A Collector’s Paradise

A string in Python is a collection of characters. Today, we’ll work with two other types of collections in Python: lists and tuples.

**Part (a):** Most of the string operators we have studied are collection operators—they work on any *iterable* object. Because lists and tuples are iterable, these operators apply also to lists and tuples.

As a warm-up, work with a partner to fill in answers for what will be printed. Assume the following assignments and consult the cheat sheets on lists and tuples for explanations of any operations that you don’t recognize.

```python
a_list = [2, 3.14, 'hi', 63]
a_tuple = ('J', 'P', 'Morgan')
a_list_of_tuples = [('Jan', 1), ('Dec', 25), ('Jul', 4)]

print(a_list[0])

print(a_list_of_tuples[1][\-1])

print(a_list[1:1])

print(a_list[len(a_list):])

print(max(a_tuple))

print(min(a_list_of_tuples))

print(a_tuple[0] < a_tuple[1])

print(3.14 in a_list)

print('Dec' in a_list_of_tuples[1])

print('Dec' in a_list_of_tuples)

print(len(a_tuple))

print(len(a_tuple[-1]))

print("0, 1, 2, 3, 4, 5".split(', '))

print("0, 1, 2, 3, 4, 5".split(' ', '))
```
print("0, 1, 2, 3, 4, 5".split( ' , ' ))  

print('.'.join(a_tuple[::-1]))  

Walk through the visualization at the Visualization 1 link on our website to check your work. Discuss with your partner how Python knows what to print at each step.

**Part (b):** Lists are our first example of a *mutable* type. A list can be changed by indexed assignment and by calling a *mutator method*. Most mutator methods do not return values (i.e., they return None). They are called for their side-effect on the list, which is modified in place.

With your partner, decide what each print statement will print when the program below is executed.

```python
ds_lst = ['do', 're']
n_lst = [6, -3, 5]
n_lst[-1] = 13

print(n_lst)                      
n_lst.extend(s_lst)               
print(n_lst)                      
n_lst.remove('do')                
print(n_lst)                      
s_lst.append(n_lst)               
print(s_lst)                      

#Line 1
del n_lst[-1]

print(n_lst)                      
print(n_lst.sort())               
print(n_lst)                      
print(s_lst)                      

#Line 2
print(s_lst.pop())                
print(s_lst)                      
print(s_lst)                      
```

Step through the execution in the Visualization 2 link to check your understanding. Before each step, discuss how the state will change and how Python determines what to print. **Question:** In going from the line labeled Line 1 to the one labeled Line 2, the value of `s_lst` changes even though none of the instructions between these two lines references `s_lst`. Be sure you can explain why this apparent anomaly occurs.