Question 1

Fill in the blanks to complete the function “even_numbers(n)”. This function should count how many even numbers exist in a sequence from 0 to the given “n”number, where 0 counts as an even number. For example, even_numbers(25) should return 13, and even_numbers(6) should return 4.

```
def even_numbers(n):
    count = 0
    current_number = 0
    while current_number <= n:  # Complete the while loop condition
        if current_number % 2 == 0:
            count += 1 # Increment the appropriate variable
        current_number += 1 # Increment the appropriate variable
    return count

print(even_numbers(25))   # Should print 13
print(even_numbers(144))  # Should print 73
print(even_numbers(1000)) # Should print 501
print(even_numbers(0))    # Should print 1
```

Question 2

Fill in the blanks to complete the “sequence” function. This function should print a sequence of numbers in descending order, from the given "high" variable to the given "low" variable. The range should make the loop run two times. Complete the range sequences in the nested loops so that the “sequence(1, 3)” function call prints the following:

3, 2, 1

3, 2, 1

```
def sequence(low, high):
    # Complete the outer loop range to make the loop run twice
    # to create two rows
    for x in range(2):
        # Complete the inner loop range to print the given variable
        # numbers starting from "high" to "low"
        # Hint: To decrement a range parameter, use negative numbers
        for y in range(high, low-1, -1):
            if y == low:
                # Don’t print a comma after the last item
                print(str(y))
            else:
                # Print a comma and a space between numbers
```

```
            print(str(y),end=", ")

sequence(1, 3)
# Should print the sequence 3, 2, 1 two times, as shown above.
```

Question 3

Fill in the blanks to complete the “counter” function. This function should count down from the “start” to “stop” variables when “start” is bigger than “stop”. Otherwise, it should count up from “start” to “stop”. Complete the code so that a function call like “counter(3, 1)” will return “Counting down: 3, 2, 1” and “counter(2, 5)” will return “Counting up: 2, 3, 4, 5”.

```
def counter(start, stop):
    if start > stop:
        return_string = "Counting down: "
        while start >= stop: # Complete the while loop
            return_string += str(start)#return_string Add the numbers to the "return_string"
            if start > stop:
                return_string += ","
            start -= 1 # Increment the appropriate variable
    else:
        return_string = "Counting up: "
        while start <= stop: # Complete the while loop
            return_string += str(start) # Add the numbers to the "return_string"
            if start < stop:
                return_string += ","
            start += 1 # Increment the appropriate variable
    return return_string


print(counter(1, 10)) # Should be "Counting up: 1,2,3,4,5,6,7,8,9,10"
print(counter(2, 1)) # Should be "Counting down: 2,1"
print(counter(5, 5)) # Should be "Counting up: 5"
```

Question 4

Fill in the blanks to complete the “even_numbers” function. This function should return a space-separated string of all positive even numbers, excluding 0, up to and including the “maximum” variable that's passed into the function. Complete the for loop so that a function call like “even_numbers(6)” will return the numbers “2 4 6”.

```
def even_numbers(maximum):

    return_string = "" # Initializes variable as a string

    # Complete the for loop with a range that includes all even numbers
    # up to and including the "maximum" value, but excluding 0.
```

```
    for number in range(2, maximum + 1, 2):

        # Complete the body of the loop by appending the even number
        # followed by a space to the "return_string" variable.
        return_string += str(number) + " "

    # This .strip command will remove the final " " space at the end of
    # the "return_string".
    return return_string.strip()

print(even_numbers(6))  # Should be 2 4 6
print(even_numbers(10)) # Should be 2 4 6 8 10
print(even_numbers(1))  # No numbers displayed
print(even_numbers(3))  # Should be 2
print(even_numbers(0))  # No numbers displayed
```

Question 5

The format of the input variable “address_string” is: numeric house number, followed by the street name which may contain numbers and could be several words long (e.g., "123 Main Street", "1001 1st Ave", or "55 North Center Drive").

Complete the string methods needed in this function so that input like "123 Main Street" will produce the output "House number 123 on a street named Main Street". This function should:

1. accept a string through the parameters of the function;
2. separate the address string into new strings: house_number and street_name;
3. return the variables in the string format: "House number X on a street named Y".

```
def format_address(address_string):
    house_number = ""
    street_name = ""

    # Separate the house number from the street name.
    address_parts = address_string.split()

    for address_part in address_parts:
        # Complete the if-statement with a string method.
        if address_part.isdigit():
            house_number = "House number " + address_part
        else:
            street_name += address_part + " "
```

```
    # Remove the extra space at the end of the last "street_name".
    street_name = street_name.strip()

    # Use a string method to return the required formatted string.
    return house_number + " on a street named " + street_name

print(format_address("123 Main Street"))
# Should print: "House number 123 on a street named Main Street"

print(format_address("1001 1st Ave"))
# Should print: "House number 1001 on a street named 1st Ave"

print(format_address("55 North Center Drive"))
# Should print "House number 55 on a street named North Center Drive"
```

Question 6

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"
  combined_list = list2 + list1 # Combine the two lists

  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', 'Alm
a']
```

Question 7

Fill in the blank to complete the “increments” function. This function should use a list comprehension to create a list of numbers incremented by 2 (n+2). The function receives two variables and should return a list of incremented consecutive numbers between “start” and “end” 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, 5]”.

```
def increments(start, end):
   return [x + 2 for x in range(start, end + 1)] # Create the required list comprehension.


print(increments(2, 3)) # Should print [4, 5]
print(increments(1, 5)) # Should print [3, 4, 5, 6, 7]
print(increments(0, 10)) # Should print [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
```

Question 8

Fill in the blanks to complete the “car_listing” function. This function accepts a “car_prices” dictionary. It should iterate through the keys (car models) and values (car prices) in that dictionary. For each item pair, the function should format a string so that a dictionary entry like ““Kia Soul“:19000” will print "A Kia Soul costs 19000 dollars". Each new string should appear on its own line.

```
def car_listing(car_prices):
  result = ""
  # Complete the for loop to iterate through the key and value items
  # in the dictionary.
  for car, price in car_prices.items():
    result += f"A {car} costs {price} dollars\n" # Use a string method to format the required string.
  return result

print(car_listing({"Kia Soul":19000, "Lamborghini Diablo":55000, "Ford Fiesta":13000, "Toyota Pri
us":24000}))

# Should print:
```

```
# A Kia Soul costs 19000 dollars
# A Lamborghini Diablo costs 55000 dollars
# A Ford Fiesta costs 13000 dollars
# A Toyota Prius costs 24000 dollars
```

Question 9

Consider the following scenario about using Python dictionaries:

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, "Guest not found") # Return the value for the given key
guest_list = { "Adam":3, "Camila":3, "David":5, "Jamal":3, "Charley":2, "Titus":1, "Raj":6, "Noemi":1, "Sakira":3, "Chidi":5}
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
```

Question 10

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 char in text.lower():
# Complete the if-statement using a string method to check if the
# character is a letter.
if char.isalpha():
# Complete the if-statement using a logical operator to check if
# the letter is not already in the dictionary.
if char not in dictionary:

        # Use a dictionary operation to add the letter as a key
        # and set the initial count value to zero.
        dictionary[char] = 0
# Use a dictionary operation to increment the letter count value
# for the existing key.

dictionary[char] += 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 11

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 return the output "Have a NICE day".

```
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"
```