1. What exactly is []? **Defining 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.) **spam[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)]? **'d'**

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

5. What is the value of spam[:2]? **['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')? **1**

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

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

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

[3.14, 11, 'cat', True]; Did not consider the append done above.

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

List oncatenation can be done using '+' operator. If 'a' and 'b' are two lists, the concatenated list can be built using a+b. On the other hand, list replication can be done using '*' operator. If 'a' is a list, we can use a*3 to replicate a three times.

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

append() adds the object at the end of the list, while insert adds it at the provided index

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

remove() and pop()

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

Question is not clear

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

Lists are mutable while tuples are immutable

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

The question is not clear. I consider that type of the tuple object is asked here. The answer is int

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

list value's tuple form:tuple(a) where a is a list

tuple value's list form:list(a) where a is a tuple

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

Question is not clear

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

copy.copy() creates a new object which stores the reference of the original elements, while copy.deepcopy() creates a new object and recursively adds the copies of nested objects present in the original elements

copy.copy():

```
import copy

old_list = [[1, 1, 1], [2, 2, 2], [3, 3, 3]]

new_list = copy.copy(old_list)

old_list[1][1] = 'AA'

print("Old list:", old_list)

print("New list:", new_list)
```

Old list: [[1, 1, 1], [2, 'AA', 2], [3, 3, 3]]

New list: [[1, 1, 1], [2, 'AA', 2], [3, 3, 3]]

copy.deepcopy()

```
import copy
old_list = [[1, 1, 1], [2, 2, 2], [3, 3, 3]]
new_list = copy.deepcopy(old_list)
old_list[1][0] = 'BB'
print("Old list:", old_list)
print("New list:", new_list)
```

Old list: [[1, 1, 1], ['BB', 2, 2], [3, 3, 3]]

New list: [[1, 1, 1], [2, 2, 2], [3, 3, 3]]