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

Ans No.1:

[] is an empty list which can be used to store value inside through append or insert method.

**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.)**

Ans No.2:

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

spam.insert(2,"hello")

print(spam)

**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)]?**

Ans No.3:

The above-mentioned statement will print ‘d’ as it will follow the following steps:

1. Will multiply str 3 by 2 which results in 33 as it’s a string.
2. 33 will be type casted from string to int by the int function.
3. 33 will be divided by 11 which is equals to 3.0.
4. 3.0 will be converted into 3 through integer function.
5. Element at 3rd index will be printed from the list spam i.e. d.

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

Ans No.4:

The value of spam[-1] will be ‘d’ as it will extract the element present at the last index.

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

Ans No.5:

This will print only the elements present till 2nd index in the list spam which are ['a', 'b']. The start\_index is not defined due to which it will take the start\_index by default as 0.

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')?**

Ans No.6:

The value of the above mentioned statement will be 1. As it will match the first element (‘cat’) present and give its index.

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

Ans No.7

This will append 99 in the end of the list bacon.

[3.14, 'cat', 11, 'cat', True, 99]

As append method appends the given value at the end of the given list.

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

Ans No.8:

The above mentioned statement will remove the first occurence of ‘cat’. The final list will look like this after printing:

[3.14, 11, 'cat', True]

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

Ans No.9:

The list concatenation operator such as ‘list1+list2’ will combine two lists into a single list. While list replication operators will replicate the existing list as another list such as the copy() method.

Note: In list replication you cannot give new lists to be added as you can in list concatenation.

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

Ans No.10:

The append() method appends an element to the last index while insert() method inserts the element at given index in the list.

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

Ans No.11:

Following are the two methods to remove an item from the list:

1. pop(): It removes the element from the list based on index. If index is not mentioned as argument it removes by default the last value from the list.
2. remove(): It removes the first occurrence of the mentioned element based on the given value.

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

Ans No.12:

1. They both occur in ordered sequence.
2. Lists and string can be sliced and indexed.
3. Both support certain operations:
   1. Concatenation: Using ‘+’ operator to concatenate two items.
   2. Repetition: We can repeat strings or lists using ‘\*’ operator.
   3. Membership: We can check the element’s presence in a string or list.

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

Ans No.13:

1. Tuples are immutable while lists are mutable.
2. Lists are represented by ‘[ ]’ while tuples are represented by ‘( )’.
3. Tuples are faster in performance while lists are slower as tuples cannot be changed and require less space for storage.

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

Ans No.14:

(42,) will be written as a tuple which contains a single element. The comma ( , ) is used to represent it as a tuple with a single element.

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

Ans No.15:

List Value's Tuple Form:

* a = (42,45)

print(f"{a} || {type(a)}")

a = list (a) # Type Casting tuple as a list

print(f"{a} || {type(a)}")

Tuple Value's List Form:

* a = [42,45]

print(f"{a} || {type(a)}")

a = tuple (a) # Type Casting list as a tuple

print(f"{a} || {type(a)}")

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

Ans No.16:

They contain reference to address of the list object in memory rather than the list itself.

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

Ans No.17:

copy.copy() affects the original nested objects. While, copy.deepcopy() will not affect the original nested object instead it will create a separate completely independent copy of the nested as well as top-body objects, so changes made to the deep copy do not affect the original list or any other copies of the list.