|  |  |
| --- | --- |
| **NAME:** | **DUE:** |
| **Tomoki Koike** | **4/21/2019** |

**Every Boiler Engineering Code – Entry Level Programming**

**Week 7 – Programming Exercises**

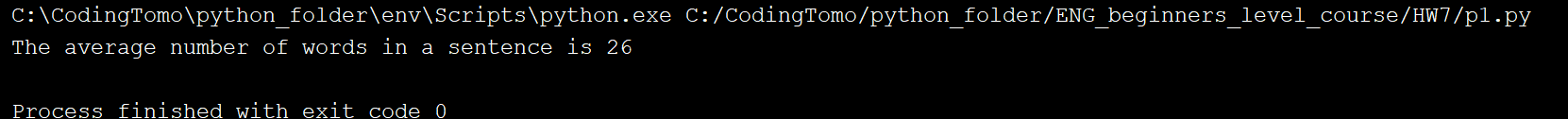
|  |  |
| --- | --- |
| **QUESTION** | **POINTS** |
| **#1** | **/15** |
| **#2** | **/15** |
| **#3** | **/15** |
| **#4** | **/15** |
| **#5** | **/15** |
| **TOTAL** | **/75** |

1. **(15 points, Average Number of Words)** There’s a text file in the attachment called ‘**text.txt’**. The text is in the file is stored as one sentence per line. Write a Python program that reads the files contents and calculates the average number of words per sentence.

**CODE:**

## PROBLEM #1  
# NAME: Tomoki Koike  
# DUE: 4/21/2019  
# DESCRIPTION: This program will read a file with sentences and  
# output the average number of words per sentence line.  
# STAND: Class of 2020  
##  
  
# Import modulus  
import statistics as stat  
  
# Functions  
# Function that reads the file  
def file\_reader(file):  
 # Opening the file  
 afile = open(file**,** 'r')  
 # Reading the first line in the file  
 aline = afile.readline()  
 # Looping to read all the lines in the file  
 # Preallocating a list to store the number of words in sentence  
 word\_count = []  
 while aline != '':  
 # Strip sentence ending  
 aline = aline.rstrip('\n')  
 # Function call to split the line into words  
 aline = sentence\_spliter(aline)  
 # Function call to output the number of words in the sentence  
 word\_count.append(word\_counter(aline))  
 # Reading the next line  
 aline = afile.readline()  
 return word\_count  
  
# Function that splits all the words in the sentence  
def sentence\_spliter(line):  
 return line.split()  
  
# Function that calculates and stores the number of words in the sentence  
def word\_counter(split\_line):  
 return len(split\_line)  
  
# Function to calculate the average number of words per sentence  
def avg\_words(word\_count):  
 return stat.mean(word\_count)  
  
def main():  
 # Function to read file and return the number of words per line  
 word\_count = file\_reader('text.txt')  
 # Output the average number of words in sentences  
 print('The average number of words in a sentence is {0}'.format(avg\_words(word\_count)))  
 return **0**if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**EXECUTION:**



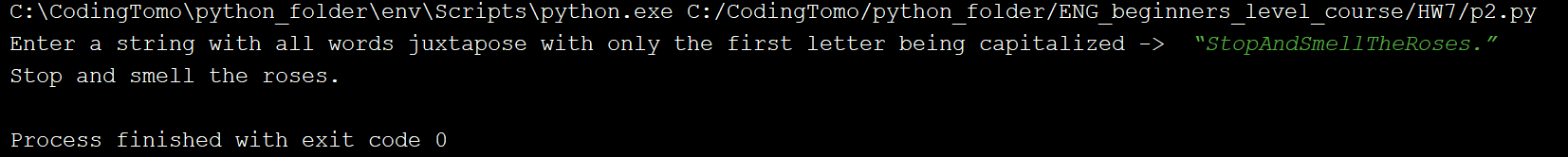
1. **(15 points, Word Separator)** Write a Python program that accepts as input a sentence in which all of the words are run together, but the first character of each word is uppercase. Convert the sentence to a string in which the words are separated by spaces, and only the first word starts with an uppercase letter. For example, the string “StopAndSmellTheRoses.” would be converted to “Stop and smell the roses.”

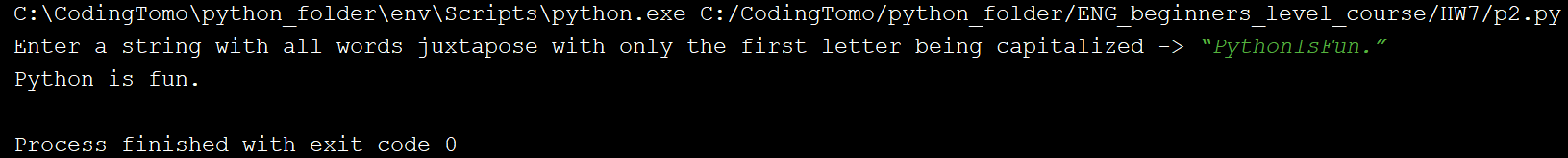
**(Use “StopAndSmellTheRoses.” And “PythonIsFun.” to test your code.)**

**CODE:**

## PROBLEM #2  
# NAME: Tomoki Koike  
# DUE: 4/21/2019  
# DESCRIPTION: This program will take in a string with a sentence in which  
# words are run together and this program will break the run-together words  
# into a complete grammatical sentence. (with input validation)  
# STAND: Class of 2020  
##  
  
# Functions  
# Function for input validation  
def getValid(prompt):  
 while True:  
 try:  
 # trying input with without any conditions at first  
 this = input(prompt)  
 except ValueError:  
 # Prints the user to input again since the input was not valid  
 print('Sorry, could not understand. Please enter again.')  
 continue  
 if str(this):  
 # Valid input  
 break  
 else:  
 # Invalid input  
 print('Error. Please enter a string.')  
 continue  
 return this  
  
# # Function to create a list with all the capitalized alphabets  
# def create\_cap\_alphabet\_list():  
# # Preallocate the list  
# cap\_alphabet = []  
# for x in range(ord('A'), ord('Z')+1):  
# # Append the capital alphabet to list  
# cap\_alphabet.append(chr(x))  
# return cap\_alphabet  
  
# Function to go through the input go through the string and  
# identify the positions of the capitalized alphabets to separate each word apart  
def word\_separater(string):  
 # Tags to keep the first position of the capital letter and the end of the word  
 tag\_i = **0** tag\_f = **0** # Counter to keep track of the number of capital letters that it has encountered so far  
 counter = **0** # Preallocate a list to store all the words of the run together sentence  
 word\_holder = []  
 for x in range(len(string)):  
 if string[x].isupper() or string[x] == '.':  
 counter += **1** if counter == **1**:  
 tag\_i = x  
 else:  
 tag\_f = x  
 sliced\_string = string[tag\_i:tag\_f]  
 word\_holder.append(sliced\_string.lower())  
 tag\_i = tag\_f  
 counter -= **1** # Capitalize the first word  
 word\_holder[**0**] = word\_holder[**0**].capitalize()  
 # Join the strings in the word holder  
 sentence = ' '.join(word\_holder)  
 # Append a period at the end of the sentence  
 sentence = sentence + '.'  
 return sentence  
  
# Main function  
def main():  
 string = getValid('Enter a string with all words juxtaposed with only the first letter being capitalized -> ')  
 sentence = word\_separater(string)  
 # Printing the result  
 print(sentence)  
 return **0**# Execute main function  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**SAMPLE EXECUTIONS:**





1. **(15 points, PowerBall Lottery)** To play the PowerBall lottery, you buy a ticket that has five numbers in the range of 1-69, and a “PowerBall” number in the range of 1-26. (you can pick the numbers yourself, or you can let the ticket machine randomly pick them for you.) Then, on a specified date, a winning set of numbers is randomly selected by a machine. If your first five number match the first five winning numbers in any order, and your PowerBall number matches the winning PowerBall number, then you win the jackpot, which is very large amount of money. If your numbers match only some of the winning numbers, you win a lesser amount, depending on how many of the winning numbers you have matched.

In the attachment, you’ll find a file named “**pbnumbers.txt**”, containing the winning PowerBall numbers that were selected between February 3, 2010 and May 11, 2016(The file contains 654 sets of winning numbers.) Each line in the file contains the set of six numbers that were selected on a given date. The number are separated by a space, and the last number in each line is the PowerBall number for that day. For example, the first line in the file shows the numbers for February 3, 2010, which were 17, 22, 36, 37, 52, and the PowerBall number 24.

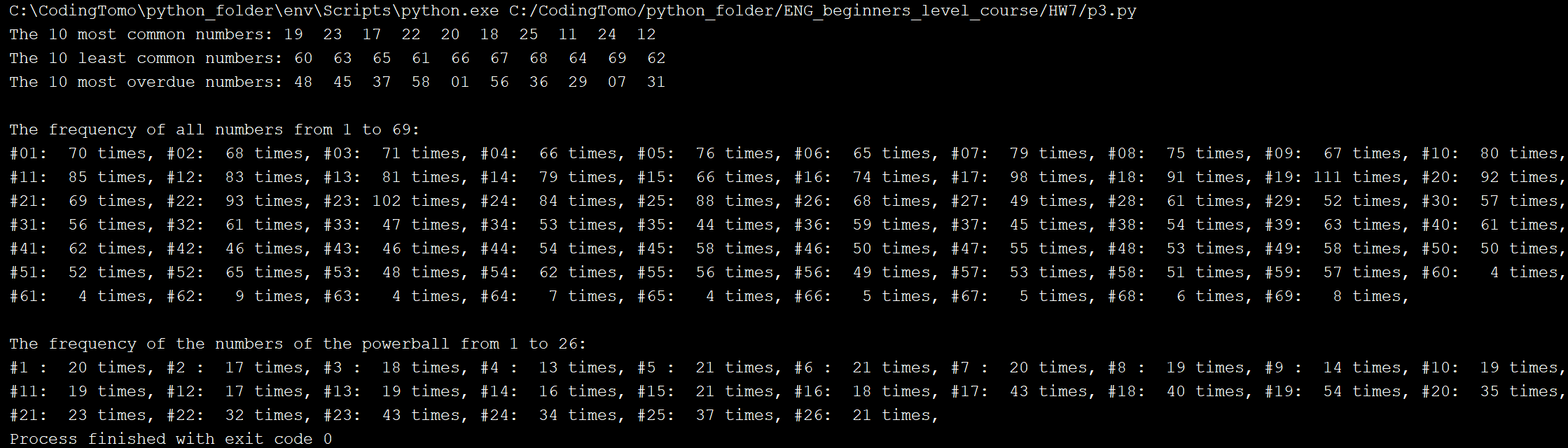
Write one Python program that work with this file to perform the following:

* Display the 10 most common numbers, ordered by frequency
* Display the 10 least common numbers, ordered by frequency
* Display the 10 most overdue numbers (numbers that haven’t been drawn in a long time), ordered from most overdue to least overdue
* Display the frequency of each number 1-69, and the frequency of each Powerball number 1-26

**CODE:**

## PROBLEM #3  
# NAME: Tomoki Koike  
# DUE: 4/21/2019  
# DESCRIPTION: This program will take in text file with data of the  
# PowerBall Lottery and there winning numbers of a specific period.  
# This program will manipulate the data to output the frequency of the  
# winning numbers and provide the most frequent numbers etc.  
# (with input validation)  
# STAND: Class of 2020  
##  
  
# import Modules  
import collections  
  
# Functions  
# Function to read the text file  
def read\_file\_all(file):  
 # Opening the file to read  
 afile = open(file**,** 'r')  
 # Reading the first line  
 aline = afile.readline()  
 # Initialize a list to contain all the values from the data  
 all\_nums = []  
 # Loop to read all the lines  
 while aline != '':  
 # stripping the sentence end  
 aline = aline.rstrip('\n')  
 # Splitting the values into individual strings  
 aline = aline.split()  
 # Append to the list  
 for x in aline:  
 all\_nums.append(x)  
 # Read the next line  
 aline = afile.readline()  
 # Closing the open file  
 afile.close()  
 return all\_nums  
  
# Function to read the text file  
def read\_file\_powerball(file):  
 # Opening the file to read  
 afile = open(file**,** 'r')  
 # Reading the first line  
 aline = afile.readline()  
 # Initialize a list to contain all the values from the data  
 powerball\_nums = []  
 # Loop to read all the lines  
 while aline != '':  
 # stripping the sentence end  
 aline = aline.rstrip('\n')  
 # Splitting the values into individual strings  
 aline = aline.split()  
 # Append to the list  
 powerball\_nums.append(int(aline[**5**]))  
 # Read the next line  
 aline = afile.readline()  
 # Closing the open file  
 afile.close()  
 return powerball\_nums  
  
# Function to read the text file and create dict to find the overdues  
def read\_file\_overdue(file):  
 # Opening the file to read  
 afile = open(file**,** 'r')  
 # Reading the first line  
 aline = afile.readline()  
 # initializing the counter to check for the overdue  
 time = **654** adict = {}  
 # Loop to read all the lines  
 while aline != '':  
 # stripping the sentence end  
 aline = aline.rstrip('\n')  
 # Splitting the values into individual strings  
 aline = aline.split()  
 # Calling the function to create dict to figure overdues  
 make\_dict\_overdue(aline**,** time**,** adict)  
 # decrement time  
 time -= **1** # Read the next line  
 aline = afile.readline()  
 # Closing the file  
 afile.close()  
 return adict  
  
# Function to create a dictionary for the frequency of the winning numbers  
def make\_dict\_freq(all\_nums):  
 # Create a empty dict  
 adict = {}  
 # Assigning the value in list as key  
 for x in range(len(all\_nums)):  
 key = all\_nums[x]  
 if key in adict:  
 adict[key] += **1** else:  
 adict[key] = **1** return adict  
  
# Function to create the overdue dict  
def make\_dict\_overdue(aline**,** time**,** adict):  
 for key in aline:  
 adict.update({key: time})  
 return adict  
  
# Function to sort dict from small to large  
def dict\_sort(adict):  
 return collections.OrderedDict(sorted(adict.items()**,** key=lambda kv: kv[**1**]))  
  
# Function to sort dict in reverse large to small  
def dict\_sort\_rev(adict):  
 return collections.OrderedDict(sorted(adict.items()**,** key=lambda kv: kv[**1**]**,** reverse=True))  
  
# Function to print results  
def output\_dict(adict):  
 counter = **0** for k in adict:  
 if counter == **10**:  
 break  
 else:  
 print(k**,** ' '**,** end='')  
 counter += **1** return  
  
# Function to print out the dictionary with all he numbers as their frequency  
def print\_freq(adict):  
 counter = **0** for x in sorted(adict):  
 print('#{0:<2}: {1:3} times, '.format(x**,** adict[x])**,** end='')  
 counter += **1** if counter % **10** == **0**:  
 print()  
 return  
  
# Main function  
def main():  
 all\_nums = read\_file\_all('pbnumbers.txt')  
 overdue\_dict = read\_file\_overdue('pbnumbers.txt')  
 freq\_dict = make\_dict\_freq(all\_nums)  
 powerball\_nums = read\_file\_powerball('pbnumbers.txt')  
 powerball\_freq = make\_dict\_freq(powerball\_nums)  
  
 # Results  
 print('The 10 most common numbers:'**,** end=' ')  
 output\_dict(dict\_sort\_rev(freq\_dict))  
 print()  
 print('The 10 least common numbers:'**,** end=' ')  
 output\_dict(dict\_sort(freq\_dict))  
 print()  
 print('The 10 most overdue numbers:'**,** end=' ')  
 output\_dict(dict\_sort\_rev(overdue\_dict))  
 print('\n')  
 print('The frequency of all numbers from 1 to 69:')  
 print\_freq(freq\_dict)  
 print('\n')  
 print('The frequency of the numbers of the powerball from 1 to 26:')  
 print\_freq(powerball\_freq)  
  
 return **0**# Executing the main function  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**EXECUTION:**



1. **(15 points Pet Class)** Write a class named Pet, which should have the following data attributes:

* \_ \_name (for the name of a pet)
* \_ \_animal\_type (for the type of animal that a pet is. Example values are ‘Dogs’, ‘Cat’, and ‘Bird’)
* \_ \_age (for the pet’s age)

The **Pet** class should have an \_ \_init\_ \_method that creates these attributes. It should also have the following methods:

* set\_name

This method assigns a value to the \_ \_name field.

* set\_animal\_type

This method assigns a value to the \_ \_animal\_type field

* set\_age

This method assigns a value to the \_ \_age field

* get\_name

This method returns the value of the \_ \_name field

* get\_animal\_type

This method returns the value of the \_ \_animal\_type field

* get\_age

This method returns the value of the \_ \_age field

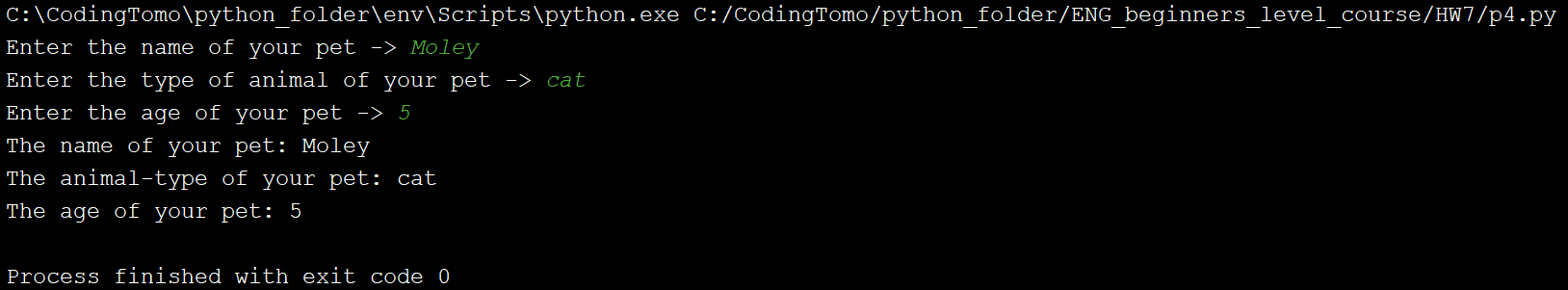
Once you have written the class, write a program that creates an object of the **Pet** class and prompts the user to enter the name, type, and age of his or her pet. This data should be stored as the object’s attributes. Use the object’s accessor methods to retrieve the pet’s name, type, and age and display this data on the screen.

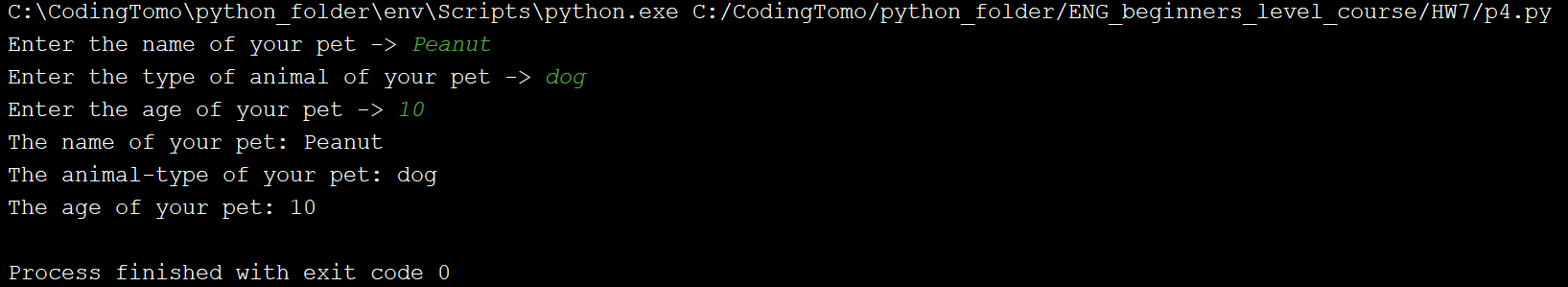
**(use ‘Moley’, ‘cat’, ‘5; ‘Peanut’, ‘dog’, ‘10’ to test your code.)**

**CODE:**

## PROBLEM #4  
# NAME: Tomoki Koike  
# DUE: 4/21/2019  
# DESCRIPTION: This program will program will create a class  
# named pet and access it from the main function to give the  
# outputs  
# STAND: Class of 2020  
##  
  
# Class  
class Pet:  
 # Data attriubutes  
 def \_\_init\_\_(self**,** name**,** animal\_type**,** age):  
 # Name of the pet  
 self.\_\_name = name  
 # Type of animal of the pet  
 self.\_\_animal\_type = animal\_type  
 # Age of the pet  
 self.\_\_age = age  
  
 # Setting value to name  
 def set\_name(self**,** name):  
 self.\_\_name = name  
  
 # Setting value to animal\_type  
 def set\_animal\_type(self**,** animal\_type):  
 self.\_\_animal\_type = animal\_type  
  
 # Setting value to age  
 def set\_\_age(self**,** age):  
 self.\_\_age = age  
  
 # Output name  
 def get\_name(self):  
 return self.\_\_name  
  
 # Output animal type  
 def get\_animal\_type(self):  
 return self.\_\_animal\_type  
  
 # Output age  
 def get\_age(self):  
 return self.\_\_age  
  
# Main function  
def main():  
 # Gathering user inputs  
 aname = input('Enter the name of your pet -> ')  
 atype = input('Enter the type of animal of your pet -> ')  
 aage = input('Enter the age of your pet -> ')  
 # Creating object  
 apet = Pet(aname**,** atype**,** aage)  
 # Printing out the results  
 print('The name of your pet: {0}'.format(apet.get\_name()))  
 print('The animal-type of your pet: {0}'.format(apet.get\_animal\_type()))  
 print('The age of your pet: {0}'.format(apet.get\_age()))  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**SAMPLE EXECUTIONS:**





1. **(15 points Car Class)** Write a class named Car that has the following data attributes:
   * \_ \_year\_model (for the car’s year model)
   * \_ \_make (for the make of the car)
   * \_ \_speed (for the car’s current speed)

The **Car** class should have an \_ \_init\_ \_ method that accepts that car’s year model and make as arguments. These values should be assigned to the object’s \_ \_year\_model and \_ \_make data attributes. It should also assign 0 to the \_ \_ speed data attribute.

The class should also have the following methods:

* accelerate

The accelerate method should add 5 from the speed data attribute each time it is called.

* brake

The brake method should subtract 5 from the speed data attribute each time it is called.

* get\_speed

The get\_speed method should return the current speed.

Next, design a program that creates a **Car** object then calls the **accelerate** method five times. After each call to the **accelerate** method, get the current speed of the car and display it. Then call the **brake** method five times. After each call to the **brake** method, get the current speed of the car and display it.

**CODE:**

## PROBLEM #5  
# NAME: Tomoki Koike  
# DUE: 4/21/2019  
# DESCRIPTION: This program will declare a class called car  
# and will compute the current velocity of the car accepting the  
# make and model of the car.  
# STAND: Class of 2020  
##  
  
# Class  
class Car:  
 # Data atributes  
 def \_\_init\_\_(self**,** year\_model**,** make):  
 self.\_\_year\_nodel = year\_model  
 self.\_\_make = make  
 self.\_\_speed = **0** # Accelerating function  
 def accelerate(self):  
 self.\_\_speed += **5** return self.\_\_speed  
 # Brake function  
 def brake(self):  
 self.\_\_speed -= **5** return self.\_\_speed  
 # Accessors  
 def set\_year\_model(self**,** year\_model):  
 self.\_\_year\_nodel = year\_model  
 def set\_make(self**,** make):  
 self.\_\_make = make  
 def set\_speed(self**,** speed):  
 self.\_\_speed = speed  
 # Mutators  
 def get\_year\_model(self):  
 return self.\_\_year\_nodel  
 def get\_make(self):  
 return self.\_\_make  
 def get\_speed(self):  
 return self.\_\_speed  
  
def main():  
 # Accept the user input for the model and make of the car  
 year\_model**,** make = input('Enter the year-model and the make of the car -> ').split()  
 # Create object of Car class  
 acar = Car(year\_model**,** make)  
 # Looping to accelerate the car 5 times  
 for x in range(**5**):  
 acar.accelerate()  
 print('The current speed of the {0} {1}: {2} mph'.format(acar.get\_make()**,** acar.get\_year\_model()**,** acar.get\_speed()))  
 # Looping to brake the car 5 times  
 for y in range(**5**):  
 acar.brake()  
 print('The current speed of the {0} {1}: {2} mph'.format(acar.get\_make()**,** acar.get\_year\_model()**,** acar.get\_speed()))  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**SAMPLE EXECUTION:**

