1. **What exactly is []?**
   * **[]** represents an empty list in Python.
2. **Assigning 'hello' as the Third Value in a List:**
   * Assuming **spam = [2, 4, 6, 8, 10]**, you can assign 'hello' as the third value with: **spam[2] = 'hello'**.
3. **Value of spam[int(int('3' \* 2) / 11)]:**
   * **spam[int(int('3' \* 2) / 11)]** evaluates to **spam[3]**, so the value is 'd'.
4. **Value of spam[-1]:**
   * **spam[-1]** gives the last element of the list, which is 'd'.
5. **Value of spam[:2]:**
   * **spam[:2]** returns a sublist containing the first two elements: **['a', 'b']**.
6. **Value of bacon.index('cat'):**
   * Assuming **bacon = [3.14, 'cat', 11, 'cat', True]**, **bacon.index('cat')** returns the index of the first occurrence of 'cat', which is 1.
7. **Effect of bacon.append(99) on the List:**
   * **bacon.append(99)** adds 99 to the end of the list, changing it to **[3.14, 'cat', 11, 'cat', True, 99]**.
8. **Effect of bacon.remove('cat') on the List:**
   * **bacon.remove('cat')** removes the first occurrence of 'cat', modifying the list to **[3.14, 11, 'cat', True, 99]**.
9. **List Concatenation and Replication Operators:**
   * Concatenation: **+** (e.g., **list1 + list2**)
   * Replication: **\*** (e.g., **list \* n** repeats the list 'n' times).
10. **Difference Between append() and insert() Methods:**

* **append()** adds an element to the end of the list, while **insert()** inserts an element at a specified index.

1. **Two Methods for Removing Items from a List:**

* **remove()** for removing by value.
* **pop()** for removing by index (or last element if no index specified).

1. **Similarities Between List and String Values:**

* Both can be sliced, concatenated, and have a length.

1. **Difference Between Tuples and Lists:**

* Lists are mutable, tuples are immutable.

1. **Typing a Tuple with Only the Integer 42:**

* **(42,)** (Note the comma).

1. **Getting List Value's Tuple Form and Vice Versa:**

* List to tuple: **tuple(my\_list)**
* Tuple to list: **list(my\_tuple)**

1. **Variables Containing List Values:**

* They contain references to lists, not the actual lists.

1. **Distinguishing copy.copy() and copy.deepcopy():**

* **copy.copy()** creates a shallow copy (references to nested objects are copied).
* **copy.deepcopy()** creates a deep copy (copies all nested objects recursively).