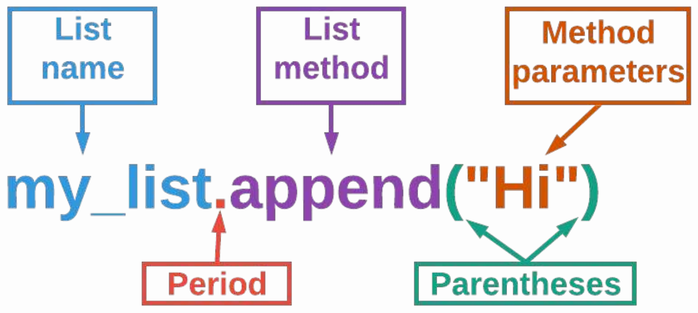
Learning Objectives - List Methods

- **Define a list method**
- **Describe the syntax of a method**
- **Identify some commonly used list methods**

Append

What is a List Method?

Lists have special commands called methods (more on methods in a later lesson). Methods have a special syntax. First, start with a list (often a variable that represents a list). Add a period after the list. Finally, add the name of the method with any parameters. Parameters are values that the method will use.



List Method with Parameters

Translation: Append the string `Hi` to the list `my_list`.

The Append Method

The `append` method adds an element to a list. `append` adds the element to the end of the list.

```
my_list = [1, 2, 3]
new_element = 4

my_list.append(new_element)
print(my_list)
```

challenge

What happens if you:

- Change the value of `new_element` to `"four"`?
- Change the value of `new_element` to `len(my_list)`?
- Change the value of `new_element` to `my_list[0]`?

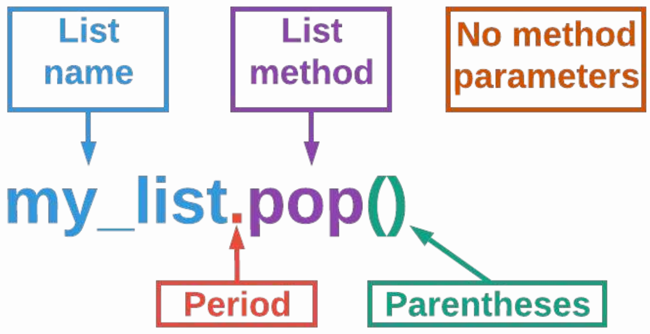
▼ **`append` versus `+`**

There is an important difference between `append` and the concatenation operator (`+`). The `+` operator only combines two lists. The `append` method can add a value of any data type to a list.

Pop

The Pop Method

There are some list methods that do not require parameters. You still must use the parentheses even if there are no parameters. The `pop` method is an example of this. The `pop` method removes and returns the last element of a list.



List Method with No Parameters

Translation: Pop (remove and return) the last element from the list `my_list`.

```
my_list = [1, 2, 3, 4]
print(my_list)
print(my_list.pop())
print(my_list)
```

challenge

What happens if you:

- Change the code to:

```
my_list = [1, 2, 3, 4]
print(my_list)
my_list.pop()
my_list.pop()
my_list.pop()
my_list.pop()
print(my_list)
```

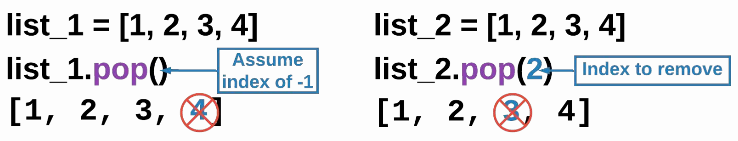
- Add one more `my_list.pop()` before the `print` statement?

▼ **pop versus `my_list[-1]`**

`my_list[-1]` returns the last element in a list. This **does not** modify the original list. The `pop` method also returns the last element of a list, but it **always** modifies the original list. The last element has been removed from the list.

Optional Parameters

The `pop` method has optional parameters. That means if you do not put anything between the parentheses, it will pop off the last element (index of -1) in the list. If you want to remove a different element, put the element's index between the parentheses.



Optional Parameters

```
my_list = [1, 2, 3, 4]
delete = 0
print(my_list.pop(delete))
print(my_list)
```

challenge

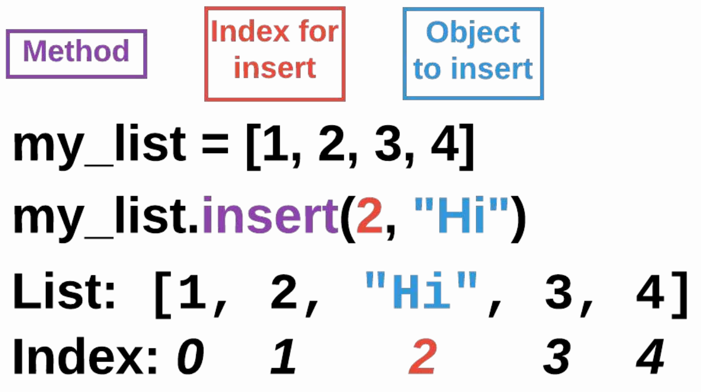
What happens if you:

- Change `delete` to `delete = 2`?
- Change `delete` to `delete = -1`?
- Change `delete` to `delete = 4`?

Insert

The Insert Method

The `insert` method allows you to add any object to an array. This method has two parameters, the index of the insertion and object to be inserted. The order is also import. The index should come first, the object second.



List Insert

▼ **append versus insert**

The `append` method will always add the object to the **end** of the list. The `insert` method gives you the ability to use **any index** you want.

```
my_list = [1, 2, 3, 4]
my_list.insert(2, "Hi")
print(my_list)
```

challenge

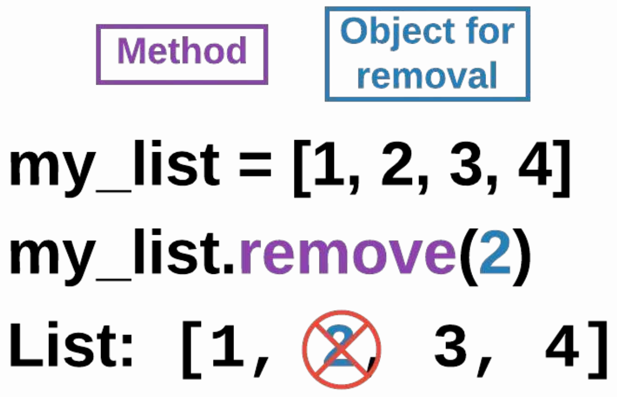
What happens if you:

- Change `insert` to `my_list.insert(3, "Hi")`?
- Change `insert` to `my_list.insert(4, "Hi")`?
- Change `insert` to `my_list.insert("Hi", 1)`?
- Change `insert` to `my_list.insert("Hi")`?

Remove

The Remove Method

The `remove` method has one parameter, the object to be removed from the list.



List Remove

```
my_list = [1, 2, 3, 3, 4]
my_list.remove(2)
print(my_list)
```

challenge

What happens if you:

- Change the `remove` method to `my_list.remove(3)`?
- Change the `remove` method to `my_list.remove(2 *2)`?
- Change the `remove` method to `my_list.remove(0)`

Remove Versus Pop

What is the difference between the `remove` and `pop` methods? They both remove an element from a list, but there are some subtle differences as well.

Pop	Remove
Removes an element	Removes an element
Removes based on index	Removes based on value
Returns the removed value	Does not return anything

```
list_1 = [1, 2, 3, 4, 5]
list_1.pop()
print(list_1)

list_2 = [1, 2, 3, 4, 5]
list_2.remove(5)
print(list_2)
```

challenge

What happens if you:

- Change the `pop` method to `list_1.pop(2)` and change the `remove` method to `list_2.remove(2)`?
- Change the code to be the following:

```
list_1 = [1, 2, 3, 4, 5]
print(list_1.pop())

list_2 = [1, 2, 3, 4, 5]
print(list_2.remove(5))
```

Count

The Count Method

The `count` method will count how many times an element appears in a list. `count` has one parameter, the element you wish to count.

```
my_var = 2
my_list = [2, "red", 2.0, my_var, "Red", 8 // 4]
print(my_list.count(2))
```

challenge

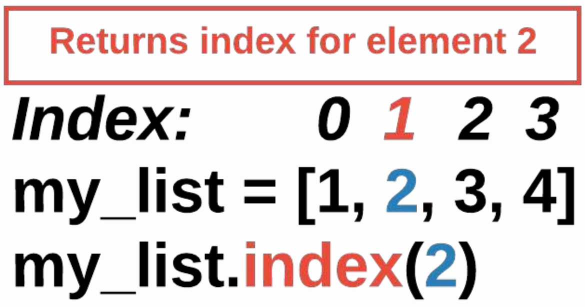
What happens if you:

- Change `2.0` to `2.00000001`?
- Change the `print` statement to `print(my_list.count("red"))`?
- Change the `print` statement to `print(my_list.count(99))`?

Index

The Index Method

The `index` method returns the index of a given element in a list. `index` has one parameter, the element in a list.



Index Method

```
my_list = ["dog", True, 16, "house", 55.9, False, 16]
index = my_list.index("house")
print(index)
```

challenge

What happens if you:

- Change the value of `index` to `my_list.index(False)`?
- Change the value of `index` to `my_list.index(16)`?
- Change the value of `index` to `my_list.index('cat')`?

Sort

The Sort Method

The `sort` method arranges a list in order. If the `sort` method does not have a parameter, then it will sort the list in ascending order. The `sort` method does not return a new list; instead it modifies the original list.

```
my_list = [23, 55, 11, 7, 82.9, -14, 0, 34]
print(my_list)
my_list.sort()
print(my_list)
```

challenge

What happens if you:

- Change the list to `my_list = ["zebra", "door", "apple", "cat", "deer", "bark"]`?
- Change the list to `my_list = [23, 15, "red", 90, -8, False]`?
- Change the list to `my_list = ["APPLE", "apple", "Apple"]`?

Reverse Sort

The `sort` method has an optional parameter to sort a list in descending order. Use `reverse=True` as the parameter to reverse sort a list.

```
my_list = [23, 55, 11, 7, 82.9, -14, 0, 34]
my_list.sort()
print(my_list)
my_list.sort(reverse=True)
print(my_list)
```

Reverse

The Reverse Method

The `reverse` method reverses the order of a list. `reverse` **is not** a reverse sort. It does not have a parameter. The `reverse` method does not return a new list, it modifies the original list.

Before: [3,"cat", 3.14, 48.03, False, "dog", "xyz", True]
After: [True, "xyz", "dog", False, 48.03, 3.14, "cat", 3]

Reverse Method

```
my_list = ["north", True, 45, 12, "red"]
print(my_list)
my_list.reverse()
print(my_list)
```

challenge

What happens if you:

- Change the list to `my_list = [1, 4, 6, 2, 7, 3, 5]`?
- Change the list to `my_list = [1]`?

Formative Assessment 1

Formative Assessment 2