1. What exactly is []?

Empty list

Eg

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.)

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

spam[2] = 'hello'

3. What is the value of spam[int(int('3' \* 2) / 11)]?

It accesses the fourth element of the list spam , which is d .

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

d (last element of the list)

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

Here start point is 0, end point is 2, therefore output is [‘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')?

It will return 1.

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

It will add the value 99 to the end of the list. Now new list is [3.14, 'cat', 11, 'cat', True, 99]

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

It will remove the first occurrence of the string 'cat' from the list. Now new list is [3.14, 11, 'cat', True, 99]

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

The list concatenation operator is + , It allows us to join together two or more sequence types (list , string, tuples) into a single types., and the list replication (or duplication) operator is \*, It allows us to create a new list ( also other sequence types like string and tuples) by replicating an existing list a certain number of times.

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

append() adds an element to the end of a list. insert() adds an element to a specific position in a list.

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

We can use remove() to remove the first occurrence of that value from the list which passed as an argument . We can also use pop() to remove an element from a list at a specific index position.

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

list values and string values are identical in many ways:

1. Both support concatenation.
2. Both can be indexed and sliced.
3. Both are ordered sequences.

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

There are several key differences between the tuples and list which are as follow:

Mutability: tuples are immutable (once created, we cannot change its content) where as lists are mutable.

Size: we can increase or decrease the size of list , by removing and adding elements in the list, while tuples are fixed in size once created.

Syntax: Tuples are created using parentheses () ,while lists are created using square brackets [].

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

tuple\_example = (23,)

It is important to include the comma , because without comma interpreter will interpret (42) as an integer value enclosed in parentheses, instead of tuple.

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 tuple form , we have to use tuple() ,it takes a list as an argument and returns a tuple.

list\_example = [1, 2, 3, 4, 5]

tuple\_example = tuple(list\_example)

print(tuple\_example) #we get (1,2,3,4,5)

To get tuple value list form , we have to use list() ,it takes a tuple as an argument and returns a list.

tuple\_example = (1, 2, 3, 4, 5)

list\_example = list(tuple\_example)

print(list\_example) #we get [1,2,3,4,5]

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

It contain reference to the location in memory where the list is stored. If we manipulate the list through any one of the variables, the change will be reflected in both variables, because they both reference to the same list object.

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

The difference between copy.copy() and copy.deepcopy() is in the types of copying they both performed.

copy.copy() makes a shallow copy of the object, it creates a new object with same contents as the original object, but contents themselves are not copied instead of this, the new object contains reference to original object contents. If changes made to the contents of the new object will also affect the original object contents.

copy.deepcopy() makes a deep copy of the object, it creates a new object with the same contents as the original object, but contents themselves are also copied, it means if changes made to the contents of the new object do not affect the original object's contents.