'''1. What exactly is []? this is an empty list.'''

In [27]:

```
'''2. In a list of values stored in a variable called spam, how would you assign the valu
e 'hello' as the
third value? (Assume [2, 4, 6, 8, 10] are in spam.)'''

spam=[2, 4, 6, 8, 10]

spam[2]='hello'

print(spam)
```

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

In [2]:

```
#Let's pretend the spam includes the list ['a', 'b', 'c', 'd'] for the next three queries
.

spam=['a', 'b', 'c', 'd']
#3. What is the value of spam[int(int('3' * 2) / 11)]?

spam=[int(int('3' * 2) / 11)]
print(spam)
```

```
[3]
```

In [5]:

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

spam=['a', 'b', 'c', 'd']
spam[-1]
```

Out[5]:

```
'd'
```

In [6]:

```
#What is the value of spam[:2]?

spam=['a', 'b', 'c', 'd']

spam[:2]
```

Out[6]:

```
['a', 'b']
```

In [ ]:

```

```

In [9]:

```
#Let's pretend bacon has the list [3.14,'cat', 11, 'cat', True] for the next three questi
ons.

#6. What is the value of bacon.index('cat')?

bacon=[3.14,'cat', 11, 'cat', True]

bacon.index('cat')
```

Out[9]:

```
1
```

In [11]:

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

bacon=[3.14,'cat', 11, 'cat', True]
bacon.append(99)
print(bacon)
```

```
[3.14, 'cat', 11, 'cat', True, 99]
```

In [13]:

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

bacon=[3.14,'cat', 11, 'cat', True]

bacon.remove('cat')

print(bacon)
```

```
[3.14, 11, 'cat', True]
```

In [ ]:

```

```

#9. What are the list concatenation and list replication operators? #In Python, you can use the' + 'operator for list concatenation and the' * 'operator for list replication.

In [ ]:

```

```

In [24]:

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

''' append() adds an element to the end of the list, while insert()
allows you to specify the position (index) at which
you want to insert an element in the list'''

list1=[1,2,3,]
list2=[1,2,3,5]

list1.append(4)
list2.insert(1,6)

print(list1)
print(list2)
```

```
[1, 2, 3, 4]
[1, 6, 2, 3, 5]
```

In [ ]:

```

```

#11. What are the two methods for removing items from a list? #remove() #pop() #this are the two methods basically to use remove an element from the list.

In [ ]:

```

```

#12. Describe how list values and string values are identical. '''The main similarity between Strings and List is that in Python both are sequences. they share some similarities like indexing slicing itretion also Both lists and strings have a len() function.'''

In [28]:

```
my_list = [1, 2, 3]
my_tuple = (1, 2, 3)
```

```
for item in my_list:
    print(item)

for item in my_tuple:
    print(item)
```

```
1
2
3
1
2
3
```

#13. What's the difference between tuples and lists? '''syntex: list[] tuple() >lists are mutable >tuples are immutable >tuples are faster than lists for accessing the elements >Lists tend to occupy slightly more memory than tuples'''

In [32]:

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

my_tuple = (42,)
```

In [ ]:

```

```

In [4]:

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

#list value turning into a tuple form

list1 = [1,2,3,4,5]
my_tuple = tuple(list1)
print(my_tuple)

#converting tuple into a list

my_tuple = (1, 2, 3)
my_list = list(my_tuple)
print(my_list)  # Output: [1, 2, 3]
```

```
(1, 2, 3, 4, 5)
[1, 2, 3]
```

In [ ]:

```

```

'''16. Variables that ''contain'' list values are not necessarily lists themselves. Instead, what do they contain? Variables that "contain" list values in Python actually contain references or pointers to the list in memory, rather than the list itself.'''

In [ ]:

```

```

'''17. How do you distinguish between copy.copy() and copy.deepcopy()? >copy.copy() creates a shallow copy that shares references to nested objects with the original. > copy.deepcopy() creates a deep copy that duplicates all nested objects