Liezel Cielo D. Tiqui
CS3C

Defining a List:
A list in Python is a group of items enclosed in square brackets, allowing for the storage of multiple values in a single variable.

Example:
```
my_list = [1, 2, 3, 'apple', 'banana']
print(my_list)
# Result:
[1, 2, 3, 'apple', 'banana']
```

List Syntax
Python's syntax for creating a list is straightforward, using square bracket notation to assign values to variables.

Example:
```
element1 = 100
element2 = 'hello world''
element3 = [1, 2, 3, 4, 5]
my_list = [element1, element2, element3]
print(my_list)
# Result:
100, 'hello world', [1, 2, 3, 4, 5]
```

Accessing List Elements
In Python, list indices start at 0 and end at -1, with negative indices counting backwards from the list's end. To access specific elements within a list, use the square bracket notation with the index within.

Example:
```
my_list = [1, 2, 3, 4, 5]
print(my_list[0])
# Result: 1
print(my_list[-1])
# Result: 5
```

Loop through a List
In Python, a loop is a method used to iterate through a list's elements, while a for loop is used to iterate over a list's elements.

Example:
```
my_list = [1, 2, 3, 4, 5]
```

```
for item in my_list:
    print(item)
# Result:
1
2
3
4
5
```

List Length:
Python's len() function is a useful tool for determining the length or number of elements in a list, providing the list's element count.

Example:
```
my_list = [1, 2, 3, 4, 5]
length = len(my_list)
print(length)
# Result:
5
```

Add Items in the List
The append() method is a syntax used to add a new element to the end of a list.

Example:
```
my_list = [1, 2, 3, 4, 5]
my_list.append(6)
print(my_list)
# Result:
[1, 2, 3, 4, 5, 6]
```

Remove Item from a List:
Functions like remove() and pop() can be used to remove an item from a list, with pop() removing the element at its supplied index and delete() removing it at its first occurrence.

Example:
```
my_list = [1, 2, 3, 4, 5]
my_list.remove(3)
print(my_list)
# Result:
[1, 2, 4, 5]
popped_item = my_list.pop(2)
print(my_list)
# Result:
[1, 2, 5]
```

The List () Constructor:
Using the list() constructor in Python, you can make a list. It builds a new list from an iterable object, like a string, tuple, or other list.

Example:
```
my_list = list([1, 2, 3, 4, 5])
print(my_list)
# Result:
[1, 2, 3, 4, 5]
```

List Methods:
Python lists offer numerous built-in methods, including insert(), pop(), delete(), sort(), reverse(), and append(), which enable users to manipulate list elements, add, remove, sort, and alter them according to their needs.

- **insert(index, element):** This function adds the supplied element to the list at the specified index. Moved to the right are already-existing items.
  Example:
  ```
  my_list = [1, 2, 3]
  my_list.insert(1, 4)
  print(my_list)
  # Result:
  [1, 4, 2, 3]
  ```

- **pop(index):** This function takes out the element at the given index and puts it back. The last element in the list is removed and returned if no index is given.
  Example:
  ```
  my_list = [1, 2, 3]
  popped_element = my_list.pop(1)
  print(popped_element)
  # Result:
  2
  print(my_list)
  # Result:
  [1, 3]
  ```

- **remove(element):** This function eliminates the list's initial instance of the given element.
  Example:
  ```
  my_list = [1, 2, 3, 2]
  my_list.remove(2)
  print(my_list)
  # Result:
  [1, 3, 2]
  ```

- **sort():** This function arranges the list's elements in ascending order. It alters the list that was initially in place.
  Example:
  my_list = [3, 1, 4, 1, 5, 9, 2, 6, 5]
  my_list.sort()
  print(my_list)
  # Result:
  [1, 1, 2, 3, 4, 5, 5, 6, 9]

- **reverse():** This function flips the list's elemental order. It also alters the previously established list.
  Example:
  my_list = [1, 2, 3]
  my_list.reverse()
  print(my_list)
  # Result:
  [3, 2, 1]

- **append(element):** This method appends the element to the end of the list.
  Example:
  my_list = [1, 2, 3]
  my_list.append(4)
  print(my_list)
  # Result:
  [1, 2, 3, 4]

Nested Lists:
Python's nested lists feature allows for the creation of multi-dimensional structures with multiple indices for accessing elements within these lists.

Example:
```
nested_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
print(nested_list[0][1])
 # Result:
2
```