CSCI 220L

Lab 11

Learning Objectives

- Practice modifying and designing classes
- Continue practicing while loops and file reading

Activities

Complete the following activities in the indicated files. For each activity, demonstrate your team's working solution to one of the instructors and then switch roles with your teammate. Make sure to include meaningful comments, docstrings, and output messages to receive full credit.

1. Introduction to classes

Examine the `Date` class in the provided file `date.py`. Then, complete the following exercises:

a. Pick two dates of your choice. In `main()`, create two `Date` objects representing those dates and assign them to variables. Print the first object, and print the second object's month using its getter method.

b. The `Date` class is currently immutable. Add setter methods for each attribute to make it mutable. Then, in `main()`, modify the first object's year using its new setter and print it again to show that it was updated.

c. There is currently no input validation in the `Date` constructor or the new setters. Modify these methods so that if a specified day is outside [1, 31] or a specified month is outside [1, 12], the value will default to 1. Test your changes in `main()`.

Hint: To avoid duplicate validation code between the constructor and setter methods, you may put this new code only in the setters. Then, call the setters from the constructor.

2. Creating a `Point` class

Create a file `point.py` that defines a class `Point` representing an (x,y) point in a Cartesian coordinate plane. The `Point` class should have the following features:

- Two attributes of type `float` representing the x and y coordinates
- A constructor that initializes the x and y values to 0
- Getter methods `get_x()` and `get_y()`
- Setter methods `set_x(x)` and `set_y(y)`
- A `__str__()` method that returns a string of the form `"(x,y)"`

Use the provided `test_point.py` file to test your implementation after each step.

3. Creating a `Circle` class

Create a file `circle.py` that defines a class `Circle` representing a circle defined by a center point and radius. You should use the `Point` implementation you created in Activity 1 by importing it into `circle.py`. The `Circle` class should have the following features:

- An attribute of type `Point` representing the center
- An attribute of type `float` representing the radius

- A constructor that accepts an x value, y value, and radius and initializes the center point and radius attributes
    - If the specified radius is negative, it should be initialized to 0
- Getter methods `get_center()` and `get_radius()`
- Setter methods `set_center(x, y)` and `set_radius(r)`
    - If the specified radius is negative, it should be reassigned to 0
- A method `get_area()` that computes and returns the circle's area
- A `__str__()` method that returns a string of the form:

```
Circle:
    Center: <(x,y)>
    Radius: <radius>
    Area:   <area>
```

Use the provided `test_circle.py` file to test your implementation after each step.

4. Finding neutron stars

You're an astrophysicist searching for neutron stars. A pulsar is a type of neutron star that occasionally emits strong bursts of energy in a phenomenon called the lighthouse effect. If we point our research satellite at a particular point and record the incoming signals, we can hopefully detect a recurring strong signal among the background noise from all the other celestial objects.

Our gathered data is available in a large file containing thousands of numbers representing signal magnitudes. In any existing file, write a function, `find_star(file_path)`, that searches through the given file for at least 5 signals in the range 4000–5000 inclusive. It should return the position of the fifth signal, or -1 if there were less than 5 in range.

Because the file is large and your coworkers' computers are old, you should make sure your function only checks as many signals as necessary. Once a neutron star is detected, it should stop iteration and return the result.

When you're done, upload a copy of your Python files to the OAKS dropbox.