Assignment 4:

## Q1

In Python, [] represents an empty list. A list is a collection of items enclosed in square brackets [] and separated by commas. when there are no items between the brackets, it denotes an empty list.

## Q2

spam = [2, 4, 6, 8, 10]

# Insert 'hello' as the third value in spam

spam.insert(2, 'hello')

# Print spam

print(spam)

## Q3

The expression int(int('3' \* 2) / 11) evaluates to 3. This is because int('3' \* 2) first converts the string '3' to the integer 3, and then int(3 / 11) divides 3 by 11 and returns the integer 3.

The expression spam[3] then accesses the third element of the list spam, which is the value 'd'.

Therefore, the value of spam[int(int('3' \* 2) / 11)] is 'd'.

## Q4

The value of spam[-1] is 'd'.

The expression spam[-1] accesses the last element of the list spam. Since the list spam only has four elements, the expression spam[-1] will return the value 'd'.

## Q5

The value of spam[:2] is ['a', 'b'].

The expression spam[:2] returns a slice of the list spam starting from the first element and ending at the second element (not including the second element). Since the list spam only has four elements, the expression spam[:2] will return a list with the first two elements of the spam list.

## Q6

The value of bacon.index('cat') is 1.

The index() method of a list returns the index of the first occurrence of the specified value in the list. In this case, the first occurrence of the value 'cat' in the list bacon is at index 1.

## Q7

The append() method of a list adds the specified value to the end of the list. In this case, the append() method will add the value 99 to the end of the list bacon.

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

## Q8

The remove() method of a list removes the first occurrence of the specified value from the list. In this case, the remove() method will remove the first occurrence of the value 'cat' from the list bacon.

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

## Q9

The list concatenation and list replication operators in Python are the + and \* operators, respectively.

The + operator concatenates two lists, creating a new list that contains the elements from both lists.

The \* operator replicates a list, creating a new list that contains the specified number of copies of the original list.

## Q10

**append() method:**

The append() method is used to add an element at the end of a list. It takes a single argument, which is the value to be added, and appends that value as a new element to the list.

**insert() method:**

The insert() method is used to insert an element at a specific index within the list. It takes two arguments: the index where the element should be inserted and the value to be inserted.

## Q11

**remove() method:**

The remove() method is used to remove the first occurrence of a specified value from a list. It takes a single argument, which is the value to be removed.

**pop() method:**

The pop() method is used to remove an item from a list at a specific index.

## Q12

List values and string values are identical in the following ways:

They are both sequences, meaning that they can be indexed, sliced, concatenated and iterated over.

They can both be used in for loops.

They can both be used in conditional statements.

## Q13

Tuples and lists are both sequence data types in Python, but they have some key differences.

**Mutability**

The main difference between tuples and lists is that tuples are immutable, while lists are mutable. This means that the elements of a tuple cannot be changed after it is created, while the elements of a list can be changed.

**Syntax:**

Tuples are defined using parentheses () or without any brackets, separating the elements with commas. Lists, on the other hand, are defined using square brackets [ ].

## Q14

To type a tuple value that only contains the integer 42, we can use the following code:

tuple\_value = (42,)

## Q15

To get a list value's tuple form, we can use the tuple() function. The tuple() function takes an iterable, such as a list, and returns a tuple containing the elements of the iterable.

To get a tuple value's list form, we can use the list() function. The list() function takes an iterable, such as a tuple, and returns a list containing the elements of the iterable.

## Q16

Variables that "contain" list values in Python do not actually contain the lists themselves. Instead, they contain references to the lists.

In Python, variables are essentially labels or names that refer to objects in memory. When we assign a list to a variable, the variable holds a reference to the memory location where the list is stored. This means that the variable points to the list rather than directly containing the list itself.

## Q17

The copy.copy() and copy.deepcopy() functions are both used to copy objects in Python. However, they work in different ways.

copy.copy() creates a shallow copy of an object. This means that the new object is a reference to the original object. If we change the original object, the change will be reflected in the new object.

copy.deepcopy() creates a deep copy of an object. This means that the new object is a complete copy of the original object. If we change the original object, the change will not be reflected in the new object.