Lab 3: Inheritance and Composition

Due: January 30, 9 p.m.

1 Introduction

Back to Municipal Trees for this lab!!

Last time we saw trees, we represented them as lists of lists in our code. This time, we will be representing trees as Software Objects and associating these objects with attributes and methods. In addition, we will be using inheritance to implement several different kinds of trees with varying properties and methods. Finally, we will touch on the idea of composition in this lab, as our representations of trees will contain *Location* objects to store longitude and latitude coordinates.

As before, you will find Lab 3 starter files on eClass and you can submit your completed code to this PrarieLearn URL:

https://us.prairielearn.com/pl/course_instance/200825/assessment/2625072

The file in the starter package is as follows:

user_code.py

Each method you must implement is described in brief below; some unit tests (e.g. doctests) have been provided for you with the starter code, as in prior labs.

2 Programming Task

The code for this assignment relates to Beech and White Ash Trees, which are well represented in our city tree inventory. Birch trees sequester more carbon per kilogram than White Ash. The amount of carbon in each tree can be estimated based on its diameter at breast height (DBH) and using allometric equations. Different tree species have different allometric parameters, which result in different calculations of carbon per kilogram of wood.

In this lab, you will implement a MunicipalTree class to represent the "average" tree, and two sub-classes of tree (Beech and White Ash). You will implement a default method to calculate the carbon content of an "average" tree based on "average" allometric parameters. You will then override properties of the MunicipalTree class in your Beech and White Ash implementations, to include allometric coefficients that are specific to these species.

More specifically, you will implement:

- A $MunicipalTree$ constructor, i.e.: $def\_\_init\_\_(self, type : str, diameter : str, owner : str, name : str, longitude : str, latitude : str)- > None$. This will initialize the various attributes of a tree according to input parameters, which are all strings. You will have to convert input parameters to the tree's attribute types. The diameter of a MunicipalTree, for example, should be stored as an integer, which the longitude and latitude will be stored as a *Location* object. Default allometric coefficients have been provided in your starter code.

- The method *calculate_carbon_content*, in the $MunicipalTree$ class. This will calculate carbon content (in kgs) stored in the wood of a tree based on its diameter and allometric coefficients. The equation you will use to calculate carbon content is in the comments of your starter code.

- A $WhiteAsh$ constructor, i.e.: $def\_\_init\_\_(self, type : str, diameter : str, owner : str, name : str, longitude : str, latitude : str)- > None$. A WhiteAsh is a MunicipalTree, yet has different allometric coefficients. This means that calls to *calculate_carbon_content* should yield a different result for an "average" tree than for a $WhiteAsh$.

- A *Beech* constructor, i.e.: $def\_\_init\_\_(self, type : str, diameter : str, owner : str, name : str, longitude : str, latitude : str)- > None$. Like the White Ash, a Beech is also a MunicipalTree, yet it also has different allometric coefficients and stores a different amount of carbon in its wood.

- The 'magic' method *__eq__*, in the *Location* class. This will compare two Locations to see if they are the same. If the two Locations have the same longitude and latitude coordinates, consider them to be 'equal'.

3 Testing and Submitting

We once again ask that you write a doctest for the methods you implement. When you write your own tests, consider edge and corner cases. What happens when we execute *calculate_carbon_content* on a tree with a diameter of 0? Make sure you can pass all of the tests we have provided as well as those you create before you submit.

To submit, you will cut and past the code you write in *user_code.py* into the Lab 3 instance of PrarieLearn. The score you get on PrarieLearn is the score you get for the lab!