1. What exactly is [ ]?

This defines the **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.)

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)]?

**‘3’\*2 = ‘33’**

**int(‘33’) = 33**

**33/11 = 3**

**int(3) = 3**

**spam[3] = ‘d’ *#(ANSWER)***

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

**spam[-1]** will return **‘d’** as it will count the index from the end of the list.

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

**spam[:2]** will return **‘a’, ‘b’.** It will count the index form starting of the list, like 0,1,2,3,4 but it does not include the extreme limit.

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')?

bacon= [3.14, 'cat', 11, 'cat', True]  
print(bacon.index("cat"))

INDEX= 1

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

**bacon.append(99)** will add the value **99** in the end of the list.

Final list will be **[3.14, 'cat', 11, 'cat', True, 99]**

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

**bacon.remove(‘cat’)** will remove the **“cat”** which is at **INDEX 1** from the list.

The final list will be **[3.14, 11, 'cat', True, 99]**

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

**list concatenation operator = +**

**list replication operator = \***

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

1. l.append(**"VALUE"**) *# ADD VALUE AT THE END OF LIST*
2. l.insert(**"INDEX","VALUE"**) *# TAKES TWO ARGUMENTS, 1ST INDEX AND 2ND VALUE TO INSERT ON THAT INDEX*

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

To remove a value we have two methods

1. l.pop(**"INDEX"**) *# REMOVE VALUE BY ITS INDEX*
2. l.remove(**"VALUE"**) *# REMOVE VALUE BY ITS VALUE*
3. l.clear() *# CLEAR THE LIST*

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

List values and String values are identical in terms of **(both have INDEX, can be SLICED, can be replicate, can be concatenate, can use LEN(), can be use with IN and NOT IN)**

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

|  |  |
| --- | --- |
| **List** | **Tuple** |
| Mutable (Elements can be changed) | Immutable (Elements cannot be changed) |
| Notation: [ ] | Notation: ( ) |
| Store any kind of data (Heterogeneous) | Store any kind of data (Heterogeneous) |
| Keep data in sequence | Keep data in sequence |

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

**t=(42,)**

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

To Get List Value’s Tuple

**l=[42, 43, 3, 2, 6, 86, 32, 45, 78, 3, 2]  
t=tuple(l)**

To Get Tuple Value’s List

**t=(42, 43, 3, 2, 6, 86, 32, 45, 78, 3, 2)  
l=list(t)**

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

They contain **reference values** of the list elements.

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

**copy.copy()** makes a shallow copy of the list that is any change in the list generated bycopy.copy() will also reflect in the original list.

**copy.deepcopy()** makes a duplicate list of the original list. SO that changes in the list generated by **copy.deepcopy()** donot reflect in the original list.