1. What exactly is []?

Answer: [] is a list type

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.)

Answer: spam[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)]?

Answer: d

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

Answer: d

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

Answer:[‘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’)?

Answer: 1

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

Answer: [3.14, ‘cat’, 11, ‘cat’, True, 99]

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

Answer: [3.14, 11, ‘cat’, True, 99]

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

Answer: list concatenation operation is + and list replication operator is \*

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

Answer: append() add the value in the end of the list but insert() can add the value anywhere in the list.

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

Answer: remove list method and del statement

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

Answer: List and string both can have the indexes and slices and both used in loops, in and not in operator and be concatenated and replicated.

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

Answer: List[] is mutable whereas tuple() is immutable, means list can be changed and tuple cannot be changed.

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

Answer: (42)

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

Answer:

List value to tuple form:

lst=[20,30,”Python”]

t1=tuple(lst)

Output: (20,30,”Python)

Tuple value’s list form:

t2= (2,4,6,”Python”)

lst1=list(t2)

Output: [2,4,6,”Python”]

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

Answer: Variable contain references to list values

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

Answer:

Copy.copy() is a shallow copy which creates a new object which stores the reference of the original elements. Shallow copy process does not recurse or creates a nested copies.

For example:

Copying a list using copy()

import copy

old\_list=[[1,2],[3,4],[5,6]]

new\_list=copy.copy(old\_list)

Print(“Old List:”,old\_list)

Print(“New List:”, new\_list)

Output:

Old List: [[1,2],[3,4],[5,6]]

New List: [[1,2],[3,4],[5,6]]

If we add ‘A’ in old list

old\_list=[[1,2],[3,4],[5,6]]

new\_list=copy.copy(old\_list)

old\_list[1][1]=’A’

Print(“Old List:”,old\_list)

Print(“New List:”, new\_list)

Output:

Old List: [[1,2],[3,’A’],[5,6]]

New List: [[1,2],[3,’A’],[5,6]]

Copy.deepcopy() is deep copy. It creates a copy which creates a new object and It only recursively added the copies of nested objects present in the original elements.

For example:

Copying a list using deepcopy()

import copy

old\_list=[[1,2],[3,4],[5,6]]

new\_list=copy.deepcopy(old\_list)

Print(“Old List:”,old\_list)

Print(“New List:”, new\_list)

Output:

Old List: [[1,2],[3,4],[5,6]]

New List: [[1,2],[3,4],[5,6]]

If we add ‘A’ in old list

old\_list=[[1,2],[3,4],[5,6]]

new\_list=copy.deepcopy(old\_list)

old\_list[1][1]=’A’

Print(“Old List:”,old\_list)

Print(“New List:”, new\_list)

Output:

Old List: [[1,2],[3,’A’],[5,6]]

New List: [[1,2],[3,4],[5,6]]