1. The format_address function separates out parts of the address string into new strings: house_number and street_name, and returns: "house number X on street named Y". The format of the input string is: numeric house number, followed by the street name which may contain numbers, but never by themselves, and could be several words long. For example, "123 Main Street", "1001 1st Ave", or "55 North Center Drive". Fill in the gaps to complete this function.

```
def format_address(address_string):
  # Declare variables
  house_no=''
  street_name=''
  # Separate the address string into parts
  part=address_string.split()
  # Traverse through the address parts
  for word in part:
    # Determine if the address part is the
    # house number or part of the street name
    if word.isdigit():
      house_no=word
  # Does anything else need to be done
  # before returning the result?
    else:
      street_name+=word+' '

  # Return the formatted string
  return "house number {} on street named {}".format(house_no,street_name)

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

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

print(format_address("55 North Center Drive"))
# Should print "house number 55 on street named North Center Drive"

```

Run

Reset

2. The highlight_word function changes the given word in a sentence to its upper-case version. For example, highlight_word("Have a nice day", "nice") returns "Have a NICE day". Can you write this function in just one line?

1 / 1 point

```
def highlight_word(sentence, word):
  return(sentence.replace(word,word.upper()))

print(highlight_word("Have a nice day", "nice"))
print(highlight_word("Shhh, don't be so loud!", "loud"))
print(highlight_word("Automating with Python is fun", "fun"))

```

Run

Reset

3. A professor with two assistants, Jamie and Drew, wants an attendance list of the students, in the order that they arrived in the classroom. Drew was the first one to note which students arrived, and then Jamie took over. After the class, they each entered their lists into the computer and emailed them to the professor, who needs to combine them into one, in the order of each student's arrival. Jamie emailed a follow-up, saying that her list is in reverse order. Complete the steps to combine them into one list as follows: **the contents of Drew's list, followed by Jamie's list in reverse order**, to get an accurate list of the students as they arrived.

1 / 1 point

```
1  def combine_lists(list1, list2):
2    # Generate a new list containing the elements of list2
3    # Followed by the elements of list1 in reverse order
4    new_list=list2
5    for i in reversed(range(len(list1))):
6      new_list.append(list1[i])
7    return new_list
8
9  Jamies_list = ["Alice", "Cindy", "Bobby", "Jan", "Peter"]
10 Drews_list = ["Mike", "Carol", "Greg", "Marcia"]
11
12 print(combine_lists(Jamies_list, Drews_list))
13
```

Run

Reset

4. Use a list comprehension to create a list of squared numbers (n*n). The function receives the variables *start* and *end*, and returns a list of squares of consecutive numbers between *start* and *end* **inclusively**. For example, squares(2, 3) should return [4, 9].

1 / 1 point

```
def squares(start, end):
 multiple=[]
 for x in range(start,end+1):
   multiple.append(x*x)

 return multiple

print(squares(2, 3)) # Should be [4, 9]
print(squares(1, 5)) # Should be [1, 4, 9, 16, 25]
print(squares(0, 10)) # Should be [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Run

Reset

5. Complete the code to iterate through the keys and values of the car_prices dictionary, printing out some information about each one.

1 / 1 point

```
def car_listing(car_prices):
  result = ""
  for name,price in car_prices.items():
    result += "{} costs {} dollars".format(name,price) + "\n"
  return result

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

Run

Reset

6. Taylor and Rory are hosting a party. They sent out invitations, and each one collected responses into dictionaries, with names of their friends and how many guests each friend is bringing. Each dictionary is a partial list, but Rory's list has more current information about the number of guests. Fill in the blanks to combine both dictionaries into one, with each friend listed only once, and the number of guests from Rory's dictionary taking precedence, if a name is included in both dictionaries. Then print the resulting dictionary.

0 / 1 point

```
def combine_guests(guests1, guests2):
  # Combine both dictionaries into one, with each key listed
  # only once, and the value from guests1 taking precedence
  guests=dict(guests1)
  guests.update(guests2)

  for key1,values1 in guests1.items():
    for key2,values2 in guests2.items():
      if key1==key2:
        guests[key1]=values1+values2

  return guests

Rorys_guests = { "Adam":2, "Brenda":3, "David":1, "Jose":3, "Charlotte":2,
  "Terry":1, "Robert":4}
Taylors_guests = { "David":4, "Nancy":1, "Robert":2, "Adam":1, "Samantha":3,
  "Chris":5}

print(combine_guests(Rorys_guests, Taylors_guests))
```

Run

Reset

7. Use a dictionary to count the frequency of letters in the input string. Only letters should be counted, not blank spaces, numbers, or punctuation. Upper case should be considered the same as lower case. For example, count_letters("This is a sentence.") should return {'t': 2, 'h': 1, 'i': 2, 's': 3, 'a': 1, 'e': 3, 'n': 2, 'c': 1}.

0 / 1 point

```
def count_letters(text):
  result = {}

  # Go through each letter in the text
  for letter in text:


    # Check if the letter needs to be counted or not
    if letter in "abcdefghijklmnopqrstuvwxyz":
      if letter not in result :
        result[letter.lower()]=0
    # Add or increment the value in the dictionary

      result[letter.lower()]+=1
  return result

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}
```

Run

Reset

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

1 / 1 point

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

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

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

1 / 1 point

```
1  colors = ["red", "white", "blue"]
2  colors.insert(2, "yellow")
3
```

- (●) ['red', 'white', 'yellow', 'blue']
- ( ) ['red', 'yellow', 'white', 'blue']
- ( ) ['red', 'yellow', 'blue']
- ( ) ['red', 'white', 'yellow']

10. What do the following commands return?

1 / 1 point

```
1  host_addresses = {"router": "192.168.1.1", "localhost": "127.0.0.1", "google":
     "8.8.8.8"}
2  host_addresses.keys()
3
```

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