1. What exactly is []?

**Ans:** [] are the brackets used to define list variables in python. They are used to store list values or tell if the variable is a list or not. For example: mylist=[1,2,3, “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]=”hello”’ in this line of code, the value of spam at index 2 is permanently changing from 6 to hello.

Spam will be: [2,4, “hello”, 8,10]

Or we can say ‘spam.insert(2, “hello”)’ in this statement hello is assigned to the second index of list spam, pushing the original values of spam to one step further.

Now the spam will be: [2,4, “hello”, 6,8,10]

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:** spam=[‘a’, ‘b’, ‘c’, ‘d’]

The value will be ‘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:** bacon list will contain: [3.14, ‘cat’, 11, ‘cat’, 99]

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

**Ans:** bacon list will be: [3.14, 11, ‘cat’, 99]

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

**Ans:** List concatenation is used to add two different lists, therefore, concatenation operator is '+' and list replication is used to duplicate the lists, hence, '*' is used as replication operators.

For example: a=[1,2,3] and b=[a,b,c]

If we say a+b then the output will be: [1,2,3,a,b,c]

If we say a*a then the output will be: [1,2,3,1,2,3] and b*b=[a,b,c,a,b,c]

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

**Ans:** append() method in list is used to add any value at the end of the list, whereas, insert() method is used to add any value to the given index. For example if we add 'new' to the a=[a,b,c] by using the append method we will get a.append("new"), output will be: a=[a,b,c,'new']. If we add 'new' by using the insert method we have to give the index value also. For example: a.insert(2, "new"), output will be: a=[a,b,'new',c]

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

**Ans:** Two methods are pop() and remove().

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

**Ans:** Both list values and string values can be accessed by index values. For instance: `string='hello'`

and `mylist=["h","e","l","l","o"]`

Both string[0] and mylist[0] will be equal.

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

**Ans:** Tuples are immutable whereas lists are mutable, i.e. we cannot change the tuple values unlike lists by indexing. Tuples are defined by using () brackets and lists are defined by [] square brackets. For example: tup=(1,2,4) is a tuple and list=[1,2,4] is a list.

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

**Ans:** tup=(42)

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

**Ans: To convert list to tuple form:**

```
list=[1,2,3]

tup=tuple(list)

tup
```

**Output:** `(1, 2, 3)`

```
To convert tuple to list:

tuple1=(1,2,3)

mylist=list(tuple1)

print(mylist)

Output: [1,2,3]
```

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

**Ans: They contain references to the list values.**

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

Ans: copy.copy() function only creates a shallow copy of the variable. This means, if we change any nested value in the original variable, it will also change the copied variable the same way. Whereas, copy.deepcopy() function creates a deep copied variable. This means, if we change any nested value in the original variable, it will not change the copied variable in any way.