1. [] represents an empty list and this is used to initialize an empty list e.g. l1 = [].

2. According to question, spam = [2,4,6,8,10], to assign hello as 3rd value which is of index 2, code is spam[2] = “hello”

3. spam[int(int(‘3’ \* 2) / 11)] = spam[int(int((‘33’)/11)] = spam[3] = 4th element of index 3 of list spam which is ‘d’ if spam = [‘a’,’b’,’c’,’d’].

4.spam[-1] is last value of list spam and it is = ‘d’.

5. spam[:2] means values of index 0,1 i.e. ‘a’,’b’

6.now bacon = [3.14,’cat’,11,’cat’,True], so bacon.index(‘cat’) is 1.

7.bacon.append(99) makes bacon as [3.14,’cat’,11,’cat’,True,99]

8.bacon.remove(‘cat’) makes it [3.14,11,’cat’,True,99] as by default 1st cat removed.

9. string concatenation uses + operator which combines strings concatenated together. E.g. ‘Sudhanshu’+’Kumar’ = ‘SudhanshuKumar’ whereas replication which uses \* operator concatenates the string with itself as many times as the no. with which it is being multiplied. E.g. ‘Sudhanshu’\*2 = ‘SudhanshuSudhanshu’.

10. insert() takes index at which new element is to be inserted e.g. bacon.append(2,’HI’) makes list as [3.14,11,’HI’,’cat’,True,99] whereas append by default inserts new element at the right of last index i.e. size of list also increases.

11. remove() and pop() where remove() takes element which is to be deleted as argument and it deletes that element passed whereas pop() deletes the last rightmost element always.

12. list values and string values are identical bcoz both of them can be accessed by same way of indexing i.e. 0 to n-1 from left and operations like concatenation, replication and many others are same for both.

13.Lists are mutable i.e. we can add and remove elements from lists and make changes in the original list which is located at a unique address in main memory whereas tuples are immutable and we can’t make any changes in original tuple located at a unique address in main memory.

14. (42)

15. we can achieve this by typecasting i.e. changing datatype where we pass any list value inside inbuilt function named tuple() or any tuple value into list() function.

16. They just contain references to the list values instead of actual lists stored at a unique address in main memory.

17. .copy() create reference to original object. If you change copied object - you change the original object. .deepcopy() creates new object and does real copying of original object to new one. Changing new deepcopied object doesn't affect original object.