**1. What exactly is []?**

**Answer**:

[] is used to represent list**. List is a value** that contains multiple values in an ordered sequence. List begins with **[** and ends with **]**. Values inside the list are called as **items**.

**2. In a list of values stored in a variable called spam, how would you assign the value 'hello' as the third value? (Assume [2, 4, 6, 8, 10] are in spam.)**

**Answer:**

spam=[2,4,6,8,10]

**spam[2]=’hello’**

**Let's pretend the spam includes the list ['a', 'b', 'c', 'd'] for the next three queries.**

**3. What is the value of spam[int(int('3' \* 2) / 11)]?**

**4. What is the value of spam[-1]?**

**5. What is the value of spam[:2]?**

**Answer:**

|  |  |  |
| --- | --- | --- |
| 3 | spam[int(int('3' \* 2) / 11)] | d |
| 4 | spam[-1] | d |
| 5 | spam[:2] | a b |

**Let's pretend bacon has the list [3.14, 'cat,' 11, 'cat,' True] for the next three questions.**

**6. What is the value of bacon.index('cat')?**

**7. How does bacon.append(99) change the look of the list value in bacon?**

**8. How does bacon.remove('cat') change the look of the list in bacon?**

Answer:

|  |  |  |
| --- | --- | --- |
| 6 | bacon.index('cat') | 1 |
| 7 | bacon.append(99) | [3.14, 'cat,' 11, 'cat', True, 99] |
| 8 | bacon**.remove('cat')** | [3.14, 11,**’cat’**, True, 99]  Only first instance is deleted |

**9. What are the list concatenation and list replication operators?**

**Answer:**

|  |  |
| --- | --- |
| Concatenation | + |
| Replication | \* |

**10. What is difference between the list methods append() and insert()?**

**Answer:**

|  |  |
| --- | --- |
| l.append(99) | **Will add the value to the end of list** |
| l.insert(1,’99’) | **Will insert value at index** 1 of the list |

**11. What are the two methods for removing items from a list?**

**Answer:**

|  |  |
| --- | --- |
| **l.remove(99)** | This will delete the specified value |
| **del(l[i])** | This will delete the value linked to index i |

**12. Describe how list values and string values are identical**.

**Answer**:

|  |
| --- |
| 1. Indexing 2. Slicing 3. Concatenation 4. Replication 5. In and not in operators 6. Len() function to calculate the length |

**13. What's the difference between tuples and lists?**

**Answer:**

|  |  |
| --- | --- |
| **Tuple()** | **List[]** |
| Data Never change i.e Immutable | Data can change- Mutable |
| Less size | More size |
| Append yes | Append yes |
| Count() and index() | Many more |

**14. How do you type a tuple value that only contains the integer 42?**

**Answer:** (42,)

**15. How do you get a list value's tuple form? How do you get a tuple value's list form?**

**Answer**:

tuple(list1)

list(tuple1)

**16. Variables that "contain" list values are not necessarily lists themselves. Instead, what do they contain?**

**Answer:** They contain the **references**

**17. How do you distinguish between copy.copy() and copy.deepcopy()?**

Answer:

|  |  |
| --- | --- |
| **Shallow Copy: copy.copy()** | **Deep Copy : copy.deepcopy()** |
| It is used to make a duplicate copy in case we pass an argument to function in form of **mutable** object. | Now problem arises when the list we want to copy is in the nested format. Here copy.deepcopy() comes handy |
| It creates a **duplicate value as well as reference** of the mutable entity like list or dictionary | **The nested list** as well gets copied which is an extra feature in addition to all the features of shallow copy. |