HW6

JAY LAURA

1. What have you learned?

This course has been a good refresher on basic Python programming. I have learned a series of things in completing the first five homework assignments. I have previously not used `zip` or `map` extensively in my work. The use of these built-in functions, along with lambda functions, has provided additional practice implementing one line, functional statements. The use of zip inspired me to look for a vectorized `NumPy` version and led to the discovery of either the `meshgrid` function or a recursive cartesian product function.

The lessons have also provided a good review of OOP using Python. The majority of my own work is functional programming and the nudge to program using classes outside of web development has been positive.

2. Weaknesses and Areas for Improvement

I would like to continue to improve my fluency with `NumPy`, begin to explore the use of some of the `SciPy` KDTree tools and learn more about network based analysis. I am also looking forward to examining spatial databases and learning about robust data querying techniques. My greatest weakness so far has been my desire to write functional code! I could definitely include more classes and leverage them, but I prefer the atomic decomposition of functionality over the segmentation of functionality into classes. I find this easier to debug and profile.

I also could invest more time in developing some higher level code profiling tools. Currently, the `wait.pid` calls, when profiling multiprocessing work provide the extent of my profiling capabilities. For simple parallelization this is not an issue as a single function is generally called and `wait.pid` is a surrogate for the processing. In more complex implementations this is not the case and more robust profiling is required. This is not something that I expect to cover in class, but this homework provides a great opportunity to document that weakness in my development toolkit.

3. Emphasis

Over the remainder of the course I would like to emphasize implementing a serial P-Compact-Regions algorithm in PySAL and exploring options for parallelization. I have not yet implemented a parallel simulated annealing algorithm and the prospect is exciting. For example, is it better to run multiple concurrent searches or determine efficient

techniques to divide the local swap phase over multiple cores. Does the problem size impact decomposition methods? Is it possible to utilize a generalized Tabu Search to solve P-Compact-Regions? What impact does that have on processing time and solution quality?

I would also love to explore the implementation of a WPS, as the web processing I have implemented does not comply with OGC standards. It would be wonderful therefore to gain some experience rolling out processing that is accessible via a webUI and a desktop GIS (QGIS).

4. Concerns

At this time I do not have any concerns. I am excited to begin working on the final project and hope that we can publish our results once an implementation is generated, debugged, and tested.

5. UML

Figure 1, below illustrates three classes that I could have used in the previous homework assignments. These mirror the OGC standard classes. The NoCLASS class is a compromise illustrating the functional code that is not wrapped in a class. The difference between this functional paradigm can be distilled to the difference between encapsulation of functionality in a class or encapsulation of functionality in the smallest possible atomic unit. In implementation, a functional program is not likely to be diagrammed using UML, but a type list.

All attributes and methods are public. Is is not possible to make a method private without performing code gymnastics. Even then, it is still possible to access all class methods.

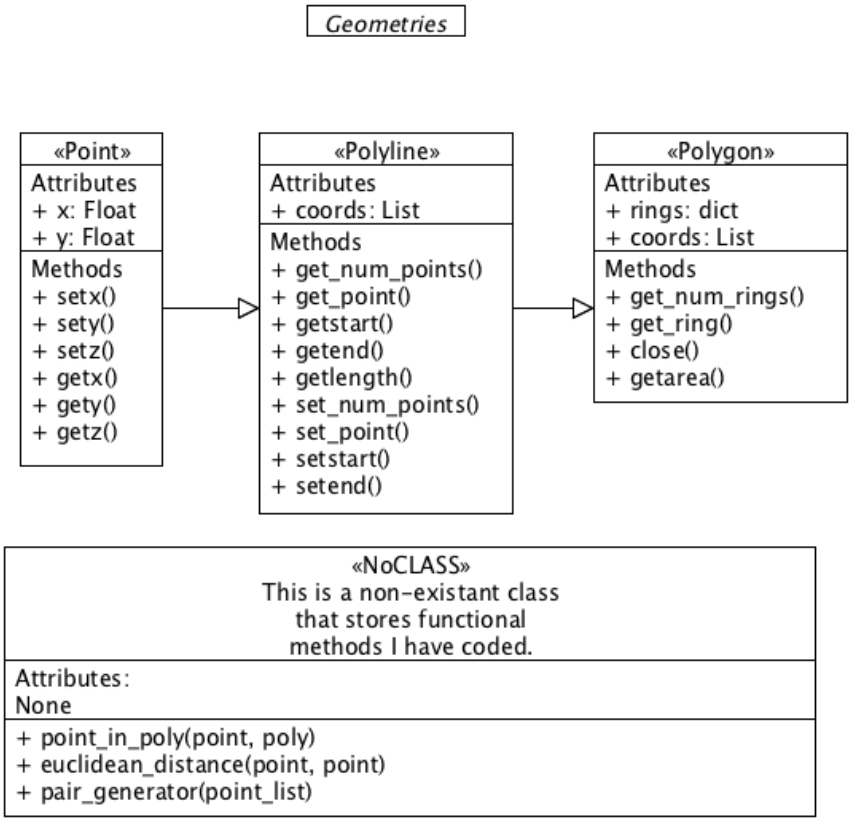

FIGURE 1. A UML diagram showing a potential implementation of the previous homework assignments using OOP.