|  |  |
| --- | --- |
| **NAME:** | **DUE:** |
| **Tomoki Koike** | **Tue. April 30th 2019** |

**Every Boiler Engineering Code – Entry Level Programming**

**Final Programming Exercises**

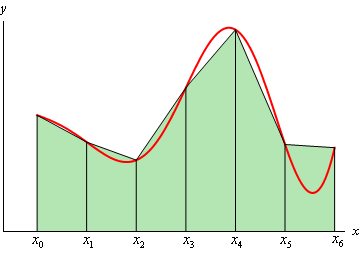
**(For this assignment, you should submit all your .py files and a word document including all the screenshots of the results.)**

|  |  |
| --- | --- |
| **Problem#** | **Points** |
| **#1** | **/25** |
| **#2** | **/30** |
| **#3** | **/30** |
| **Total** | **/85** |

**The python files, function/class files, and the text files were in different directories for my project. Thus, to run it on my Pycharm I used pathlib (Path). This path is unique to my files on my computer and will send errors if other people try to run my code on their computer. Please be aware of this and take it into consideration.**

1. **(25 points, Numerical Integration)** Numerical integration is the approximation of the area under a curve by using numerical techniques. In this lab, you will use the trapezoid rule to approximate definite integrals with the following equation:

A =

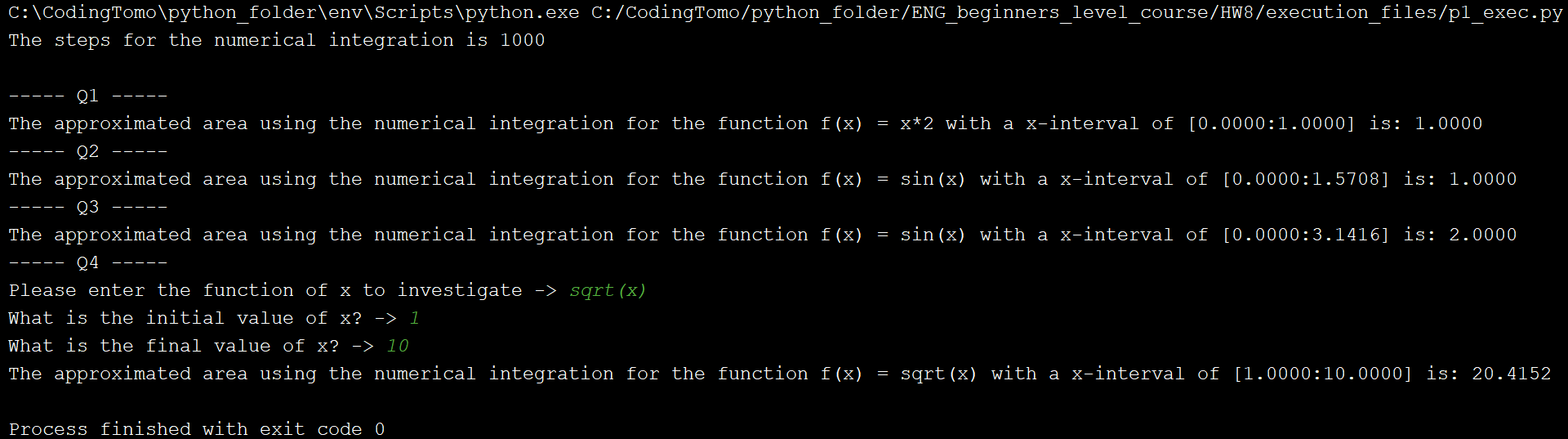


The first step to integrating a function numerically, that is, computing the area under the curve, is to discretize the x-axis between the limits of integration (min x-value and max x-value). To do this, decide how many trapezoids you want to divide the area into (for example, 1000, this number should be defined as a constant. The larger the number, the more accurate result of the area you will get). Next, derive a formula to compute the height of each trapezoid based on the number of trapezoids. Finally derive an expression to compute the bases of each rectangle. Based on the above, write a Python program that can integrate any function between specified limits of integration, and perform the integrations listed below.

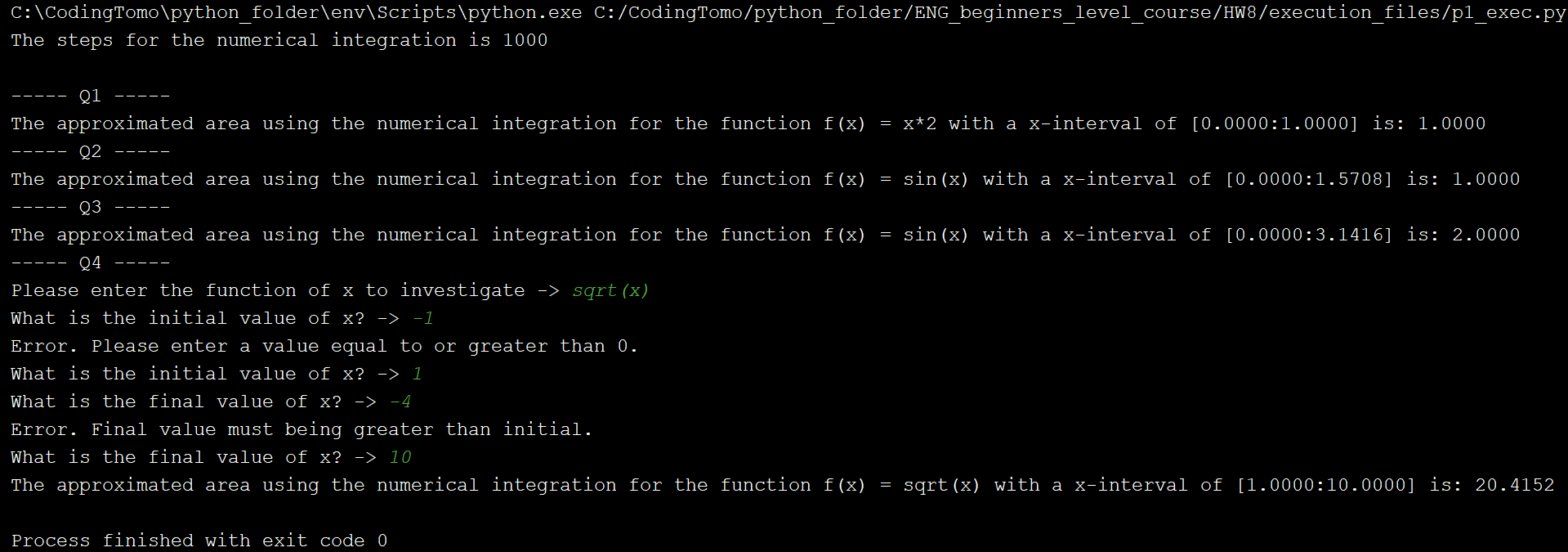
1. **(5 points)** Compute the area under y = x 2 between x = 0 and x = 1.
2. **(5 points)** Compute the area under y = sin(x) between x = 0 and x = π/2.
3. **(5 points)** Compute the area under y = sin(x) between x = 0 and x = π.
4. **(10 points)** Compute the area under y = between x = a and x = b where a and b are user specified values. Account for invalid user input cases of a < 0 and a > b. For each case of invalid input, immediately output to the user what the error was. Allow the user a total of three chances to enter valid input for each input request. If the user enters incorrect input three times in a row for an input request, display a different error and exit the program.

**SAMPLE EXECUTIONS:**

**>>VALID INPUT:**



**>>INVALID INPUT**



1. **(30 points, Gas Price)** In the attachment, there’s a text file named GasPrices.txt. The file contains the weekly average prices for a gallon of gas in the United States, beginning on April 5th, 1993, and ending on August 26th, 2013. Each line in the file contains the average price for a gallon of gas on a specific date. Each line is formatted in the following way:

**MM-DD-YYYY:price**

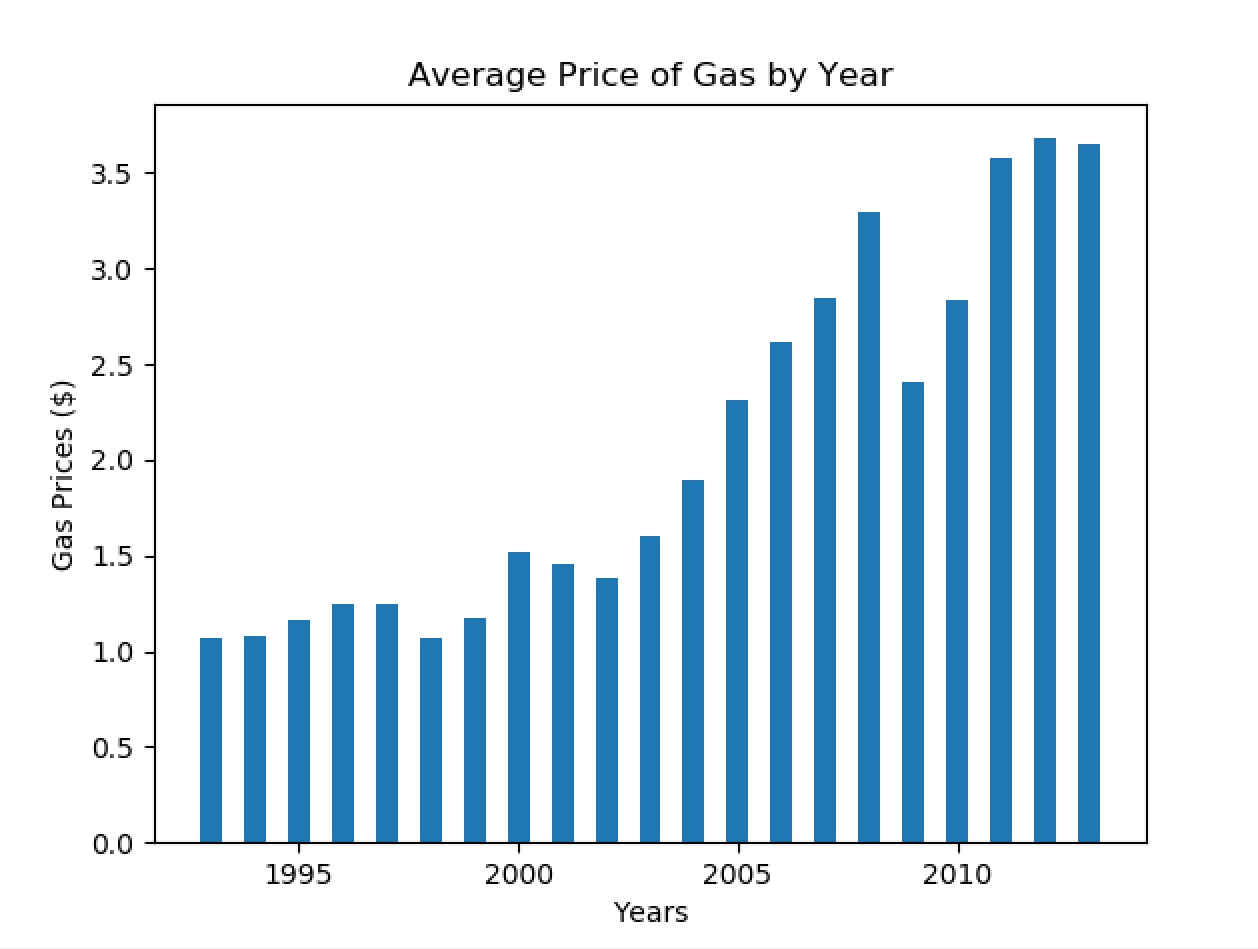
**MM** is the two-digit month, **DD** is the two-digit day, and **YYYY** is the four-digit yare. Price is the average price per gallon of gas on the specified date.

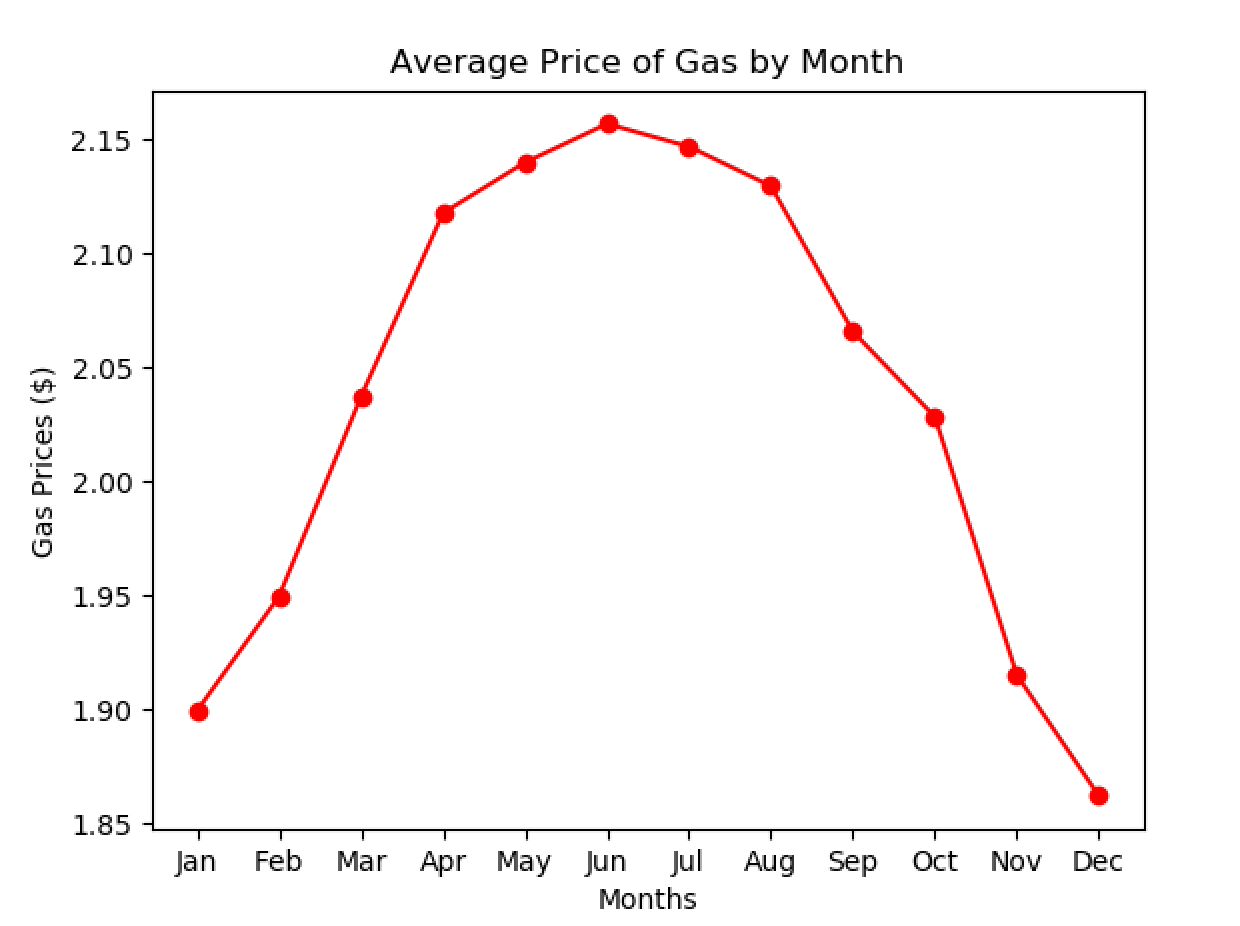
For this assignment, you are to write one or more programs that read the contents of the file and perform the following calculations:

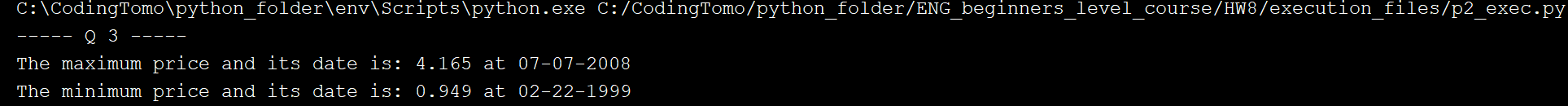
1. **Average Price Per Year (6 points)**: Calculate and display average price of gas per year, for each year in the file. (The file’s data starts in April of 1993, and it ends in August 2013. Use the data that is present for the years 1993 and 2013). Then use *matplotlib* to plot the average price of gas per year as a bar chart. Be sure to display meaningful labels along the X and Y axes, as well as the tick marks.
2. **Average Price Per Month (6 points)**: Calculate and display the average price for each month in the file. Then use matplotlib to plot the average price per month to a line graph. Be sure to display meaningful labels along the X and Y axes, as well as the tick marks.
3. **Highest and lowest Prices Per Year (6 points)**: For each year in the file, determine the date and amount for the lowest price, and the highest price.
4. **List of Prices, Lowest to Highest (6 points)**: Generate a text file that lists the dates and prices, sorted from the lowest price to the highest.
5. **List of Prices, Highest to Lowest (6 points)**: Generate a text file that lists the dates and prices, sorted from the highest price to the lowest.

You can write one .py file to perform all these calculations, or you can write different programs, one for each calculation.

**EXECUTION:**







1. **(30 points, Class Inheritance) Employee, ProductionWorker and ShiftSupervisor Classes**

Write an **Employee** class that keeps data attributes for the following pieces of information:

* Employee name
* Employee number

Next, write a class named **ProductionWorker** that is a subclass of the **Employee** class. The **ProductionWorker** class should keep data attributes for the following information:

* Shift number (an integer, such as 1, 2 or 3)
* Hourly pay rate

The workday is divided into two shifts: day and night. The shift attribute will hold an integer value representing the shift that the employee works. The day shift is shift 1 and the night shift is shift 2.

Next, write a class named **ShiftSupervisor Class** that is also a subclass of the **Employee** class.In a particular factory, a shift supervisor is a salaried employee who supervises a shift. In addition to a salary, the shift supervisor earns a yearly bonus when his or her shift meets production goals. The **ShiftSupervisor** class should keep a data attribute for the annual salary, and a data attribute for the annual production bonus that a shift supervisor has earned.

Write the appropriate accessor and mutator method for each class.

Once you have written all the classes, write a separate program that creates objects of **ProductionWorker** class and **ShiftSupervisor** class, and prompts the user to enter data for each of the object’s data attributes (you need import the module which includes the definitions of the classes in this program). Store the data in the objects, then use the objects’ accessor methods to retrieve them and display them on the screen.

**EXECUTION:**

