1. What exactly is []?

**Ans:- Square brackets are used to create a list in Python. [] creates an empty 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.insert(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)]?

**Ans:- ‘d’**

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

**Ans:- ‘d’**

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

**Ans:- [‘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')?

**Ans:- 1**

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

**Ans:- This will append 99 in the end of the list and list will be like the below:**

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

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

**Ans:- [3.14, 11, 'cat', True, 99]**

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

**Ans:- List concatenation operators help in creating a new string made from two or more strings together.**

**Ex:**

**fname =’Abhinay’**

**lname=’Bandooni’**

**name = fname+’ ’+lname**

**print(name)**

**List replication operator, on the other hand, does not change the string. Instead, it replicates the string temporarily.**

**Ex:**

**stringpattern =’###-’**

**print(stringpattern\*3)**

**The above code will print stringpattern 3 times without making any changes in variable itself.**

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

**Ans:- append(object) -> adds element in the end of the list**

**insert(index,object)-> adds element in the end of the list if location is not passed along with element itself. Otherwise the element is added in the desired location.**

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

**Ans:- pop(index) -> removes element from the end of the list if index is not passed as argument. Otherwise, removes element from specified index.**

**remove(object) -> removes first occurrence of object from list.**

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

**Ans:- They both can be traversed almost the same way, i.e. using loops. Accessing a character/element in both data types is similar.**

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

**Ans:- Tuples are immutable, whereas lists are mutable. Mutability allows us to change the data value. In an existing list, we can make as many changes as we can. But in tuples, we can’t make any changes. The only alternative we have is to create another tuple using the existing one but the tuple can’t be modified.**

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

**Ans:-**

**tp = (42)**

**print(tp)**

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

**Ans:- Converision between tuples and list is done with tuple() and list().**

**Ex:-**

**tuple1 = (1,2,3,4)**

**list1=list(tuple1) ---------------🡪 tuple converted to list**

**tuple2 = tuple(list1) ----------🡪 list converted to list**

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

**Ans:- Technically, when we create (let’s say) a list and assign it to a variable, we are actually creating a reference to the object that contains values of list. Variable name has the reference to the object only, with the help of which we can access that object and subsequently access it’s values.**

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

**Ans:- The copy.copy() does the shallow copying of compound objects and copy.deepcopy() does deep copying of compound objects.**

**Shallow copy – Here a new compound object will be created and reference to each object inside original compound object will be assigned in new compound object. Any changes made to the objects of compound object will be reflected in original compound object as the newly created object has reference to objects of original compound object.**

**Deep copy – Here a new compound object will be created and copies each object value inside original compound recursively. Any changes made to objects of compound object will not be reflected in original compound object.**

**Shallow copy Ex:**

**import copy**

**l1 = [1,2,3,[22,33],11]**

**l2=copy.copy(l1)**

**l2[3][0]=100**

**print(l1)**

**print(l2)**

**Output:**

**[1, 2, 3, [100, 33], 11]**

**[1, 2, 3, [100, 33], 11]**

**Deep Copy Ex:**

**import copy**

**l1 = [1,2,3,[22,33],11]**

**l2=copy.deepcopy(l1)**

**l1[3][0]=100**

**print(l1)**

**print(l2)**

**Output:**

**[1, 2, 3, [100, 33], 11]**

**[1, 2, 3, [22, 33], 11]**