5. List Methods

append(item)

```
L = [10, 20, 30]
L.append(40)
```
→ L: [10, 20, 30, 40]

insert(i, item)

```
L.insert(3, 35)
```
→ L: [10, 20, 30, 35, 40]

```
L.insert(-2, 32)
```
→ L: [10, 20, 30, 32, 35, 40]

pop() - Remove and return last item in list

```
value = L.pop()
```
→ value:40, L: [10, 20, 30, 32, 35]

pop(i) - Remove and return item at index i

```
value = L.pop(2)
```
→ value: 30, L: [10, 20, 32, 35]

sort()

```
L = [3, 4, 6, 2, 3, 5]
L.sort()
```
→ L: [2, 3, 3, 4, 5, 6]

reverse()

```
L = [2, 3, 3, 4, 5, 6]
L.reverse()
```
→ L: [6, 5, 4, 3, 3, 2]

index(item) - Returns index of first occurrence of item

```
L = ['dead', 'parrot', 'grail', 'parrot']
L.index('parrot')
```
→ Returns: 1

There is no find like with strings

count(item)

```
L
```
→ L: ['dead', 'parrot', 'grail', 'parrot']

```
L.count('parrot')
```
→ Returns: 2

```
L.count('grail')
```
→ Returns: 1

```
L.count('holy')
```
→ Returns: 0

remove(item) - removes first occurrence of item

```
L = [1, 2, 3, 2, 4]
L.remove(2)
```
→ L: [1, 3, 2, 4]

clear() - removes everything from the ist

```
L = [1, 2, 3, 4]
```

`L.clear()` → L: []

---

extend(L2) - L1.extend(L2) ≡ L1 = L1 + L2 ≡ L1 += L2

```
L1 = [3, 4, 5]
L2 = [8, 9]
L1.extend(L2)
```
→ L1: [3, 4, 5, 8, 9], L2: [8, 9]

```
L1 = [3, 4, 5]
L2 = [8, 9]
L1 = L1 + L2
```
→ L1: [3, 4, 5, 8, 9], L2: [8, 9]

```
L1 = [3, 4, 5]
L2 = [8, 9]
L1 += L2
```
→ L1: [3, 4, 5, 8, 9], L2: [8, 9]

6. Built-In List Functions

sorted(collection) - Returns a list

`L = [3, 4, 6, 2, 3, 5]`	
`sorted(L)`	→ Returns: [2, 3, 3, 4, 5, 6]
`L`	→ L is not changed. L: [3, 4, 6, 2, 3, 5]

Works with strings. Returns a list since strings are immutable.

`sorted('iowa')`	→ Returns: ['a', 'i', 'o', 'w']

---

reversed(collection) - Returns a list iterator. We'll convert to list.

`L = [2, 4, 6, 8]`	
`list(reversed(L))`	→ Returns: [8, 6, 4, 2], L: [2, 4, 6, 8]

---

len / max / min / sum

`L = list(range(1, 101))`	→ L: 1 to 100
`len(L)`	→ 100
`max(L)`	→ 100
`min(L)`	→ 1
`sum(L)`	→5050

sum only works on lists of numbers. But

`sum[]`	→ Returns: 0

---

max / min / sorted on list of strings - Uses dictionary order

```
L = ['category', 'cat', 'dog', 'catholic', 'catalog']
sorted(L)
```

→ Returns: ['cat', 'catalog', 'category', 'catholic', 'dog']

`max(L)`	→ 'dog'
`min(L)`	→ 'cat'

---

all - Returns True if every list item is True. Only for list of Booleans

`all([True, True, True])`	→ True
`all([True, False, True])`	→ False
`all([4 < 9, 'at' in 'cat', 4 == abs(4)])`	→ True

---

any - Returns True if any list item is true. Only for list of Booleans

`any([False, True, False])`	→ True
`any((False, False, False))`	→ False
`any([4 == 7, 'x' in 'cat', 4 == abs(-4)])`	→ True
`any([4 == 7, 'x' in 'cat', -4 == abs(-4)])`	→ False

7. Shuffling a List

`import random`
`L = list(range(1, 11))` → L: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

`random.shuffle(L)` → L: [3, 7, 9, 1, 6, 5, 10, 4, 8, 2]

Two more str methods now that we have lists: split() and join()

1. Split

`event = 'Mississippi Blues Festival'`
`event.split()` → Returns: ['Mississippi', 'Blues', 'Festival']
`event` → 'Mississippi Blues Festival' (unchanged)

`river = event.split()[0]` → river: 'Mississippi'

`river.split()` → Returns: ['Mississippi']

`river.split('ss')` → Returns: ['Mi', 'i', 'ippi']

`river.split('i')` → Returns: ['M', 'ss', 'ss', 'pp', '']
Notice the last element is the empty string

Split with multiple assignment:

- Multiple assignment works with a list on RHS
  `x, y, z = [1, 2, 3]` → x: 1, y: 2, z: 3

  `x, y, z = 'python is great'.split()`
  → x: 'python', y: 'is', z: 'great'

2. join - the opposite of split

`'-=-'.join(['cat', 'dog', 'mouse'])` → 'cat-=-dog-=-mouse'

String Separator Must be a list of strings

Common use – Make a string from a list of characters

Randomize a string - Useful since strings are immutable

`import random`
`s = 'lumberjack'`
`L = list(s)` → L: ['l', 'u', 'm', 'b', 'e', 'r', 'j', 'a', 'c', 'k']
`random.shuffle(L)` → L: ['a', 'e', 'l', 'r', 'k', 'c', 'j', 'm', 'b', 'u']
`s = ''.join(L)` → s: 'aelrkcjmbu'