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". **1 point**

```
def confirm_length(word):

    # Complete the condition statement using a string operation.
    if word != " ":
        # 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
```

Run Reset

```
1
21
6
0
```

2. 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 point**

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 char in string:
        # Complete the if-statement using a string method.
        if char.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
```

Run Reset

```
17
16
21
```

3. Consider the following scenario about using Python lists: **1 point**

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(reverse = True)
    return distances


print(sort_distance([2,4,0,15,8,9]))
# Should print [15, 9, 8, 4, 2, 0]
```

Run Reset

```
[15, 9, 8, 4, 2, 0]
```

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]. **1 point**

```
def even_numbers(first, last):
  return [ i for i in range(first, last, 2) ]


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]
```

Run Reset

```
[4, 6, 8, 10, 12]
[0, 2, 4, 6, 8]
[2, 4, 6]
```

5. Fill in the blanks to complete the "endangered_animals" function. This function accepts a dictionary containing a list of endangered animals (keys) and their remaining population (values). For each key in the given "animal_dict" **1 point**

list of endangered animals (keys) and their remaining population (values). For each key in the given "animal_dict" dictionary, format a string to print the name of the animal, with one animal name per line.

```
def endangered_animals(animal_dict):
    result = ""
    # Complete the for loop to iterate through the key and value items
    # in the dictionary.
    for key,value in animal_dict.items():
        # Use a string method to format the required string.
        result += key + "\n"
    return result


print(endangered_animals({"Javan Rhinoceros":60, "Vaquita":10, "Mountain Gorilla":200, "T

# Should print:
# Javan Rhinoceros
# Vaquita
# Mountain Gorilla
# Tiger
```

Run

Reset

```
Javan Rhinoceros
Vaquita
Mountain Gorilla
Tiger
```

6. Consider the following scenario about using Python dictionaries: **1 point**

Tessa and Rick are hosting a party. Together, they sent out invitations, and collected the responses in a dictionary, with names of their friends and the number of guests each friend will be bringing.

Complete the function so that the "check_guests" function retrieves the number of guests (value) the specified friend "guest" (key) is bringing. This function should:

1. accept a dictionary "guest_list" and a key "guest" variable passed through the function parameters;
2. print the values associated with the key variable.

```
def check_guests(guest_list, guest):
  return guest_list.get(guest) # Return the value for the given key


guest_list = { "Adam":3, "Camila":3, "David":5, "Jamal":3, "Charley":2, "Titus":1, "Raj":

print(check_guests(guest_list, "Adam")) # Should print 3
print(check_guests(guest_list, "Sakira")) # Should print 3
print(check_guests(guest_list, "Charley")) # Should print 2
```

Run

Reset

```
3
3
2
```

7. Use a dictionary to count the frequency of numbers in the given "text" string. Only numbers should be counted. Do not count blank spaces, letters, or punctuation. Complete the function so that input like "1001000111101" will return a dictionary that holds the count of each number that occurs in the string {'1': 7, '0': 6}. This function should: **1 point**

1. accept a string "text" variable through the function's parameters;
2. initialize an new dictionary;
3. iterate over each text character to check if the character is a number'
4. count the frequency of numbers in the input string, ignoring all other characters;
5. populate the new dictionary with the numbers as keys, ensuring each key is unique, and assign the value for each key with the count of that number;
6. return the new dictionary.

```
def count_numbers(text):
  # Initialize a new dictionary.
  dictionary = {}
  # Complete the for loop to iterate through each "text" character.
  for num in text:
    # Complete the if-statement using a string method to check if the
    # character is a number.
    if num.isdigit():
      # Complete the if-statement using a logical operator to check if
      # the number is not already in the dictionary.
      if num in dictionary:
        dictionary[num] +=1
          # Use a dictionary operation to add the number as a key
          # and set the initial count value to zero.
      # Use a dictionary operation to increment the number count value
      # for the existing key.
      else:
        dictionary[num] = 1
  return dictionary

print(count_numbers("1001000111101"))
# Should be {'1': 7, '0': 6}

print(count_numbers("Math is fun! 2+2=4"))
# Should be {'2': 2, '4': 1}

print(count_numbers("This is a sentence."))
# Should be {}

print(count_numbers("55 North Center Drive"))
# Should be {'5': 2}
```

Run

Reset

```
{'1': 7, '0': 6}
{'2': 2, '4': 1}
{}
{'5': 2}
```

8. What do the following commands return when animal = "Hippopotamus"? **1 point**

```
print(animal[3:6])
print(animal[-5])
print(animal[10:])
```

- popo, t, mus
- ppo, t, mus
- pop, t, us
- ppop, o, s

9. What does the list "colors" contain after these commands are executed? **1 point**

```
colors = ["red", "white", "blue"]
colors.insert(2, "yellow")
```

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

10. What do the following commands return? **1 point**

```
host_addresses = {"router": "192.168.1.1", "localhost": "127.0.0.1", "google": "8.8.8.8"}
host_addresses.keys()
```

- dict_keys(['router', 'localhost', 'google'])
- dict_keys({"router": "192.168.1.1", "localhost": "127.0.0.1", "google": "8.8.8.8"})
- dict_keys(["router", "192.168.1.1", "localhost", "127.0.0.1", "google", "8.8.8.8"])
- dict_keys(['192.168.1.1', '127.0.0.1', '8.8.8.8'])

Upgrade to submit