In [ ]:
```
1. What exactly is []?
```

In [ ]:
```
[] denotes to List or we can simply sayn that it is a empty list because it doesn't hold any values.
l1=[]
```

In [ ]:
```
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.)
```

In [3]:
```
spam=[2,4,6,8,10]
spam.insert(3,"hello")
spam
```

Out[3]:
```
[2, 4, 6, 'hello', 8, 10]
```

In [ ]:
```
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 [ ]:
```
Here,mind that '3' is a string and it will replicate 2 times and becomes 33 and then 33/11 returns 3 and then int(3).
So now the value of spam is 'd'.
```

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

In [ ]:
```
It returns 'd' because -1 is a negative index so it starts from backward.
```

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

In [6]:
```
spam=['a','b','c','d']
b=spam[:2]
b
```

Out[6]:
```
['a', 'b']
```

In [ ]:
```
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 [8]:
```
bacon=[3.14,'cat',11,'cat',True]
b1=bacon.index('cat')
```

```
b1
```

Out[8]:
```
1
```

In [ ]:
```
7. How does bacon.append(99) change the look of the list value in bacon?
```

In [6]:
```
Append function in python is used to add the given value at the last position.
bacon=[3.14,'cat',11,'cat',True]
bacon.append(99)
bacon
```

Out[6]:
```
[3.14, 'cat', 11, 'cat', True, 99]
```

In [ ]:
```
8. How does bacon.remove('cat') change the look of the list in bacon?
```

In [7]:
```
bacon=[3.14,'cat',11,'cat',True]
bacon.remove('cat')
bacon
```

Out[7]:
```
[3.14, 11, 'cat', True]
```

In [ ]:
```
9. What are the list concatenation and list replication operators?
```

In [10]:
```
Basically,list concatenation(+) is used to concat two list and
list replication(*) is used to replicate the given list the times given.
l1=[1,2,3]
l2=[4,5,6]
res=l1+l2
res
```

Out[10]:
```
[1, 2, 3, 4, 5, 6]
```

In [12]:
```
l3=[1,2,3]
res1=l3*2
res1
```

Out[12]:
```
[1, 2, 3, 1, 2, 3]
```

In [ ]:
```
10. What is difference between the list methods append() and insert()?
```

In [ ]:
```
append() is used to append the given element in the last position of the list.
insert() is used to insert the given element by the index number.
```

In [ ]:
```
11. What are the two methods for removing items from a list?
```

In [ ]:
```
remove()-It takes directly the element to be removed.
pop()-It takes the index number to remove the element.
```

In [ ]:
```
12. Describe how list values and string values are identical.
```

In [ ]:
```
The main similarity between Strings and List is that in Python both are sequences.
A string is a character's sequence between single or double quotations.
A list is an item sequence in which each item could be anything such as a float, an integer, a string, etc
```

In [ ]:
```
13. What's the difference between tuples and lists?
```

In [ ]:
```
List is an object in python that is used to store values.
It uses [].
It is mutable.

Tuple is an object in python is also used to store values.
It used ().
It is an immutable.
```

In [ ]:
```
14. How do you type a tuple value that only contains the integer 42?
```

In [14]:
```
t1=(42,)
t1
```

Out[14]:
```
(42,)
```

In [ ]:
```
15. How do you get a list value's tuple form? How do you get a tuple value's list form?
```

In [15]:
```
l1=[20,21,22,23,24,25]
res=tuple(l1)
res
```

Out[15]:
```
(20, 21, 22, 23, 24, 25)
```

In [17]:
```
t1=(20,21,22,23,24,25)
res1=list(t1)
res1
```

Out[17]:
```
[20, 21, 22, 23, 24, 25]
```

In [ ]:
```
16. Variables that "contain" list values are not necessarily lists themselves. Instead, what do they
contain?
```

In [ ]:
```
They contain references to list values.
```

In [ ]:
```
17. How do you distinguish between copy.copy() and copy.deepcopy()?
```

In [ ]:
```
The copy.copy() function will do a shallow copy of a list, while the copy.deepcopy() function will do a deep copy of a list.
That is, only copy.deepcopy() will duplicate any lists inside the list.
```