1.What exactly is []?

Ans. [] are third (square) brackets which in python are used to define and create lists.

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.)

Ans. spam.insert(2,"hello")

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)]?

Ans. ‘d’

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

Ans.’d’

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

Ans. [‘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')?

Ans. 1

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

Ans. It becomes [3.14, 'cat', 11, 'cat', True,99]

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

Ans. [3.14, 11, 'cat', True,99]

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

Ans.

list concatenation operator is + which concats multiple list elements into a single list. It combines different lists.

e.g. = list1,list2 = [9,8,7],[0,9,6]

list3= list1+list2

print(list3}

#output [9,8,7,0,9,6]

List replication operator \* replicates the same list elements the number of times as specified. it makes a copy of the same elements a number of times as specified.

E.g. – list1=[9,8,7]\*2

#output [9,8,7,9,8,7]

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

Ans. key difference is the position of the element to be added to the list.

append() method always adds a specified element to the end of the list as append takes only one argument i.e. the element to add to the list.

Insert() method takes two arguments , first argument is the index in which the element is to be added to and second argument is the element which is to be inserted.

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

Ans. two main methods are remove() and pop() and there is one more called clear():

remove() method removes the first occurrence of the specified element in the method .

pop() method removes the element from the specified index in the method & returns the list and if index is not specified then removes the last element of the list and returns the list.

clear( ) method clears full list and removes all the elements at once.

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

Ans. list values and string values are identical by the way they are accessed through indexing and slicing operations.

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

Ans.

key difference between list and tuples are list is a mutable object due to unhashable property and tuple is an immutable object due to hasable property.

list cannot be used as key in dictionaries and tuples can be used as keys in dictionaries due to hashable property.

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

Ans. r = (42,)

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

Ans.

tuple1 = (9,6,7,8)

list1 = list(tuple1)

list1 = [9,7,8,6]

tuple1 = tuple(list1)

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

Ans.

Lists are themselves objects and variables are just mere names or references of objects created. Variables that contains list values are just mere references of those list objects memory locations. Several variables can refer to the same list object at once. And if one reference is modified the modification is reflected to all the references and the main memory location.

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

Ans. Copy module has these two functions:

copy.copy() creates a shallow copy of the object and copy.deepcopy() creates a deep copy of an object .

in .copy() it creates a new object but does not create new objects of the elements inside the object. Rather it just references the object elements . These copies gets modifed as soon as the original object gets modified as they are the references of the elements of the original objects.

And in .deepcopy() even the new compound object is created and alongwith the object elements copies are also created and they are just not mere references of the objects inside the compound object. Even if the original copy gets modified the deepcopy remains as it is.