1. What exactly is []?

It’s appeal to inner structure of variable, exactly to one of element, like a set, tuple, dictionary

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.)

Let's pretend the spam includes the list ['a', 'b', 'c', 'd'] for the next three queries.

1. 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]

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

Replication – ‘\*’

Concatenation = .join() , ‘+’ , ‘%s %s %s’ % (s1, s2, s3)

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

append() – adds an item to the end of a list

insert() – adds an item to specified position in the list

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

remove(), pop(), del x[i]

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

List and string are same sequences, but list may contain different types, but inner string only char symbols

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

Tuples are immutable, list mutable

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

var = (42)

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

tuple\_var = (41, 42)

list\_var =[41, 42]

from\_tuple\_to\_list = list(tuple\_var)

from\_list\_to\_tuple = tuple(list\_var)

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

It may contains references to object in memory, for example

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

copy.copy() – it’s like a copy only “first level” of object. If we copy list[] addresses of copy this list will be different, but nested object will be the same. But deepcopy will have copied nested object also, they will have different address, not as in initial list.