Python Basics Assignment 4

1. What exactly is [ ]?

Ans: The empty list represented by [] is a list that contains no items. This is similar to '' which represents an empty string

```
# Python program to declare
# empty list

# list is declared
a = []

print("Values of a:", a)
print("Type of a:", type(a))
print("Size of a:", len(a))
```

```
Values of a: []
Type of a: <class 'list'>
Size of a: 0
```

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[2]='hello' (Note: Lists follows zero based indexing)

```
# Example
spam=[2,4,6,8,10]
print(spam)
spam[2]='hello' #List uses zero based indexing
print(spam)
```

```
[2, 4, 6, 8, 10]
[2, 4, 'hello', 8, 10]
```

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' ('3' * 2 is the string '33', which is passed to int() before being divided by 11. This eventually evaluates to 3, spam[3] is equal to d.)

```
spam=['a','b','c','d']
print("spam[int(int('3'*2)//11)] ->",spam[int(int('3'*2)//11)])
```

```
spam[int(int('3'*2)//11)] -> d
```

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

Ans: 'd' (Lists support Negative indexing, Hence spam[-1] returs 'd'). -1 index returns last item of the list

```
spam=['a','b','c','d']
print('spam[-1] -> ',spam[-1])
```

```
spam[-1] ->  d
```

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

Ans: spam[:2] returns all elements in the list spam from 0 to 2 excluding 2

```
print(spam)
print(spam[:2])
```

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

Let's pretend bacon has the list [3.14,'cat',11,'cat',True] for the next three question

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

Ans: The value of bacon.index('cat') is 1 (Index method returns the index of first occuerence of 'cat')

```
bacon=[3.14,'cat',11,'cat',True]
print("bacon.index('cat') ->",bacon.index('cat'))
```

```
bacon.index('cat') -> 1
```

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

Ans: The append method adds new elements to the end of the list

```
# Example
print(bacon)
bacon.append(99) # appends 99 to the end of the list
print(bacon)
```

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

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

Ans: The remove method removes the first occurence of the element in the list

```
print(bacon)
bacon.remove('cat')
print(bacon)
```

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

9. what are the list concatenation and list replication operations?

Ans: The operator for list concatenation is +, while the operator for replication is *. (This is the same as for strings.)

```
# Example
list_1 = ['ML','DL','AI','CV','NLP']
list_2 = ['RNN','CNN','SVN']
print(list_1 + list_2) # List Concatenation
print(list_2*2) # List Replication
```

```
['ML', 'DL', 'AI', 'CV', 'NLP', 'RNN', 'CNN', 'SVN']
['RNN', 'CNN', 'SVN', 'RNN', 'CNN', 'SVN']
```

10. what is the difference between the list method append() and insert()?

Ans: While append() will add values only to the end of a list, insert() can add them anywhere in the list.

```
#Examples
list = [1,2,3,4,5]
list.append(100)
print(list)
list.insert(2,'iNeuron')
print(list)
```

```
[1, 2, 3, 4, 5, 100]
[1, 2, 'iNeuron', 3, 4, 5, 100]
```

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

Ans: The del statement and the remove() method are two ways to remove values from a list

```
# assign list
numbers = [1, 2, 3, 2, 3, 4, 5]

# use del
del numbers[2]

numbers
```

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

```
# assign list
numbers1 = [1, 2, 3, 2, 3, 4, 5]

# use remove()
numbers1.remove(3)
numbers1
```

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

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

Ans: Below are some reasons how list values and string values are identical:

1. Both lists and strings can be passed to len() function
2. Have indexes and slices, be used in for loops,
3. Can be concatenated or replicated
4. Can be used with the in and not in operators.

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

Ans: Lists are Mutable, Indexable and Slicable. they can have values added, removed, or changed. Tuples are Immutable but Indexable and Slicable. the tuple values cannot be changed at all. Also, tuples are represented using parentheses, (), while lists use the square brackets, [].

```
list_num = [1,2,3,4]
tup_num = (1,2,3,4)

print(list_num)
print(tup_num)
```

```
[1, 2, 3, 4]
(1, 2, 3, 4)
```

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

Ans:(42,) (The trailing comma is mandatory. otherwise its considered as a int by python Interpreter)

```
tup1=(42)
tup2=(42,)
print(type(tup1))
print(type(tup2))
```

```
<class 'int'>
<class 'tuple'>
```

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

Ans: The tuple() and list() functions, respectively are used to convert a list to tuple and vice versa

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

Ans: They contain references to list values.

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

Ans: The copy.copy() function will do a shallow copy of a list, while the copy.deepcopy() function will do a deep copy of a list. That is, only copy.deepcopy() will duplicate any lists inside the list.

```
# importing copy module
import copy

# initializing list 1
li1 = [1, 2, [3,5], 4]


# using copy for shallow copy
```

```
li2 = copy.copy(li1)

# using deepcopy for deepcopy
li3 = copy.deepcopy(li1)
```