1. What exactly is []?

[] is an empty list. List value can be found inside [].

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.)

spam.insert(2, "hello")

```
In [1]:  1 spam = [2,4,6,8,10]
         2 spam

Out[1]: [2, 4, 6, 8, 10]

In [2]:  1 spam.insert(2, "hello")
         2 spam

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

In [ ]:  1
```

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)]?

Value of above expression is - 'd'

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

Value of spam[-1] is- 'd'

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

Value of spam[-1] is- ['a', 'b']

```
In [1]:  1 spam = ['a','b','c','d']
         2 spam

Out[1]: ['a', 'b', 'c', 'd']

In [2]:  1 spam[int(int('3'*2)/11)]

Out[2]: 'd'

In [5]:  1 spam[:2]

Out[5]: ['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')?

Value of bacon.index('cat') is- 1

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

99 will be added to the list in the end.
List of bacon becomes = [3.14, 'cat', 11, 'cat', True, 99]

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

First 'cat' will be removed from the list.

List of bacon becomes= [3.14, 11, 'cat', True, 99]

```
In [4]:  1 bacon = [3.14, 'cat', 11, 'cat', True]
         2 bacon
Out[4]: [3.14, 'cat', 11, 'cat', True]

In [5]:  1 bacon.index('cat')
Out[5]: 1

In [6]:  1 bacon.append(99)
         2 bacon
Out[6]: [3.14, 'cat', 11, 'cat', True, 99]

In [7]:  1 bacon.remove('cat')
         2 bacon
Out[7]: [3.14, 11, 'cat', True, 99]
```

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

Operator for the list concatenation is +

Operator for the list replication is *

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

append() method adds the value at the last position of the list

while insert() method adds the value at the specified index position

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

remove(), pop() are the two methods for removing items from a list.

And clear() removes all the data from a list.

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

list and string both are sequences.

Difference between them are, lists can be mutable but strings are immutable.

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?

Lists are mutable and tuples are immutable.

Some operations can work on lists, but not on tuples.

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

(42,)

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

To get list values in tuple form = tuple(list)

To get tuple values in list form = list(tuple)

In [27]:
```
t1=(5,6,8,9)
t2=list(t1)
print(t2)
print(type(t1))
print(type(t2))
```

```
[5, 6, 8, 9]
<class 'tuple'>
<class 'list'>
```

In [28]:
```
l1=[5,6,7,8]
l2=tuple(l1)
print(l2)
print(type(l1))
print(type(l2))
```

```
(5, 6, 7, 8)
<class 'list'>
<class 'tuple'>
```

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

They can contain integers, floats, lists, tuples, all of these.

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

copy.copy() returns shallow copy.

copy.deepcopy() returns deep copy.

The difference between shallow and deep copying is only relevant for compound objects (objects that contain other objects, like lists or class instances):

- **A *shallow copy* constructs a new compound object and then (to the extent possible) inserts *references* into it to the objects found in the original.**
- **A *deep copy* constructs a new compound object and then, recursively, inserts *copies* into it of the objects found in the original.**