CS118 Exercises
Arrays 1

For each of the following tasks, create a Python program to accomplish it.

1. A "list" in Python is a variable which can hold more than one value. They are defined by the use of square brackets:

```
L = [1, 2, 3, 4, 5];
```

· Ask the user to provide you with five positive numbers and store each in its own variable.

· After the values have been received, have your program copy them into a variable named `L` just like the line of code above, but using the five variables instead of the constants shown.

2. To reference a vector, we use an *index value*. Prepare a Python program: create a five-element list, `L2`. Multiply the values at positions one, two, and four of the list `L2` and display the result.

3. *Slices* are a portion of an array and are commonly obtained by using the *range operator* (**:**). *Augmentation* is the process of making an array larger by appending another array. Python uses the + operator to *concatenate* two lists together. Write a Python program that creates creates a five-element list, `L`, and from `L` creates two variables `x` and `y` which get their values from the odd and even index positions of `L`, respectively using slices only! [Remember that 0 is considered even] Concatenate `x` and `y` to create a new list, `z`. Create a new variable `L3` that is prepared by augmenting `L` with `z`. Display `L3`.

4. *Diminution* makes an array smaller by eliminating elements. Write a Python program that, using a loop, creates a 5 element list, `L4`, of integers that lie between 10 and 20 by using the random module's `randint()` function. Then have the program eliminate the value at index 2 by using the list method `pop()`. Next eliminate the value at index 4. Answer this with a comment in your program: Why does an error occur?

5. Make an empty list, `L5`. Write nested loops that collect 6 values from the user – one at a time. The inner loop should collect three values, storing all in a single list, `M`. When that loop completes, append `M` onto `L5` and redefine `M` to be the empty list before continuing with the outer loop. After both loops are complete, display `L5`.

6. Make an empty list, `L6`. Write a loop that collects 6 values from the user, one at a time, appending each value onto `L6` using the `append()` method of lists. Display `L6` when it is complete.

7. Create a Python program: First, create a five-element list, `L`. Next, create an *alias* to `L`, named `LA`. Show that `L` and `LA` refer to the same list by appending a new value to `LA`, then displaying `L`. Use the range operator to create `L7` as a duplicate of `L` and not an alias. Demonstrate that `L7` is distinct from `L` by appending -99 to `L7` and displaying both `L` and `L7`.

8. Write a Python program that creates a five-element list, `L`. Sort `L` by applying the list method `sort()`. Apply the `random` module's `shuffle` function to a duplicate (not an alias) `L8`, of `L`. Display both `L` and `L8`.

9. Write a Python program that creates a five-element list, `L`. Sort `L` by applying the list method `sort()`. Apply the list method `reverse()` to a duplicate, `L9`, of `L`. Display both `L` and `L9`.

10. The `set()` function converts a list into a set (another Python data type) and can be used to remove duplicate elements. But a good programmer should know how to do this by hand. Write a Python program that creates a list, `L`, with many duplicate values. Create a new list, `L10`, that contains only unique elements of `L` but do not use any `set`-oriented functions. Display both `L` and `L9`. [HINT: use the `not in` operator]

11. Using the same list, `L`, you created in #10, write a Python program that uses the `set()` function to create a new list, `L11` that contains only the unique elements of `L`. Display both `L` and `L11`.