Software Dev. & Problem Solving I GCIS-123

Due Date: see MyCourses Homework 1

Goals of the Assignment

This assignment provides an opportunity for you to practice Git, TDD, functions, conditionals and turtle.

Instructions:

- *You are required to use the following GitHub Classroom repository for this assignment:* [*https://classroom.github.com/a/LP5_mU2W*](https://classroom.github.com/a/LP5_mU2W)
  - *You are required to commit regularly (not only once at the end).*
  - *A header comment (author, title) per .py file and a doc string comment for each function are required.*
  - ***This is an individual assignment: please submit your own work.***

Assignment Problem:

A Health Monitoring System (**HMS**) is a sophisticated system consisting of a wearable wireless device (like a bracelet) with sensors that are synchronized with an application for a doctor to access a patient's medical information. The device measures pulse, oxygen levels, CO2 levels, temperature, blood pressure etc.

A. **Write python code for each of the following program functions**:

- ✓ ***get_pulse()***: this function gets the patient's pulse three times per hour for a total of 4 hours. Hint: you should call this function 4 times to represent the readings every hour.

- ✓ ***average_pulse()***: this function accepts three pulse values for each hour then calculates and returns their average per hour, but only if all pulse values are positive, else returns None.

- ✓ ***abnormal_pulse()***: this function accepts the four averages and returns the hour number (1 = first hour; 2 = second hour etc) in which the average is less than ***min_level*** or greater than ***max_level*** . In this assignment, you may assume that at most one hour is abnormal or return *None* if all levels are normal. A normal pulse is considered to be between 60 (min_level) and 90 (max_level), both inclusive.

- ✓ ***warning_doctor()***: this function prints a warning message to the screen containing the hour number of an abnormal level

- ✓ ***draw_chart()***: use turtle to visually show the pulse averages for all four hours. Hint: a bar chart might be a good approach.

- ✓ ***main()*** function.

B. Testing using pytest

Remember to use TDD:

a. Write tests
b. Stub out functions
c. Run tests (fail)
d. Implement the solution
e. Run tests (pass)

Following the TDD process, create the following test functions:

1. test_average_pulse(). The test should pass if all three pulse values and the average are positive, and fail otherwise.
2. test_abnormal(). The test should pass when all four values are within the interval [min_value, max_value], and fail otherwise.

Include at least 5 test functions in total.

Separating the tests from the production code will attract higher marks for this section.

Submission Instructions & Grading

Please submit the following two items:

- Commit all your source code to **Spring_Assignment1 GIT** repository ***before*** the deadline.
- Uploaded your source code file to **MyCourses** ***before*** the deadline

Exceptional Performance	**Competent Performance**	**Acceptable Performance**	**Developing Performance**	**Beginning Performance**	**Unacceptable Performance**
5 (100%)	**4 (95%)**	**3 (88%)**	**2 (75%)**	**1 (50%)**	**0 (0%)**
The solution is complete and correct. Instructions were followed completely. or The student made a small number of minor errors for the first time, e.g. wrongly named files, output that does not match specifications, etc. A warning should be issued regarding any errors.	The solution is functionally complete other than a small number of functional errors. Instructions were followed completely. or The solution is completely correct but the student made a small number of errors that they have made at least once before, e.g. wrongly named files, output that does not match specifications, etc. An explanation should be issued regarding any of these errors. Instructions were followed completely.	The solution includes at least 80% of the required functionality and all instructions have been followed. or The student's effort is obvious based on the volume of work and commit history, but there are a modest number of errors that range from relatively minor to significant that can be easily fixed by a member of the course staff with minimal effort.	The solution includes at least 50% of the required functionality and all instructions have been followed. The student's effort based on volume of work and commit history is obvious. There is at least some attempt to implement the missing functionality. Errors can be easily fixed by a member of the course staff in a short amount of time.	The student began the solution, and there is obvious effort, but more than 50% of the required functionality is missing or broken. This may include code that exists but does not compile and run, but could be fixed with significant effort.	Little or no effort. Almost all of the required functionality is missing. The commit history shows very little activity on behalf of the student.