1. What exactly is []?

[ ] exactly refers to 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.)

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

Out[1]:
```
[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 [2]:
```
spam=['a','b','c','d']
spam[int(int('3'*2)/11)]  # it gives the value of third index of list spam which is 'd
```

Out[2]:
```
'd'
```

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

In [9]:
```
spam[-1]  # the value of spam[-1] is 'd'
```

Out[9]:
```
'd'
```

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

In [11]:
```
spam[:2]  # the value of spam[-1] is ['a','b']
```

Out[11]:
```
['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 [17]:
```
bacon =  [3.14, 'cat', 11, 'cat', True]
bacon.index('cat')
#  The index() method in Python returns the index of the first occurrence of the speci
```

Out[17]:
```
1
```

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

In [18]:
```
bacon.append(99)
bacon
```

Out[18]: [3.14, 'cat', 11, 'cat', True, 99]

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

In [23]:
```
bacon.remove('cat')
bacon
# the remove() function is a list method used to remove the first occurrence of a spec
```

Out[23]: [3.14, 11, 'cat', True]

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

'+' is a list concatenation operator and '*' is a list replication operator.

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

List method append() adds an element at the end of the list while, insert() method can be used to insert a value at any desired position. It takes two arguments-element and the index at which the element has to be inserted. Egs. of each as follows:

In [30]:
```
a=[1,2,3,4,5]
a.append(6)
a
```

Out[30]: [1, 2, 3, 4, 5, 6]

In [31]:
```
a.insert(2,7)
a
```

Out[31]: [1, 2, 7, 3, 4, 5, 6]

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

del and pop() are the two methods of removing items from a list.

In [32]:
```
a=[1,2,3,4]
a.pop(0)
a
```

Out[32]: [2, 3, 4]

In [33]:
```
del a[2]
a
```

Out[33]: [2, 3]

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

In [34]:
```
#The list values and string values are identical because both are sequence of values.
a=[1,2,3,4]
b='ineuron'
print(a[2])
print(b[2])
```

```
3
e
```

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

The difference between tuples and list is that lists are mutable and [ ] are used for list to store the data. Whereas tuples are immutable and ( ) are used for tuples to store the data.

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

In [37]:
```
tup = (42,)  # comma after the value is necessary to make single value a tuple
type(tup)
```

Out[37]:
```
tuple
```

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

In [40]:
```
# to get a list value's tuple form we can simply use tuple().
a = [2,3,4,5]
tuple(a)
```

Out[40]:
```
(2, 3, 4, 5)
```

In [41]:
```
# to get a tuple value's list form we can simply use list().
b = (4,5,6)
list(b)
```

Out[41]:
```
[4, 5, 6]
```

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

Variables that contain list values are not necessarily lists themselves, instead they contain reference to a list.

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

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.