**Week 4 Graded Assessment**

**Question 1**

Fill in the blanks to complete the “confirm\_length” function. This function should return how many characters a string contains as long as it has one or more characters, otherwise it will return 0. Complete the string operations needed in this function so that input like "Monday" will produce the output "6".

def confirm\_length(word):

    # Complete the condition statement using a string operation.

    if len(word) > 0:

        # Complete the return statement using a string operation.

        return len(word)

    else:

        return 0

print(confirm\_length("a")) # Should print 1

print(confirm\_length("This is a long string")) # Should print 21

print(confirm\_length("Monday")) # Should print 6

print(confirm\_length("")) # Should print 0

**Question 2**

Fill in the blank to complete the “highlight\_word” function. This function should change the given “word” to its upper-case version in a given “sentence”. Complete the string method needed in this function so that a function call like "highlight\_word("Have a nice day", "nice")" will

def highlight\_word(sentence, word):

    # Complete the return statement using a string method.

    return sentence.replace(word, word.upper())

print(highlight\_word("Have a nice day", "nice"))

# Should print: "Have a NICE day"

print(highlight\_word("Shhh, don't be so loud!", "loud"))

# Should print: "Shhh, don't be so LOUD!"

print(highlight\_word("Automating with Python is fun", "fun"))

# Should print: "Automating with Python is FUN"

**Question 3**

Consider the following scenario about using Python lists:

Employees at a company shared  the distance they drive to work (in miles) through an online survey. These distances were automatically added by Python to a list called “distances” in the order that each employee submitted their distance. Management wants the list to be sorted in the order of the longest distance to the shortest distance.

Complete the function to sort the “distances” list. This function should:

1. sort the given “distances” list, passed through the function’s parameters; ;
2. reverse the sort order so that it goes from the longest to the shortest distance;
3. return the modified “distances” list.

def sort\_distance(distances):

    distances.sort() # Sort the list

    distances.reverse() # Reverse the order of the list

    return distances

print(sort\_distance([2,4,0,15,8,9]))

# Should print [15, 9, 8, 4, 2, 0]

**Question 4**

Fill in the blank to complete the “even\_numbers” function. This function should use a list comprehension to create a list of even numbers using a conditional if statement with the modulo operator to test for numbers evenly divisible by 2. The function receives two variables and should return the list of even numbers that occur between the “first” and “last” variables**exclusively** *(meaning don’t modify the default behavior of the range to exclude the “end” value in the range)*. For example, even\_numbers(2, 7) should return [2, 4, 6].

def even\_numbers(first, last):

  return [even for even in range(first, last) if even % 2 == 0]

print(even\_numbers(4, 14)) # Should print [4, 6, 8, 10, 12]

print(even\_numbers(0, 9))  # Should print [0, 2, 4, 6, 8]

print(even\_numbers(2, 7))  # Should print [2, 4, 6]

**Question 5**

Fill in the blanks to complete the “countries” function. This function accepts a dictionary containing a list of continents (keys) and several countries from each continent (values).  For each continent, format a string to print the names of the countries only. The values for each key should appear on their own line.

def countries(countries\_dict):

    result = ""

    # Complete the for loop to iterate through the key and value items

    # in the dictionary.

    for countries in countries\_dict.values():

        # Use a string method to format the required string.

        result += "{}".format(countries) + "\n"

    return result

print(countries({"Africa": ["Kenya", "Egypt", "Nigeria"], "Asia":["China", "India", "Thailand"], "South America": ["Ecuador", "Bolivia", "Brazil"]}))

# Should print:

# ['Kenya', 'Egypt', 'Nigeria']

# ['China', 'India', 'Thailand']

# ['Ecuador', 'Bolivia', 'Brazil']

**Question 6**

Consider the following scenario about using Python dictionaries:

Tessa and Rick are hosting a party. Both sent out invitations to their friends, and each one collected responses into dictionaries, with names of their friends and how many guests each friend was bringing. Each dictionary is a partial guest list, but Rick's guest list has more current information about the number of guests.

Complete the function to combine both dictionaries into one, with each friend listed only once, and the number of guests from Rick's dictionary taking precedence, if a name is included in both dictionaries. Then print the resulting dictionary. This function should:

1. accept two dictionaries through the function’s parameters;
2. combine both dictionaries into one, with each key listed only once;
3. the values from the “guests1” dictionary taking precedence, if a key is included in both dictionaries;
4. then print the new dictionary of combined items.

def combine\_guests(guests1, guests2):

  guests2.update(guests1) # Use a dictionary method to combine the dictionaries.

  return guests2

Ricks\_guests = { "Adam":2, "Camila":3, "David":1, "Jamal":3, "Charley":2, "Titus":1, "Raj":4}

Tessas\_guests = { "David":4, "Noemi":1, "Raj":2, "Adam":1, "Sakira":3, "Chidi":5}

print(combine\_guests(Ricks\_guests, Tessas\_guests))

# Should print:

# {'David': 1, 'Noemi': 1, 'Raj': 4, 'Adam': 2, 'Sakira': 3, 'Chidi': 5, 'Camila': 3, 'Jamal': 3, 'Charley': 2, 'Titus': 1}

**Question 7**

Complete the function so that input like "This is a sentence." will return a dictionary that holds the count of each letter that occurs in the string: {'t': 2, 'h': 1, 'i': 2, 's': 3, 'a': 1, 'e': 3, 'n': 2, 'c': 1}. This function should:

1. accept a string “text” variable through the function’s parameters;
2. iterate over each character the input string to count the frequency of each letter found, (only letters should be counted, do not count blank spaces, numbers, or punctuation; keep in mind that Python is case sensitive);
3. populate the new dictionary with the letters as keys, ensuring each key is unique, and assign the value for each key with the count of that letter;
4. return the new dictionary.

def count\_letters(text):

  # Initialize a new dictionary.

  dictionary = {}

  # Complete the for loop to iterate through each "text" character and

  # use a string method to ensure all letters are lowercase.

  for letter in text.lower():

    # Complete the if-statement using a string method to check if the

    # character is a letter.

    if letter.isalpha():

      # Complete the if-statement using a logical operator to check if

      # the letter is not already in the dictionary.

      if letter not in dictionary:

           # Use a dictionary operation to add the letter as a key

           # and set the initial count value to zero.

           dictionary[letter] = 0

      # Use a dictionary operation to increment the letter count value

      # for the existing key.

      dictionary[letter] += 1

      #\_\_\_ # Increment the letter counter.

  return dictionary

print(count\_letters("AaBbCc"))

# Should be {'a': 2, 'b': 2, 'c': 2}

print(count\_letters("Math is fun! 2+2=4"))

# Should be {'m': 1, 'a': 1, 't': 1, 'h': 1, 'i': 1, 's': 1, 'f': 1, 'u': 1, 'n': 1}

print(count\_letters("This is a sentence."))

# Should be {'t': 2, 'h': 1, 'i': 2, 's': 3, 'a': 1, 'e': 3, 'n': 2, 'c': 1}

**Question 8**

What do the following commands return when animal = "Hippopotamus"?

print(animal[3:6])

print(animal[-5])

print(animal[10:])

**Answer:** pop, t, us

**Question 9**

What does the list "colors" contain after these commands are executed?

colors = ["red", "white", "blue"]

colors.insert(2, "yellow")



['red', 'white', 'yellow']



['red', 'yellow', 'blue']



['red', 'yellow', 'white', 'blue']



['red', 'white', 'yellow', 'blue']

**Question 10**

What do the following commands return?

host\_addresses = {"router": "192.168.1.1", "localhost": "127.0.0.1", "google": "8.8.8.8"}

host\_addresses.keys()



dict\_keys({"router": "192.168.1.1", "localhost": "127.0.0.1", "google": "8.8.8.8"})



dict\_keys(['router', 'localhost', 'google'])



dict\_keys(['192.168.1.1', '127.0.0.1', '8.8.8.8'])



dict\_keys(["router", "192.168.1.1", "localhost", "127.0.0.1", "google", "8.8.8.8"]

**Additional Questions**

**Question 1**

Complete the for loop and string method needed in this function so that a function call like "alpha\_length("This has 1 number in it")" will return the output "17". This function should:

1. accept a string through the parameters of the function;
2. iterate over the characters in the string;
3. determine if each character is a letter (counting only alphabetic characters; numbers, punctuation, and spaces should be ignored);
4. increment the counter;
5. return the count of letters in the string.

def alpha\_length(string):

    character = ""

    count\_alpha = 0

    # Complete the for loop sequence to iterate over "string".

    for letter in string:

        # Complete the if-statement using a string method.

        if letter.isalpha():

            count\_alpha += 1

    return count\_alpha

print(alpha\_length("This has 1 number in it")) # Should print 17

print(alpha\_length("Thisisallletters")) # Should print 16

print(alpha\_length("This one has punctuation!")) # Should print 21

**Question 2**

Consider the following scenario about using Python lists:

A professor gave his two assistants, Jaime and Drew, the task of keeping an attendance list of students in the order they arrive in the classroom. Drew was the first one to note which students arrived, and then Jaime took over. After the class, they each entered their lists into the computer and emailed them to the professor. The professor wants to combine the two lists into one, in the order of each student's arrival. Jaime emailed a follow-up, saying that her list is in reverse order.

Complete the code to combine the two lists into one in the order of: **the contents of Drew's list, followed by Jaime’s list in reverse order**, to produce an accurate list of the students as they arrived. This function should:

1. accept two lists through the function’s parameters;
2. reverse the order of “list1”;
3. combine the two lists so that “list2” comes first, followed by “list1”;
4. return the new list.

def combine\_lists(list1, list2):

  combined\_list = [] # Initialize an empty list variable

  list1.reverse() # Reverse the order of "list1"

  list2.extend(list1) # Combine the two lists

  combined\_list = list2

  return combined\_list

Jaimes\_list = ["Alma", "Chika", "Benjamin", "Jocelyn", "Oakley"]

Drews\_list = ["Minna", "Carol", "Gunnar", "Malena"]

print(combine\_lists(Jaimes\_list, Drews\_list))

# Should print ['Minna', 'Carol', 'Gunnar', 'Malena', 'Oakley', 'Jocelyn', 'Benjamin', 'Chika', 'Alma']

**Question 3**

Fill in the blank to complete the “squares” function. This function should use a list comprehension to create a list of squared numbers (using either the expression n\*n or n\*\*2). The function receives two variables and should return the list of squares that occur between the “start” and “end” variables**inclusively** *(meaning the range should include both the “start” and “end” values)*. Complete the list comprehension in this function so that input like “squares(2, 3)” will produce the output “[4, 9]”.

def squares(start, end):

    return [n\*\*2 for n in range(start, end+1)] # Create the required list comprehension.

print(squares(2, 3)) # Should print [4, 9]

print(squares(1, 5)) # Should print [1, 4, 9, 16, 25]

print(squares(0, 10)) # Should print [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

**Question 4**

Consider the following scenario about using Python dictionaries and lists:

Tessa and Rick are hosting a party. Before they send out invitations, they want to add all of the people they are inviting to a dictionary so they can also add how many guests each friend is bringing to the party.

Complete the function so that it accepts a list of people, then iterates over the list and adds all of the names (elements) to the dictionary as keys with a starting value of 0. Tessa and Rick plan to update these values with the number of guests their friends will bring with them to the party. Then, print the new dictionary.

This function should:

1. accept a list variable named “guest\_list” through the function’s parameter;
2. add the contents of the list as keys to a new, blank dictionary;
3. assign each new key with the value 0;
4. print the new dictionary.

def setup\_guests(guest\_list):

    # loop over the guest list and add each guest to the dictionary with

    # an initial value of 0

    result = {} # Initialize a new dictionary

    for element in guest\_list: # Iterate over the elements in the list

        result[element] = 0 # Add each list element to the dictionary as a key with

            # the starting value of 0

    return result

guests = ["Adam","Camila","David","Jamal","Charley","Titus","Raj","Noemi","Sakira","Chidi"]

print(setup\_guests(guests))

# Should print {'Adam': 0, 'Camila': 0, 'David': 0, 'Jamal': 0, 'Charley': 0, 'Titus': 0, 'Raj': 0, 'Noemi': 0, 'Sakira': 0, 'Chidi': 0}