```
In [ ]: 1. What exactly is []?
        Ans : - [ ] is a empty list, like a =[ ]
```

```
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
        Let's pretend the spam includes the list ['a','b','c','d'] for the next thr
```

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

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

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

```
In [13]: spam=['a','b','c','d']
         spam[int(int('3' * 2) / 11)]

Out[13]: 'd'
```

```
In [15]: #4. What is the value of spam[-1]?
         spam[-1]

Out[15]: 'd'
```

```
In [16]: #5. What is the value of spam[:2]?
         spam[:2]

Out[16]: ['a', 'b']
```

```
In [28]: bacon=[3.14, 'cat', 11, 'cat', True]
```

```
In [21]: #6. What is the value of bacon.index('cat')?
         bacon.index('cat')

Out[21]: 1
```

```
In [30]: #7. How does bacon.append(99) change the look of the list value in bacon?
         bacon.append(99)
         bacon

Out[30]: [3.14, 'cat', 11, 'cat', True, 99]
```

```
In [32]: #8 How does bacon.remove('cat') change the look of the list in bacon?
         bacon.remove('cat')
         bacon

Out[32]: [3.14, 11, True, 99]
```

```
In [ ]: #9 What are the list concatenation and list replication operators?
        # ( * ) is list replication operator ( + ) is list concatination operator
```

```
In [38]: l1=[1,2]
```

```
In [34]: #list concatenation
```

```
l1+l1
```

Out[34]:
```
[1, 2, 1, 2]
```

In [35]:
```
#list replication operators
l1*2
```

Out[35]:
```
[1, 2, 1, 2]
```

In [ ]:
```
#10. What is difference between the list methods append() and insert()?
Anas. append() Appends object to the end of the list
insert() Insert object before index
```

In [39]:
```
l1.append("A")
l1
```

Out[39]:
```
[1, 2, 'A']
```

In [42]:
```
l1.insert(1,"B")
l1
```

Out[42]:
```
[1, 'B', 2, 'A']
```

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

In [43]:
```
l1.remove("B")
l1
```

Out[43]:
```
[1, 2, 'A']
```

In [44]:
```
l1.pop()
l1
```

Out[44]:
```
[1, 2]
```

In [ ]:
```
#12 Describe how list values and string values are identical.
Both lists and strings can be passed to len()
Have indexes and slices
Can be used in for loops
Can be concatenated or replicated
Can be used with the in and not in operators
```

In [ ]:
```
#13 What's the difference between tuples and lists?
Lists : are mutable - they can have values added, removed, or changed. lists
Tuples : are immutable; they cannot be changed at all. Tuples are written u
```

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

In [47]:
```
t1=(14,)
t1
```

Out[47]:
```
(14,)
```

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

In [53]:
```
t=(1,2,"ajeet")
t
```

```
Out[53]: (1, 2, 'ajeet')
```

```
In [54]: l=list(t)
         l
```

```
Out[54]: [1, 2, 'ajeet']
```

```
In [55]: t=tuple(l)
         t
```

```
Out[55]: (1, 2, 'ajeet')
```

```
In [ ]: #16 Variables that "contain" list values are not necessarily lists themselv
        Ans. They contain references to list values
```

```
In [ ]: #17. How do you distinguish between copy.copy() and copy.deepcopy()?
        Ans . The copy.copy() function will do a shallow copy of a list,
        The copy.deepcopy() function will do a deep copy of a list. only copy.deepc
```