EECS Lab 2: Your First Object

Due: January 23, 9 p.m.

1 Introduction

Welcome to Lab number 2!

This week's lab will be the first one in which you construct a software object. We'll be working with city data again, but this time we won't be looking at a tree inventory. Instead, we'll be looking at date from an inventory of sewer infrastructure! This data once again can be found in the Mississauga repository of open city data, at this URL:

https://data.mississauga.ca/datasets/storm-sewer-network-storm-node

What do cities do with this kind of data? Just like tree inventories, sewer inventory data informs municipal decisions related to, for example:

- Maintenance planning. Information on the condition of sewers helps prioritize maintenance efforts.
- Capacity planning. Inventory data helps cities know if they can handle weather events without causing flooding.
- Water quality management. Inventory data helps assess the impact of runoff on the quality of lake water.

In this lab you will be constructing *Sewer* objects using sewer inventory data, which means you will be working on a *Sewer* class. You will also override some of the 'magic' methods that are associated with this class. Remember that you can use the *dir* command to list all methods associated with a given class. To list methods associated with the class *Sewer*, then, type $print(dir(Sewer))$ in the file where your *Sewer* implementation resides.

As before, you will find Lab 2 starter code (*user_code.py*) on eClass and you can submit your completed code via PrarieLearn:

https://us.prairielearn.com/pl/course_instance/200825/assessment/2625071

2 Programming Task

The methods you must implement are described in brief below; more details and some doctests can be found in the starter code. We ask that you implement:

1. The *Sewer* class constructor:
   $\_\_init\_\_(self, asset\_id, asset\_type, install\_date, works\_yard, ward, x\_coord, y\_coord)$.
   Input arguments are all strings; you must convert them to the types of *Sewer* attributes. The 'asset_id' must be converted to an int, 'lon' and 'lat' must be converted to floats, and install_date must converted to a 'date' object. The format of strings used to define a date will always be 'YYYY/MM/DD' and you can create a date object using this method: $date(year, month, day)$.

2. The class method $\_\_lt\_\_(self, other)$. This method, which will override the 'magic' $\_\_lt\_\_$ method, will compare one *Sewer* ($self$) to an input *Sewer* ($other$). Consider a sewer that to be 'less than' another if its asset_id is **less than** that of the other sewer.

3. The class method $\_\_str\_\_(self)$. This method, which will override the 'magic' $\_\_str\_\_$ method, will create a string representation of a *Sewer* ($self$). Use this format: "$ID < asset\_id >: < asset\_type >$". An example follows: "*ID 2030569: Double Catchbasin*".

4. The method $construct\_sewers(sewers)$. The input argument ($sewers$) will be a list of lists of strings; each list of strings represents one *Sewer*. The method should return a list of *Sewer* objects, one for each list of strings.

5. The method $sewers\_in\_boundary(items, boundary\_box)$. This method should scan list of Sewer objects ($items$) and return a list of booleans; each boolean indicates if the location of the corresponding Sewer (i.e. its longitude and latitude coordinates) lie inside the boundary box. The boundary box will be defined a rectangle using four floating point numbers, as follows: $[min\_longitude, min\_latitude, longitude\_span, latitude\_span]$. Consider using *map* and *filter* to implement this method efficiently.

3 Testing and Submitting

We once again ask that you write doctests for the methods you implement. When you write tests, consider edge and corner cases. What happens when the *boundary_box* argument to $sewers\_in\_boundary(items, boundary\_box)$ has a length or width of 0? Are you confident that your code will still work? 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 2 instance of PrarieLearn. The score you get on PrarieLearn is the score you get for the lab!