1. What exactly is []?

**Ans: []**  represents it’s a list.

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:**

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

spam[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:** ‘d’

Explanation: int(int('3' \* 2) / 11) = 3

As spam[3] = **‘d’**

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

**Ans: ‘d’**

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

**Ans: ‘c’**

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**: 1

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

**Ans**: After appending 99,

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

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

**Ans:** After removal:

bacon = [3.14, 11, 'cat', True]

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

**Ans: List** **concatenation** means joining the list to each other in the given sequence of concatenation. In concatenation we can join different of same string.

Eg: ["hello" , "I"] + ["am" , "Ankita"]🡪 ['hello', 'I', 'am', 'Ankita']

**List Replication** means duplicating the elements in the same list n number of times inside the same list.

Eg: : ["hello" , "I"] \* 3 🡪 ['hello', 'I', 'hello', 'I', 'hello', 'I']

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

**Ans:** append method add the element at the end of the list and insert method add the element before the element of the mentioned index.

Eg l = [3.14, 'cat', 11, 'cat', True]

: **Append**

l.append(5)

🡪[3.14, 'cat', 11, 'cat', True, 5]

**Insert**

l.insert(3,’abc’)

🡪[3.14, 'cat', 11, ‘abc’,'cat', True, 5]

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

**Ans: pop()** and **remove()** are the two methods to remove items from the list.

Pop() remove the last element from the list whereas the remove() removes the first occurrence of the element mentioned from the list.

Eg: l = [3.14, 'cat', 11, 'cat', True]

l.pop()

print(l) 🡪 [3.14, 'cat', 11, 'cat']

l.remove('cat')

print(l) 🡪 [3.14, 11, 'cat']

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

**Ans:** List and String value are iterable. We can perform slicing operations on the string as well as list.

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

**Ans: Tuples** are immutable entity whereas **lists** are mutable entity. Many built in operations can be performed on list compared to tuple. On a tuple we can perform only count and index operation.

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

**Ans:**

l = []

l.append(42)

t = tuple(l)

print(t) 🡪 (42,)

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

**Ans:**

l1 = [3.14, 'cat', 11, 'cat', True]

t = ("abc",1, True)

list(t) 🡪 ["abc",1, True]

tuple(l1) 🡪 (3.14, 'cat', 11, 'cat', True)

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

**Ans:** Variable that hold value of list contains **value of the element**

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

**Ans:** In **Shallow copy,** a new copy is created in which the values in the new copy are reference to the original. So if changes are made in the new copy it gets reflected the original also.

Eg:

import copy

l1 = [1, 2, [3,5], 4]

l2 = copy.copy(l1)

print("Orignal list before shallow copy:",l1)

l2[2][0] = 8

print("Orignal list after shallow copy:",l1)

🡪Orignal list before shallow copy: [1, 2, [3, 5], 4]

Orignal list after shallow copy: [1, 2, [8, 5], 4]

In **Deep copy,** a new copy is created in which the values in the new copy have no refrence to the original. It creates a separate new list. So if changes are made in the new copy it do not get reflected the original.

Eg:

import copy

l1 = [1, 2, [3,5], 4]

l2 = copy.deepcopy(l1)

print("Orignal list before deep copy:",l1)

l2[2][0] = 8

print("Orignal list after deep copy:",l1)

🡪Orignal list before deep copy: [1, 2, [3, 5], 4]

Orignal list after deep copy: [1, 2, [3, 5], 4]