1. What exactly is []?

It represents an empty list. A list is a data structure that can hold an ordered collection of elements.They serve as a starting point for creating and initializing an empty list.

1. 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 [17]:
```
# We can use insert function to specifically add an element to an index here is an example given for the same
import pandas as pd
import numpy as np
spam=[2,4,6,8,10]
spam.insert(2,'hello')
```

In [18]:
```
spam
```

Out[18]:
```
[2, 4, 'hello', 6, 8, 10]
```

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

list ['a', 'b', 'c', 'd']

Let`s break it down

1. int('3'*2) = multiplying string '3' with 2 gives a result 33 and int(33)=33
2. int(33/11)= int(3)
3. list[3]= d thus the answer is d.

In [22]:
```
spam=['a','b','c','d']
spam[int(int('3' * 2) / 11)]
```

Out[22]:
```
'd'
```

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

spam[-1] will give the last element of the list as result

In [30]:
```
spam=['a','b','c','d']
spam[-1]
```

Out[30]:
```
'd'
```

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

spam[:2] would include all elements from the beginning of the list up to, but not including, index 2.

In [31]:
```
spam[:2]
```

Out[31]:
```
['a', 'b']
```

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

It gives the first occurance of the string 'cat' in the given list

In [35]:
```
bacon=[3.14, 'cat', 11, 'cat', True]
bacon.index('cat')
```

Out[35]:
```
1
```

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

It will append the given value 99 at the end of the list

In [36]:
```
bacon.append(99)
```

In [37]:
```
bacon
```

Out[37]:
```
[3.14, 'cat', 11, 'cat', True, 99]
```

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

It removes the first instance of the string 'cat' occurance in the list

In [41]:
```
bacon=[3.14, 'cat', 11, 'cat', True]
```

In [42]:
```
bacon.remove('cat')
bacon
```

Out[42]:
```
[3.14, 11, 'cat', True]
```

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

In Python, the list concatenation operator is the plus sign (+), and the list replication operator is the asterisk (*).

In [46]:
```
#Here is the execution
# here two lists are added sequentially as shown in the result
list1=[1,2,3]
list2=[4,5,6]
list1+list2
```

Out[46]:
```
[1, 2, 3, 4, 5, 6]
```

In [47]:
```
#here list is replicating 3 times as shown in the result
list1*3
```

Out[47]:
```
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

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

append() - it takes single argument 'value' and appends at the end of the list insert() - takes two arguments 1-index 2-value

In [48]:
```
list1.append('cat')
```

In [49]:
```
list1
```

Out[49]:
```
[1, 2, 3, 'cat']
```

In [50]:
```
list1.insert(1,'dog')
```

In [51]:
```
list1
```

Out[51]:
```
[1, 'dog', 2, 3, 'cat']
```

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

remove() & pop() are two methods used to remove items from the list

remove()- takes one argument as value and removes it from the list pop()- will remove the item from the end of the list

In [52]:
```
list2.pop()
```

Out[52]: 6

In [56]:
```
list2.remove(4)
```

In [57]:
```
list2
```

Out[57]: [5]

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

Lists and Strings have identical features. 1.Sequential Data- both are sequential datatypes 2.Indexing- both of them will support indexing which allows accessing individual elements based on their position 3.Slicing- both list and string support slicing 4.Iteration- both can be used in for loop to iterate through each element.

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

Tuples and list are both data structures in python.

1.tuples are immutable means their elements can `t be changed after creation.Once the tuple is created you can` t add,remove or modify elements. List are mutable. and can be modified. 2.Tuples are typically used to represent a collection of related items, where as list can hold different types of elements

1. Tuples are generally light weighted interms of memory and faster to create than list. Lists need more memory
2. Tuples have fewers methods like count(), index() since they are immutable. on the otherhand list have several built-in methods like append(),insert(),remove() etc.

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

In [84]:
```
my_tuple=(42)
```

In [85]:
```
type(my_tuple)
```

Out[85]:
```
int
```

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

In [86]:
```
list=[1,2,3,4]
tuple=(10,20,34,56,78)
```

In [ ]:
```
#to convert list values to tuple form we can use tuple()
list=[11,22,33,44]

tuple(list)
```

In [ ]:
```
my_tuple=(1,2,4,6,7)
my_list=list(my_tuple)
```

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

In Python, variables are essentially labels or references to objects in memory.

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

copy.copy() creates a shallow copy that shares references to nested objects, while copy.deepcopy() creates a deep copy with completely independent copies of both the top-level object and its nested objects.

In [96]:
```
import copy
original_list=[1,2,[3,4],5]
shallow_copy_list=copy.copy(original_list)
```

```
deep_copy_list=copy.deepcopy(original_list)

original_list[3]=99
print(original_list)
print(shallow_copy_list)
print(deep_copy_list)
```

```
[1, 2, [3, 4], 99]
[1, 2, [3, 4], 5]
[1, 2, [3, 4], 5]
```

In [ ]: