CISC 140: Lab 2

Contents

1 Introduction

This programming lab relies on the fundamentals learned and tested in Lab 01 while designing and implementing your solution as Python objects. This lab will require you to both design and implement several objects and then use them to solve a problem. You will also learn how to manipulate and handle CSV, or comma-separated value, files.

The project for this lab is based upon several of the Creative Exercises from Section 3.2 in the textbook. The general problem statement is as follows:

> Compose a data type Element for entries in the periodic table of elements. Include data-type values for the element, atomic number, symbol, and atomic weight and accessor methods for each of these values. Then, compose a data type PeriodicTable that reads values from a file to create an array of Element objects and responds to queries from standard input so that a user can type a molecular formula like H2O and have the program respond by writing the molecular weight. Develop APIs and implementations for each data type.

2 Designing and Creating Your Element Object

To begin, we need to understand what we are designing. As you have learned in Scientific Mind, an element is a substance that cannot be chemically broken down into a simpler substance. At time of writing, there are 94 elements which naturally occur on Earth and another 24 which we can artificially synthesize in a laboratory. As mentioned above, each element has a set of signatures, some natural and some artificially, which can uniquely identify or describe the element. Some of these features are briefly described below:

Name
The name of the element, which is used both in common conversation or scientific writing.

Examples: Hydrogen, Oxygen, Gold, Lead, Uranium

Atomic Number
The number of protons in the element's atomic nucleus.

Examples: 1, 8, 79, 82, 92

Symbol
The one or two letter abbreviation used as a short-hand for the element. This can be used many places, but especially in chemical formulas.

Examples: H, O, Au, Pb, Ur

Atomic Weight
The average mass of an atomic nucleus of this element.

Examples: 1.008, 15.999, 196.97, 207.2, 238.03

Period
The row that the element belongs to on the periodic table.

Examples: 1, 2, 6, 6, 7

Group
The column that the element belongs in on the periodic table. Note that not every row has every column/period and that some elements fall outside of a formal "group" designation.

Examples: 1, 16, 11, 14, n/a

Category
A broad classification of elements where elements in the same category share similar physical and chemical properties

Examples: Reactive nonmetal, Reactive nonmetal, Transition metal, Post-transition metal, Actinide

Phase

The state of matter the element takes at 0°C and 1 atmosphere.

Examples: Gas, Gas, Solid, Solid, Solid

Designing Instance Variables **5pts.**

What instance variables will you need to declare in order to adequately define the Element class? What should be the data type you should expect in each instance variable, based on the data to be stored within it? The above definition of features is sufficient to define what you need but you may feel free to open the accompanying periodic-table.csv file as well.

the instance variables that will need to be declared are self._, atomic_number, name, symbol ,group, period, element_category, atomic_weight, phase

The expected data type of each of the variables will be a string as the information from the csv file will be together for a specific element

Constructor/Initializer Design **5pts.**

Now that we have out instance variables designed, we need to design our constructor or initializer. What parameters should the constructor take? Should any of these be parameters be optional? Why or why not?

the parameters the initializer will take are atomic_number,

name, symbol ,group, period, element_category, atomic_weight, phase

since these are what we are working with.

They should all be used but the only one i can think of that would be

optional is the group variable since some of the elements dont have a

group, this is why i have if the element does not have to state that.

Method Design **5pts.**

The original question specified that each instance variable should have an accompanying accessor method, however it did not specify anything about mutator methods. Should our Element class have mutator methods? If so, what methods and why? If not, why not? Will you need any private helper methods? Why or why not?

each variable will have a get accessor method to be able to pull the specific data

then it returns the data. I do not have any mutator methods since they are only pulling

the data, so inside of each accessor method there is only a (self)

The only variable that has a helper function will

be the group variable since some of the elements do not have a group. and

its if group is none return that element has no group, else return the group name

Implement Element

Implement the Element class within the file element.py as designed above. As the original question specifies, you should implement an accessor method for each of you instance variables. For the time, you may assume that any data passed into the Element object is of the appropriate type. **10pts.**

Test Element

Write code that will appropriately create and test your Element class. This code should be included in an appropriate if statement (if __name__== "__main__") within the element.py file. **10pts.**

3 Reading a CSV file

As a brief aside, we need to discuss how to read in data from our CSV file. A CSV file is a comma-separated value file and, as the name implies, each value in a row is separated (delineated) by a comma.

There are different variations of CSV file. One common form of CSV file will have all of their values also surrounded by double quotation marks (") in order to explicitly delineate what is the value. An alternate format will only have the double quotation marks surrounding values that contain commas.

File Analysis **5pts.**

Open the accompanying CSV file periodic-table.csv in your text editor of choice. Explain the format of the file in detail, including which of the two above mentioned formats this file is in, the number of rows and columns, what data is contained within the file, and if there is any missing data.

The csv file for the periodic table contains all the information about each element, the atomic_number, name, symbol, group, period, element_category, atomic_weight, phase. There are also a total of 119 rows including the first row which are the column names. there are a total of 8 columns. The csv file is the second variation since the information for each row is just separated by commas and the whole row does not have any type of quotation marks.

There is also no data missing from the file everything is there even an n/a for the elements with no group that we talked about before. so the file even handled that. i did notice that the csv file vales have no type of quatation marks anywhere so i just went with the second variation since it stated commas which the csv file does have.

File Reading Algorithm **5pts.**

Write an in-depth algorithm for how you will implement reading from the periodic-table.csv file. You may choose what the resulting output for the algorithm is or methodology for consuming this information in the PeriodicTable object later on if you wish. (Hint: If you don't know what to do or how to do it, consider a two-dimensional array. There are other acceptable solutions, but this would be fine. Just be sure to PROGRAMMATICALLY figure out the number of rows and columns in the file before creating your two-dimensional array.)

For the algorithm i will first start with the instream so that the file program can read the text file. Next the it has a while for getting the next row and reading it an having a +1 for and another while loop for the read split and store the elements data by using quotation marks and commas and storing them into the array and then finally printing out the whole array which will be the 119 lines including the first row for headers.

4 Designing and Creating Your PeriodicTable Object

Finally, we are at the point where we define our PeriodicTable object! This object will digitally represent the periodic table of all known elements and will contain data on each element. We are assuming that the data will be contained in the periodic-table.csv CSV file. However, we do want to allow ourselves some flexibility in source file so another file name could be specified at construction as long as the file was in the exact same format (meaning CSV format with the same number of columns in the same order).

Designing Instance Variables **5pts.**

What instance variables will you need to declare in order to adequately define the PeriodicTable class? What should be the data type you should expect in each instance variable, based on the data to be stored within it? Remember that he PeriodicTable class needs to store and find data from the elements specified within a given CSV file.

the instance variables that will need to be declared are self._, atomic_number,

name, symbol ,group, period, element_category, atomic_weight, phase

The expected data type of each of the variables will be a string as the

information from the csv file will be together for a specific element

Constructor/Initializer Design **5pts.**

Now that we have out instance variables designed, we need to design our constructor or initializer. What parameters should the constructor take? Should any of these be parameters be optional? Why or why not?

The parameters will be self._1-118 and each parameter will equal an element have all of the data stored in a string, while still using the csv file of course but this way its reassured and it will help for using the search algorithm and user input.

Query Algorithm **10pts.**

Remember that the original question states that the PeriodicTable data type should be able to “respond to queries from standard input so that a user can type a molecular formula like H2O and have the program respond by writing the molecular weight.” Write an in-depth algorithm for defining this process, including what would happen if an invalid equation was entered. You treat the word “exit” as valid input, resulting in the termination of the program.

Well for this i will be using a search algorithm that is looking for the element name and will ask the user for an element they want to see. the search algo will look for the element in the text and retrieve it, and since its also hard coded in using strings it will return the whole string. So all the information for the element will be shown

Method Design **5pts.**

We need to do now is determine what methods we will need to define in our PeriodicTable class in order to complete the above functionality. List all methods below, including their purpose/functionality, their classification (accessor or mutator), and their visibility (public or "private").

each variable will have a get accessor method to be able to pull the specific data

then it returns the data. I do not have any mutator methods since they are only pulling

the data, so inside of each accessor method there is only a (self). and these are accessor.

Implement PeriodicTable

Implement the PeriodicTable class within the file periodic_table.py as designed above. Implement your initializer/constructor and all methods as designed. (Hint: Don't forget to refer back to and use your file reading algorithm from section 3! Additionally, if you have questions about how to read from a file, you might want to refer to Section 3.1, pages 380 – 387.) **10pts.**

10pts.

Test PeriodicTable

Write code that will appropriately create and test your PeriodicTable class. This code should be included in an appropriate if statement (if __name__== "__main__") within the periodic_table.py file. However, this code should only run if the following command is given to run periodic_table.py at the command prompt:

```
>> python periodic_table.py -test
```

Implement PeriodicTable Query Functionality

Any other command line parameters, including no parameters, should result in the program entering into its "query functionality" aspect. Implement your algorithm as defined above. Be sure to provide the user with reasonable on-screen prompts for any input the program expects and any outputs the program gives. **10pts.**

5 Rubric

Question	Points	Score
Element Instance Variable Design	5	
Element Constructor Design	5	
Element Method Design	5	
Implement Element	10	
Test Element	10	
File Analysis	5	
File Reading Algorithm	5	
PeriodicTable Instance Variable Design	5	
PeriodicTable Constructor Design	5	
PeriodicTable Query Algorithm	10	
PeriodicTable Method Design	5	
Implement PeriodicTable	10	
Test PeriodicTable	10	
Implement PeriodicTable Query Functionality	10	
Total:	100	