty_list)
empty_list.insert(0, 10) # Adds the element 10 at index 0, or inserted before 1st element of list
print('empty_list (after .insert(0, 10)) ->', empty_list)
empty_list.insert(3,100) # Adds the element 100 at index 3, or inserted before 4th element of list
print('empty_list (after .insert(3,100)) ->', empty_list)
empty_list.remove(7) # removes a specific value
print('empty_list (after .remove(7)) ->', empty_list)
empty_list.clear() # removes all elements
print('empty_list (after .clear()) ->', empty_list)
print(nl, 'Next')
print('list_str (unchanged  from above) ->', list_str)
print('min(list_str) ->', min(list_str)) # finds the minimum value which is lowest in alphabetical order
print('max(list_str) ->', max(list_str)) # finds the minimum value which is largest in alphabetical order
print('sorted(list_str) ->', sorted(list_str)) # creates a COPY of the original list sorting the new list alphabetically
print('list_str ->', list_str) # original list is still unchanged
list_str.sort() # sort methods changes the original string
print('list_str ->', list_str)
list_str.reverse() # flips the list from first to last and last to first
print('list_str ->', list_str)
print('list_str.count("o") ->', list_str.count("o")) # how many elements are within a list
print('list_str.index("o") ->', list_str.index("o")) # which position within a list an element appears (if not element, than error)
print('len(list_str) ->', len(list_str)) # displays total number sof elements within the list
```