1.What exactly is []?

Solution-

The empty list value, which is a list value that contains no items. This is similar to how '' is the empty string value.

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.)

Solution-

```
In [10]: spam = [2,4,6,8,10]

In [11]: spam[2]='hello'

In [12]: spam

Out[12]: [2, 4, 'hello', 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)]?

```
In [15]: spam[int(int('3' * 2) / 11)]

Out[15]: 'd'
```

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

```
In [16]: spam[-1]

Out[16]: 'd'
```

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

```
In [18]: spam[:2]

Out[18]: ['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')?

```
In [23]: bacon = [3.14, 'cat', 11, 'cat', True]
```

```
In [25]: bacon.index('cat')
```

```
Out[25]: 1
```

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

```
In [26]: bacon.append(99)
```

```
In [27]: bacon
```

```
Out[27]: [3.14, 'cat', 11, 'cat', True, 99]
```

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

```
In [28]: bacon.remove('cat')
```

```
In [29]: bacon
```

```
Out[29]: [3.14, 11, 'cat', True, 99]
```

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

Solution-

The operator for list concatenation is +, while the operator for replication is *. (This is the same as for strings.)

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

Solution-

The only difference between append() and insert() is that **insert function allows us to add a specific element at a specified index of the list unlike append() where we can add the element only at end of the list.**

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

Solution-

Remove and POP methods are used for removal of the item.

```
In [37]: bacon
Out[37]: [11, 22, 3.14, 11, 'cat', True, 99, 11]

In [38]: bacon.remove(22)

In [39]: bacon
Out[39]: [11, 3.14, 11, 'cat', True, 99, 11]

In [40]: bacon.pop(2)
Out[40]: 11

In [41]: bacon
Out[41]: [11, 3.14, 'cat', True, 99, 11]
```

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

Solution-

The similarity between Lists and Strings in Python is that both are sequences. The differences between them are that firstly, Lists are mutable but Strings are immutable. Secondly, elements of a list can be of different types whereas a String only contains characters that are all of String type.

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

Solution-

The key difference between the tuples and lists is that while the tuples are immutable objects the lists are mutable.

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

```
In [53]: a=[42]

In [55]: a=tuple(a)

In [56]: a

Out[56]: (42,)

In [57]: type(a)

Out[57]: tuple
```

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

```
In [1]: a=[1,2,3,4,5,6]

In [7]: print("type of a is",type(a))
        print("Converting to tuple --->",tuple(a))

        print("Converting to list --->",list(tuple(a)))

type of a is <class 'list'>
Converting to tuple ---> (1, 2, 3, 4, 5, 6)
Converting to list ---> [1, 2, 3, 4, 5, 6]
```

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

Solution-

Variables will contain references to list` values rather than list values themselves. But for strings and integer values, variables simply contain the string or integer value.

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

Solution-

Copy() create reference to original object. If you change copied object - you change the original object. . deepcopy() creates new object and does real copying of original object to new one.