Lab 6 – PART 1

Topics Covered:

- Lists
- Dictionaries
- List and Dictionary Functions

Pre-Lab Questions

Question 1: Look at the following line of code, which creates a list of words. We want to create a for loop that will traverse through each **word in the list**. What is/are the correct way(s) to do this?

```
Words = ["hello","good","nice","as","at","baseball","absorb"]
```

a) for string in words:
b) for word in words:
c) for word in list:
d) All of the above
e) Both a) and b)

Question 2: What is the output of the following code snippet?

```
grocery_list = ["eggs", "bread", "cheese"]
for groc in grocery_list:
    print(grocery_list)
```

a) 0 1 2
b) eggs bread cheese
c) eggs
   bread
   cheese
d) ['eggs', 'bread', 'cheese']
   ['eggs', 'bread', 'cheese']
   ['eggs', 'bread', 'cheese']
e) Nothing; this code would produce an error

Question 3: What does the following function do?

```
def my_function(words):
    result = []
    for i in words:
        result.append(len(i))
    print(result)
```

Hint: words is a list of words

a) Prints the length of the last word in the list
b) Prints a list with the lengths of all the words in the list
c) Prints the total length of all the words in the list
d) Prints a list with each word followed by its length

Question 4: What does the following function do?

```
def my_function(one,two):
    for a in one:
        for b in two:
            if a == b:
                return True
    return False
```

Hint: the function will take in 2 lists of numbers as input

a) Returns True if there is at least one identical element in each list
b) Returns True if there is exactly one identical element in each list
c) Returns True if there is at least one identical element in both lists at the same index
d) Returns True if there is exactly one identical element in both lists at the same index

Question 5: What does the following function do?

```
def my_function(words):
    for i in words:
        if len(i) >= 4:
            words.remove(i)
    return words
```

Hint: words is a list of words

a) Removes any words that are less than 4 characters in length
b) Removes any words that are 4 or more characters in length
c) Shortens any words that are 4 or more characters in length down to the first 4 chars
d) Nothing; this code would produce an error

Question 6: What is the output of the following code snippet?

```
def my_function(strings):
    word_dict = {"short": [], "long": []}
    for s in strings:
        if len(s) > 4:
            word_dict["long"].append(s)
        else:
            word_dict["short"].append(s)
    return word_dict

# Create a list of words
words=["hello","as","great","nice","cat","baseball","absorb"]

# Call the function and print the output
res = my_function(words)
print(res["short"])
print(res["long"])
```

a) ['hello', 'great', 'baseball', 'absorb']
   ['as', 'nice', 'cat']
b) ['as', 'nice', 'cat']
   ['hello', 'great', 'baseball', 'absorb']
c) ['hello', 'as', 'great', 'nice', 'cat', 'baseball', 'absorb']
d) ['as', 'nice', 'cat']
e) ['hello', 'great', 'baseball', 'absorb']

Question 7: What is the output of the following code snippet?

```
def my_function(nums):
    my_dict = {}
    for num in nums:
        v = num % 10
        if v not in my_dict:
            my_dict[v] = [num]
        else:
            my_dict[v].append(num)
    return my_dict

print(my_function([37, 85, 14, 107, 94]))
```

a) {3: [37], 8: [85], 1: [14, 107], 9: [94]]}
b) {2: [37, 85, 14, 94], 3: [107]}
c) {7: [37], 5: [85], 4: [14], 7: [107], 4: [94]}
d) {7: [37, 107], 5: [85], 4: [14, 94]}

Activity:

1. Write a program that asks the user to enter ten unique numbers. The program must add each number to a list if they aren't already contained in the list. When the list contains ten unique numbers, the program displays the contents of the list and quits.

2. Write a program with a function that takes in a list, *strings*, and returns the number of those strings that satisfy both of the following conditions: 1) the length of the string is 3 characters or longer, **and** 2) the first and last characters of the string are the same.

   For example, if the strings list is: ['bgh', 'yuy', 'aa', 'wer', '1661'], the output should be 2 because the elements 'yuy' and '1661' are 3 or more characters long and start and end with the same character. The other elements from the list don't satisfy both conditions.

   Hint: If you have a string such as word = "abc", you can use word[-1] to access the last element in the string.

3. Write a program with a function that takes in a list, *groceries*, and creates a dictionary that stores all the items in lists organized (i.e. the key) by the first letter in the grocery's name. For example, if the groceries list is ["salami", "cucumber", "cheese", "milk", "salsa"], then the dictionary would contain 3 keys: "s", "c", and "m". Each would have a list as follows: "s": ["salami", "salsa"]; "c": ["cucumber", "cheese"]; "m": ["milk"].
   Use the following function shell to help you get started.

```
def make_dict(groceries):
    groc_dict = {}
    for ___________________:
        if _________not in __________:
            ____________________
        else:
            ____________________

    print(groc_dict)
```

Lab 6 – PART 2

1. **Follow the instructions below to construct a complete program.**

   Let's look at concatenation and sorting of lists. We are going to write a function z_first_sort that returns the words in a list sorted, BUT with all words that begin with z at the front:

One way that we can do this is creating two lists, one for words that begin with a ‘z’ and one for words that don’t begin with a ‘z’. Then, we can traverse through all the words in our list and append each word to the appropriate list. We can then sort our lists and concatenate them. Let's look at the code.

We can start by creating a function that with the 2 lists:

```
def z_first_sort(words):
    # We will need two lists
    zresult =[]
    result =[]
```

Now, lets traverse through our list. If the first letter is a z, we will add it to the zresult list, otherwise we will add it to result. We will use an if statement to check:

```
    for word in words:
        if word.lower()[0] == 'z':
            # If it does, add it to the first list
            zresult.append(word)
        else:
            # Does not begin with a 'z'
            result.append(word)
```

Notice here how we used word.lower() to make the word lowercase, because python doesn’t consider "z" and "Z" to be the same letter. Also notice how we reference the first letter of the word, we use the index which starts at 0. Remember that in programming the first element is always index 0 and not 1.

Now, after this loop finishes, we will have 2 lists, one for the words that start with z and another for the rest of the words. Lets sort these lists so that they are in alphabetical order:

```
    zresult.sort()
    result.sort()
```

Now, we have 2 sorted lists, we just have to join them together into one list. In python, we can do this using concatenation, and can simply add the lists together. Our function will return the concatenated list:

```
    return zresult + result
```

Here is the final code:

```
def z_first_sort(words):
    # We will need two lists
    zresult =[]
    result =[]
    for word in words:
        if word.lower()[0] == 'z':
            # If it does, add it to the first list
            zresult.append(word)
```

```
        else:
            # Does not begin with a 'z'
            result.append(word)

    zresult.sort()
    result.sort()

    return zresult + result
```

Let's test this out!

```
# Define a list of words
words=["hello","good","nice","as","at","baseball","absorb","sword","a","tall","so","bored"
,"silver","hi","pool","we","am","seven","do","you","want","ants","because","that's","how",
"you","get","zebra","zealot","zoo","xylophone","asparagus"]

# Print the result of using z_first_sort
print(z_first_sort(words))
```

The output is:

['zealot', 'zebra', 'zoo', 'a', 'absorb', 'am', 'ants', 'as', 'asparagus', 'at', 'baseball', 'because', 'bored', 'do', 'get', 'good', 'hello', 'hi', 'how', 'nice', 'pool', 'seven', 'silver', 'so', 'sword', 'tall', "that's", 'want', 'we', 'xylophone', 'you', 'you']

Which is exactly what we wanted – the list to be sorted with all the words beginning with z at the front.

2. **Identify errors and fix the code.**

The following code segment is a short program that copies the elements from an integer list into another integer list and adds 1 to each element. It then prints each element of both lists: the original one without modifications, and the one that has 1 added to every element. There are 4 errors. Find and fix these errors. Hint: remember how to copy a list without pointing to the same list in memory.

```
values = [1,2,3,4,5]
newValues = values
for i in range(len(values)+1):
    values[i] +=1
    print("Old Value at index {} is: {} ".format(i, newValues[i]))
    print("New Value at index {} is: {} \n".format(i, newValues[i]))
```