class PhoneNumber:
    """
    A PhoneNumber represents a single phone number made up of the
    area code, exchange, and the line number.
    
    EX: (415) 552-7909
    ^    ^   ^
    |    |   |
    |    |   number
    |    exchange
    area code
    """

def __init__(self, area_code, exchange, number):
    """
    Creates a new PhoneNumber from the provided area
code, exchange and number.
    """
    self.area_code = area_code
    self.exchange = exchange
    self.number = number

def call(self):
    """
    Calls this PhoneNumber.
    """
    print("Calling (" + str(self.area_code) +")  " +
    str(self.exchange) + "-" + str(self.number))
    print("ring... ring... Hello?")

def print_number(self):
    """
    Prints a pretty version of this PhoneNumber.
    """
    print("(" + str(self.area_code) + ")  " +
    str(self.exchange) \ + "-" + str(self.number))

class PhoneBook:
    """
    A PhoneBook is a collection of names and phone numbers.
    """
    def __init__(self):
        """
        Creates a new PhoneBook that is initially empty.
        """
        self.contacts = {}
def add_number(self, name, phone_number):
    """
    Adds the provided name and PhoneNumber to this PhoneBook. 
    Will replace the number if the name already exists in this 
    PhoneBook.
    """
    self.contacts[name] = phone_number

def delete_contact(self, name):
    """
    Removes the provided name and the associated PhoneNumber 
    from this PhoneBook.
    """
    # This is how to remove from a dict. We might not have 
    used this before.
    del self.contacts[name]

def call(self, name):
    """
    Calls the phone number associated with the provided name.
    """
    self.contacts[name].call()
    print("Hi this is " + name + ").")

def get_phone_number(self, name):
    """
    Returns the PhoneNumber associated with the provided name.
    """
    return self.contacts[name]

def get_contacts_in_area_code(self, area_code):
    """
    Returns a list of all PhoneNumbers in this PhoneBook that 
    have 
    the given area_code.
    """
    result = []
    for name in self.contacts:
        num = self.contacts[name]
        if num.area_code == area_code:
            result.append(num)

    return result
from phonebook import *

# Make some new phone numbers
num1 = PhoneNumber(916, 272, 8010)
num2 = PhoneNumber(916, 274, 2805)
num3 = PhoneNumber(415, 552, 7909)

# Try printing them
num1.print_number()
num2.print_number()
# print(num1)
# print(num2)

# Try calling
num1.call()

# Make a new phone book
my_contacts = PhoneBook()

# add some contacts
my_contacts.add_number("Nick", num1)
my_contacts.add_number("Justin", num2)

# try calling the contacts
my_contacts.call("Nick")
my_contacts.call("Justin")

# experiment with getting the phone number from the phone book
num4 = my_contacts.get_phone_number("Justin")
print(num1 == num4)
print(num2 == num4)

numbers = my_contacts.get_contacts_in_area_code(916)
for num in numbers:
    num.print_number()
    # print(num)
1. For the following code, write its output. If there is an error, describe the error and the cause, and include the output up until the error.

```python
def histogram(words, stop_words=[]):
    ""
    Return a dictionary mapping each word (separated by white
    space) in the string words to
    the number of times it occurs. Exclude words that appear
    in stop_words.
    ""
    d = {}
    for w in words:
        if not w in stop_words:
            c = d.setdefault(w, 0)
            d[w] = c + 1
    return d

phrase = "I didn’t ask for a dime"
d = histogram(phrase, ["for"])print(d["a"])
print(d["dime"])
```

2. In homework 6, you will be using statistical tools to analyze datasets. One common measure for the difference/distance between two datasets is the mean squared error. MSE is computed as follows:

For each point in one dataset:
- compute the difference between it and the corresponding point in the other dataset
- square this difference
- Take the average of these squared differences.

Compute the MSE between f, g, and h. What can these numbers tell you?

<table>
<thead>
<tr>
<th>x</th>
<th>f(x)</th>
<th>g(x)</th>
<th>h(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>9</td>
<td>4</td>
</tr>
</tbody>
</table>
1. This code will cause an error. This function iterates through its string parameter with a for-loop. As it is written, the iterating variable will take on values character by character instead of word by word. So, when the word “dime” is searched for as a key in the dictionary, it doesn’t appear in the dictionary. One way to iterate through the first string word by word would be to change the code to: for w in words.split():

2. The MSE difference between f and g is \((4-1)^2 + (5-3)^2 + (6-9)^2) / 3 = 7.333.\]
   The MSE difference between f and h is \((4-8)^2 + (5-6)^2 + (6-4)^2) / 3 = 9.\]
   The MSE difference between g and h is \((1-8)^2 + (3-6)^2 + (9-4)^2) / 3 = 27.667.\]