1. What exactly is []?
It is 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.)
Program:
```
spam = [2, 4, 6, 8, 10]
spam.insert(2,'hello')
Spam
```
Output:
```
[2, 4, 'hello', 6, 8, 10]
```
In above program, since we have to insert the value hello as third one, then index number will be 2, so in insert , 2 is mentioned as the index.

Let's pretend the spam includes the list ['a', 'b', 'c', 'd'] for the next three queries.
If we include the list mentioned above, then spam will become like below
```
spam.append(['a', 'b', 'c', 'd'])
Spam
```
Output:
```
[2, 4, 'hello', 6, 8, 10, ['a', 'b', 'c', 'd']]
```
3. What is the value of spam[int(int('3' * 2) / 11)]?
Answer: 6
4. What is the value of spam[-1]?
Answer: ['a', 'b', 'c', 'd']
5. What is the value of spam[:2]?
Answer: [2, 4]

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')?
Answer : 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, 99]

9. What are the list concatenation and list replication operators?
+ is the list concatenation operator
* is the list replication operator
Example:
```
def listfunction():
   l1 = ['Hi']
   l2 = l1 +l1
   l3 = l1*3
```

```
    return l2,l3
```
Output: (//Output in Tuples)

```
listfunction()
(['Hi', 'Hi'], ['Hi', 'Hi', 'Hi'])
```

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

List.append() - will add the mentioned value at the end of the list.

List.insert()- will insert the value at the mentioned index position.

Example:

```
list1 = ['a',12,'hi',19]
list1.append(23)
List1
```
Output:

```
['a', 12, 'hi', 19, 23]
```
Example:

```
list1.insert(2,67)
List1
```
Output:

```
['a', 12, 67, 'hi', 19, 23]
```

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

Pop() and remove() methods are used to remove the data from the list.

Pop()- removes based on index. By default it removes the –1 index value. But if we give index number it will remove that particular value residing in that index

Remove()- actually removes the first occurance of the mentioned value.

Example:

```
List1 = ['a', 12, 67, 'hi', 19, 23]
list1.pop()
Output: 23
List1
```
Output:

```
['a', 12, 67, 'hi', 19]
```
Example:

```
list1.remove(12)
List1
```
Output:

```
['a', 67, 'hi', 19]
```

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

String is immutable object. Lists are mutable. But they both can be iterated in loops. Can be used in index and slicing operation. Can be concatenated and replicated. But in string once created, we can not replace, delete any character with original object(Immutable).

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

Tuples – It is a immutable object ....But Lists are Mutable object
Items in Tuples can not be replaced or changed once created, which is called immutablitiy.
Lists can be altered even after creation. Which is called Mutablility.

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

Answer:

```
 a = (42,)
type(a)
tuple
```

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

Example:

```
list1=['hello',67,'art',56]  // This is a list
list1 = tuple(list1)    // Converting list into tuple form
list1   // Checking the tuple form of the list
```

Output:

```
('hello', 67, 'art', 56)
```

Example:

```
tuple1 = ('hello',67,'art',56) // This is a tuple
tuple1 = list(tuple1) // converting tuple into list form
tuple1 // checking the list form of tuple
```

Output:

```
['hello', 67, 'art', 56]
```

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

They contain reference to that values.

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

Copy.copy() is a shallow copy, which creates a object and stores the reference of the original object when comes to independent items. But Shallow copy does not create copy of nested objects instead it just copies the reference of nested objects. So any change in the nested object in shallow copy, will be reflected in the original also. But independent item change will not be reflected in original object.

Copy.deepcopy()

A deep copy creates a new object and recursively adds the copies of nested objects which is present in the original object. Here any change in the indenpendent or nested object items will not change the original object because of different memory location.