1. What exactly is []?

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.)

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)]?

spam[int(int('3' \* 2) / 11)] = spam[int(int('33') / 11)] = spam[int(33 / 11)]

= spam[3] = ‘d’

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

spam[-1] = ‘d’

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

spam[:2] = ['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')?

bacon.index('cat') = 1

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

99 gets appended at the end of the list

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

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

first occurrence of ‘cat’ gets removed from the list

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

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

List concatenation:

+ operator : list1 = list1 + list2

extend function : list1.extend(list2)

\* operator: result = [\*list1, \*list2] (this works like pointers in C)

List Replication:

\* operator: result :

result = a \* 3 (this replicates list a 3 times)

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

append() : adds item to end of list

eg: spam. append(item)

insert() : adds item at specific index of the list

eg: spam. insert(index, item)

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

remove() method : removes first occurrence of an element from list

spam= ["a", "b", "c", “d”, “b”, “c”, “a”]

spam.remove("c")

print(spam) gives spam= ["a", "b", “d”, “b”, “c”, “a”]

pop() method: removes specified index. if index is not specified it removes last element.

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

spam.pop(1 )

print(spam) gives spam= ['a','c','d','c','b','a']

spam.pop( )

print(spam) gives spam= ['a','b','c','d','c','b']

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

Both Lists & Strings are sequences. Lists are mutable string is immutable. Lists can contain elements of different types but String has only character elements.

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

tuples are immutable, lists are mutable

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

(42,)

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

list to tuple: using tuple method and passing list as argument

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

t = tuple(l)

t = ('a','b','c','d')

tuple to list: using list method and passing tuple as argument

t = ('a','b','c','d')

l = list(t)

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

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

reference to the list values

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

copy.copy() is shallow copy. where reference of nested objects is copied. changing original list changes new list.

old\_list = [[‘a’,’a’,’a’], [‘b’,’b’,’b’], [‘c’,’c’,’c’]]

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

old\_list .append([‘d’,’d’,’d’])

old\_list[1] = [‘z’,’z’,’z’]

old\_list = [[‘a’,’a’,’a’], [‘z’,’z’,’z’], [‘c’,’c’,’c’],[‘d’,’d’,’d’]]

new\_list = [[‘a’,’a’,’a’], [‘z’,’z’,’z’], [‘c’,’c’,’c’]]

copy.deepcopy() is deep copy. where nested objects is copied recursively. changing original list does not change new list. old list and new list are independent of each other.

old\_list = [[‘a’,’a’,’a’], [‘b’,’b’,’b’], [‘c’,’c’,’c’]]

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

old\_list .append([‘d’,’d’,’d’])

old\_list[1] = [‘z’,’z’,’z’]

old\_list = [[‘a’,’a’,’a’], [‘z’,’z’,’z’], [‘c’,’c’,’c’],[‘d’,’d’,’d’]]

new\_list = [[‘a’,’a’,’a’], [‘b’,’b’,’b’], [‘c’,’c’,’c’]]