DAEN 500-1 – Data Analytics Fundamentals

Spring 2021 Final Examination Exercise

3/5 – 3/6/2021

Final Submission Deadline: NLT 11:59PM (EST). Saturday, March 6, 2010

*Failure to submit ON TIME will result in DAEN COURSE FAILURE*

Name:\_\_\_Kyle Murphy\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ GMU G#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Student Signature (Honor Certification): \_\_\_\_Kyle Murphy\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

This exam is **OPEN BOOK/OPEN NOTES**. You may consult any of the course texts, and the various reference materials recommended in the syllabus. ***The exam of course IS NOT “Open Web”,*** especially in that you may NOT utilize expert “help” sites such as Stack Overflow, or other programming help or collaboration sites.

HONOR CODE CERTIFICATION

**Your signature above declares that you have followed the conditions of this exam, and that the work is yours alone**. **Specifically:**

This must be your own work, authored and completed by you. As stated earlier, this is an “open source exam” – allowing books, notes or courseware, as well as *general* expert advice gained PRIOR to exam. YOU MAY NOT, HOWEVER, SEED OR USE ANY ADVICE ON HOW TO SOLVE THE QUESTION OR ANY CODE WRITTEN BY ANY OTHER INDIVIDUAL. *Any violation will result in an immediate failure in the exam and for the course, as well as referral to the GMU Honor Committee for determination of any other appropriate disciplinary consequences.*

*NOTE: Your* ***submission*** *of any responses, files, programs, etc. in response to the DAEN500 final exam instructions, will also be your personal certification of your full compliance with the spirit and letter of the* ***GMU Honor Code*** *standards for take home and/or in-class exams.*

Additionally, you are restricted from discussing the substance of the questions on this exam with any other individual, until after you have submitted your final response for grading. The completed exam -- with your answers embedded in this docx document (add extra pages as necessary) should be submitted following instructions contained in the Final Exam Instructions BB site. If you have any trouble submitting and have extra parts of the answers you have trouble appending to this document, you may simply submit additional pages separately (the exam submission site is set for multiple submissions, just in case). Make certain all are submitted PRIOR TO THE DEADLINE!

 FINAL EXAM PROBLEMS

COMPLETE ALL & INSERT ANSWERS BELOW QUESTIONS

# Problem 1: Python Programming Problem (15 Points Total)

* **Design and implement a Python program that is based on the following requirements: a) program will find all numbers which are divisible by 7 but are not a multiple of 5; and b) numbers between 2000 and 3200.**
* **INSERT (cut&paste) your Python code in space below and *then insert a screen shot in space below, showing code, your successful run, input and output.***

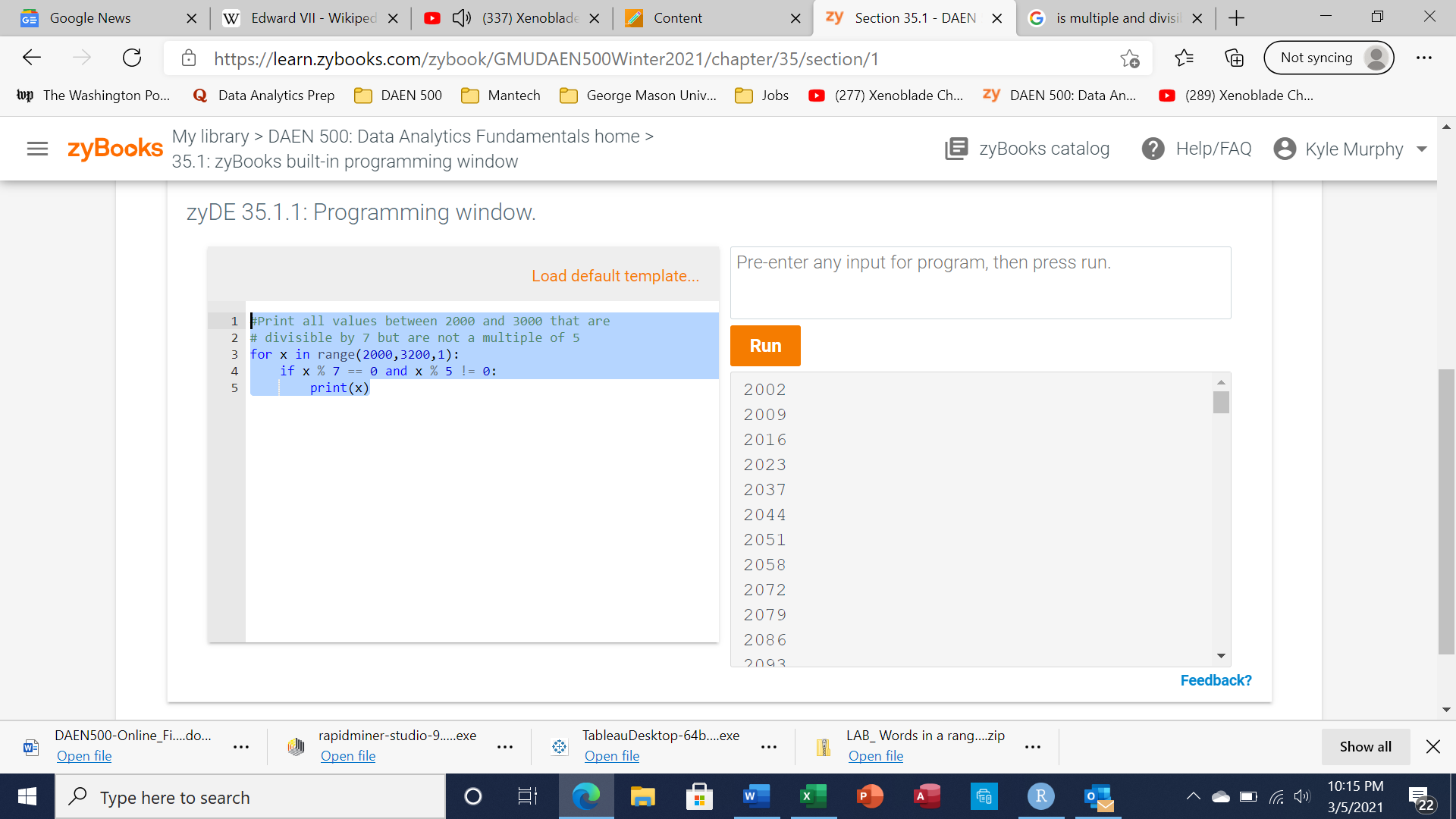
***#Print all values between 2000 and 3000 that are***

***# divisible by 7 but are not a multiple of 5***

**for x in range(2000,3200,1):**

**if x % 7 == 0 and x % 5 != 0:**

**print(x)**



NOTE of alternative for help: To help test your code, you also may use a Python “programming window” found in the **Zybooks Section 35 Additional Material** OR any relevant ID

# Problem 2: Python Programming Problem

# (15 Points Total)

* **Design and implement a Python program that is based on the following requirements:**

**a) define a class which has *at least two* methods**

* + **Method 1 – getString: to get a string from console input; and,**
  + **Method 2 - printString: to print the string in upper case.**

**b) demonstrate code works using three different test input strings**

* ***INSERT* *code below* and *INSERT* a screen shot of the program and successfully run output that *includes test input for input strings (test strings must include (a) all upper case, (b) all lower case, and (c) mix of upper and lower case).***

**# Establishes class with constructor**

**# and the two methods getString and printString**

**class Login:**

**def \_\_init\_\_(self):**

**self.userName = ""**

**def getString(self):**

**self.userName = str(input())**

**def printString(self):**

**print("Pet Care Website")**

**print("User Login:", self.userName.upper())**

**# Main section of the code that creates the instance**

**# of Login class as my\_login and calls getString**

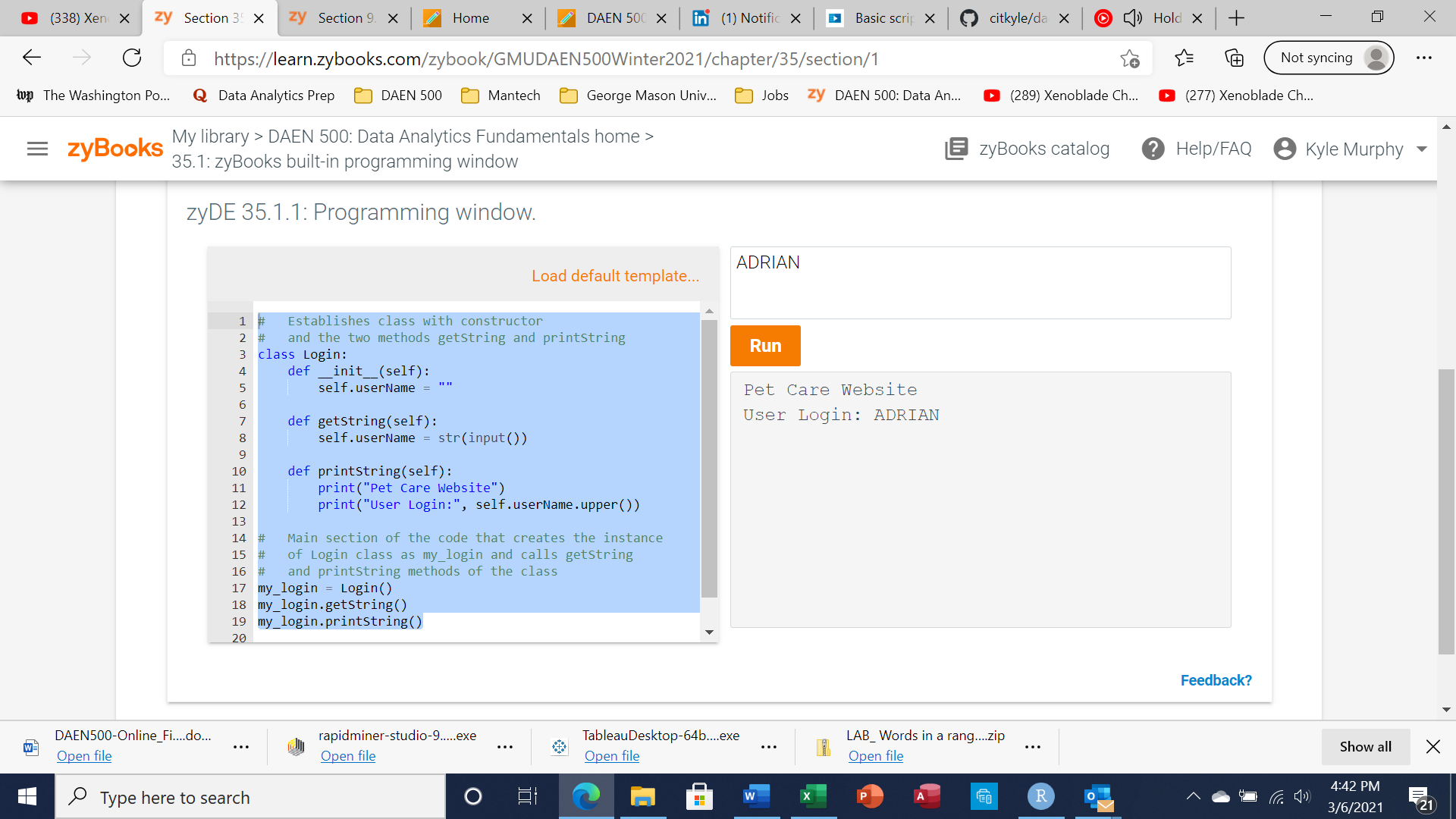
**# and printString methods of the class**

**my\_login = Login()**

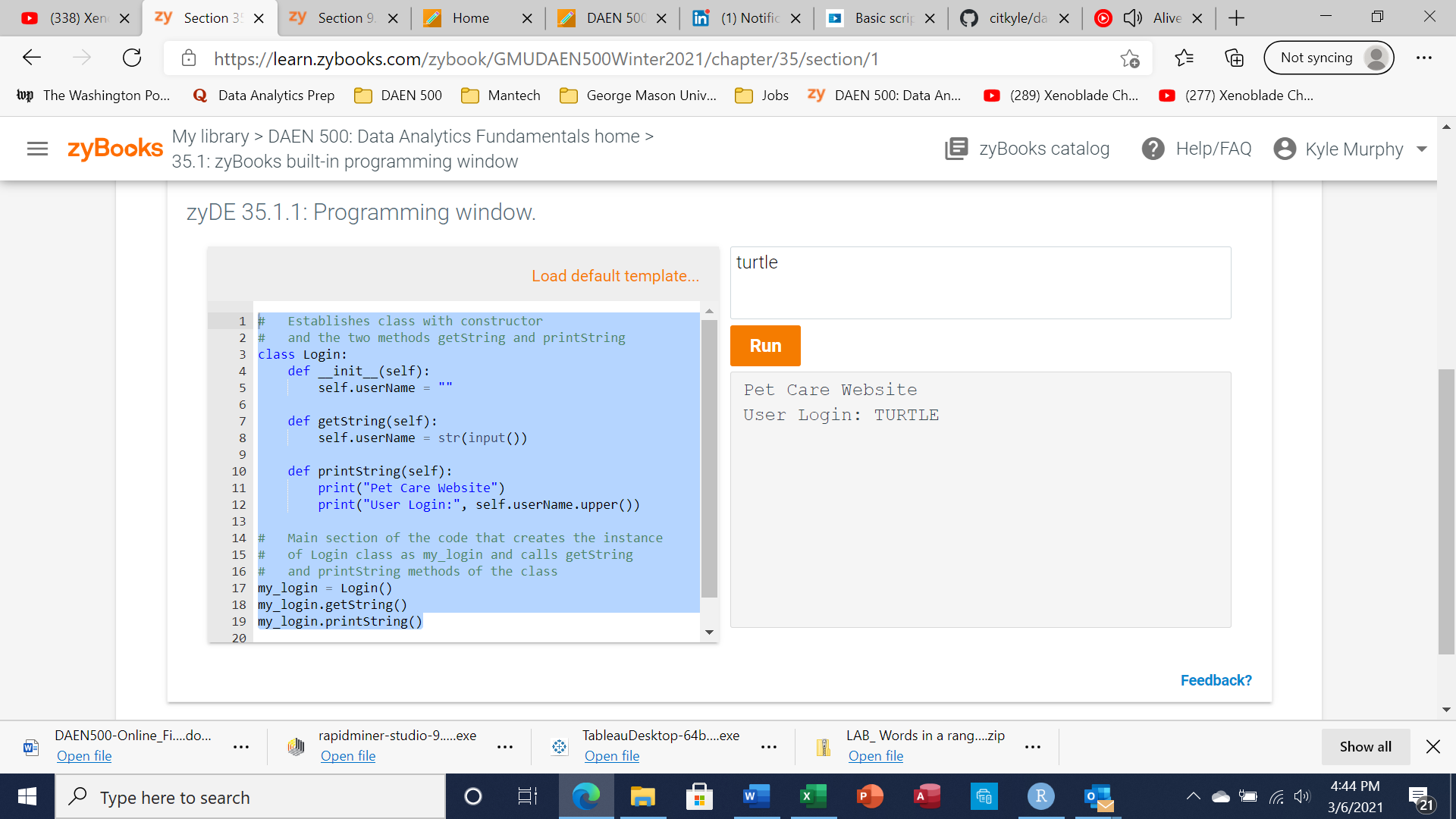
**my\_login.getString()**

**my\_login.printString()**

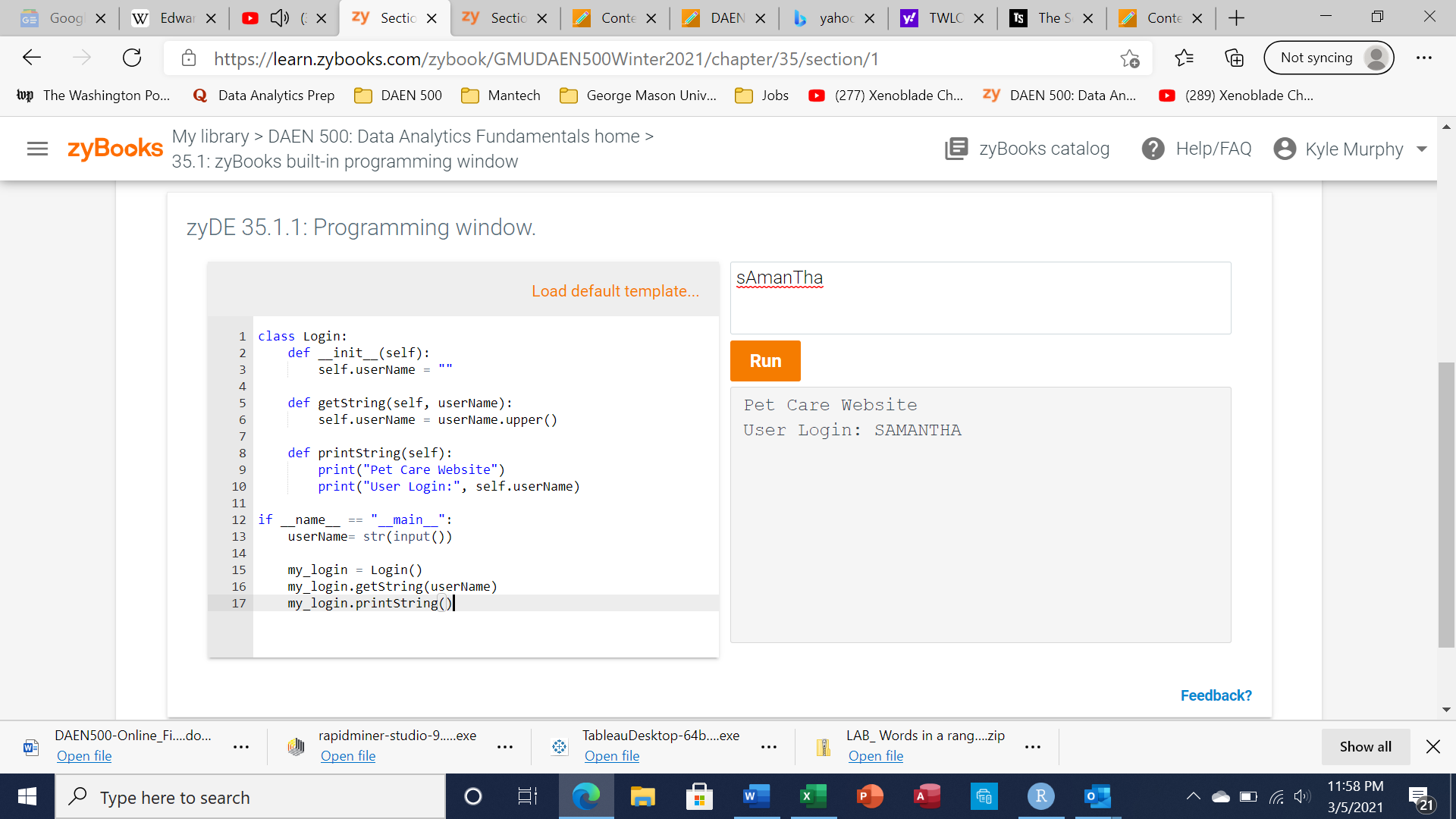
1. **Upper Case Input: “ADRIAN”**



**b) Lower Case Input: “turtle”**



1. **Mixed Upper and Lower Case: “sAmanTha”**





# Problem 3: R Programming Problem

# (20 Points Total)

* **Perform the following problems using R:**
  + Create a vector of courses (e.g., MATH 101) you have taken previously. Make sure you have at least 8 courses. Name the vector myCourses
  + Get the length of the vector myCourses
  + Get the first two courses from myCourses
  + Get the 3rd and 4th courses from myCourses
  + Sort myCourses using a method
  + Sort myCourse in the reverse direction
* *INSERT* *code below* and *INSERT* a screen shot of the program and successfully run output.

**#Creates the vector myCourses with the courses specified**

**myCourses <- c('DAEN500','CS504','AIT580','STAT515','OR531','AIT524','AIT582',**

**'AIT614','AIT620','AIT622','AIT624','AIT664','AIT726','AIT736')**

**#Pulls the length of the vector myCourses**

**length(myCourses)**

**#Pulls the first two courses of the vector myCourses**

**myCourses[1:2]**

**#Pulls the first 3rd and 4th courses of the vector myCourses**

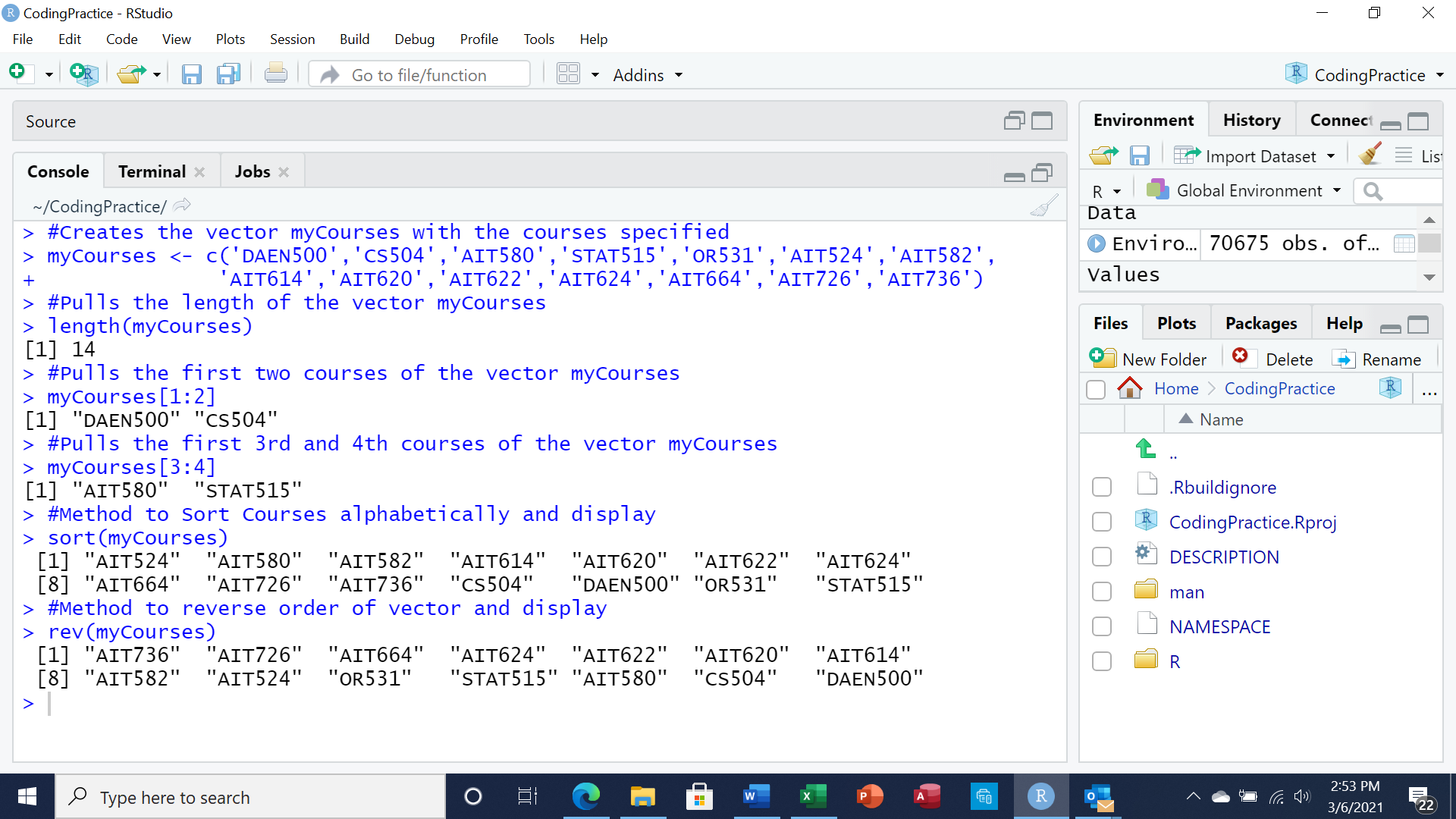
**myCourses[3:4]**

**#Method to Sort Courses alphabetically and display**

**sort(myCourses)**

**#Method to reverse order of vector and display**

**rev(myCourses)**





# Problem 4: Principal Component Analysis

# (25 points)

**Provide a description of the following:**

1. What is a component – Provide a description (5 points)

A component is a new variable created from a weighted combination of predictor variables. The weights are determined by all the correlations of the predictor variables.

1. Principal Component Analysis – Provide a description.(5 points)

Principal component analysis is the transformation of variables into fewer components to make analysis easier. The first principal component will show data with the greatest variability along the x axis while the second principal component will be shown along the y axis.

1. **Provide an specific example of Principal Component Analysis(15 points)**

An example of principle component analysis is 5 hockey player statistics variables measured: Maximum Speed, plus/minus rating, pass percentage, minutes played per game, and assists.

Two principal components may arise from this data correlating with principle component 1 being created from a correlation between the variables plus/minus, assists, and minutes played per game and principle component 2 created from a correlation between the variable speed and pass percentage. Principle component 1 had more data variability so it is specified as the x axis and principle component 2 is specified as the new y axis.

# Problem 5: Multiple vs. Logistic

# (30 points)

# Describe: What is difference between Multiple Regression and Logistic Regression? What circumstances might determine which to use? (10 points)

# Demonstrate: Using any data, and any tool set you’ve learned about, show differences (20 points)

# SUGGESTION: may be solved using RapidMiner, or other toolsets, BOTH TO ANALYZE AND TO VISUALIZE REGRESSION DIFFERENCES.

Step 1: Perform a quick search of the [UCIS public data archive](https://archive.ics.uci.edu/), a well-curated site which you already have seen as part of your introductory RapidMiner training.

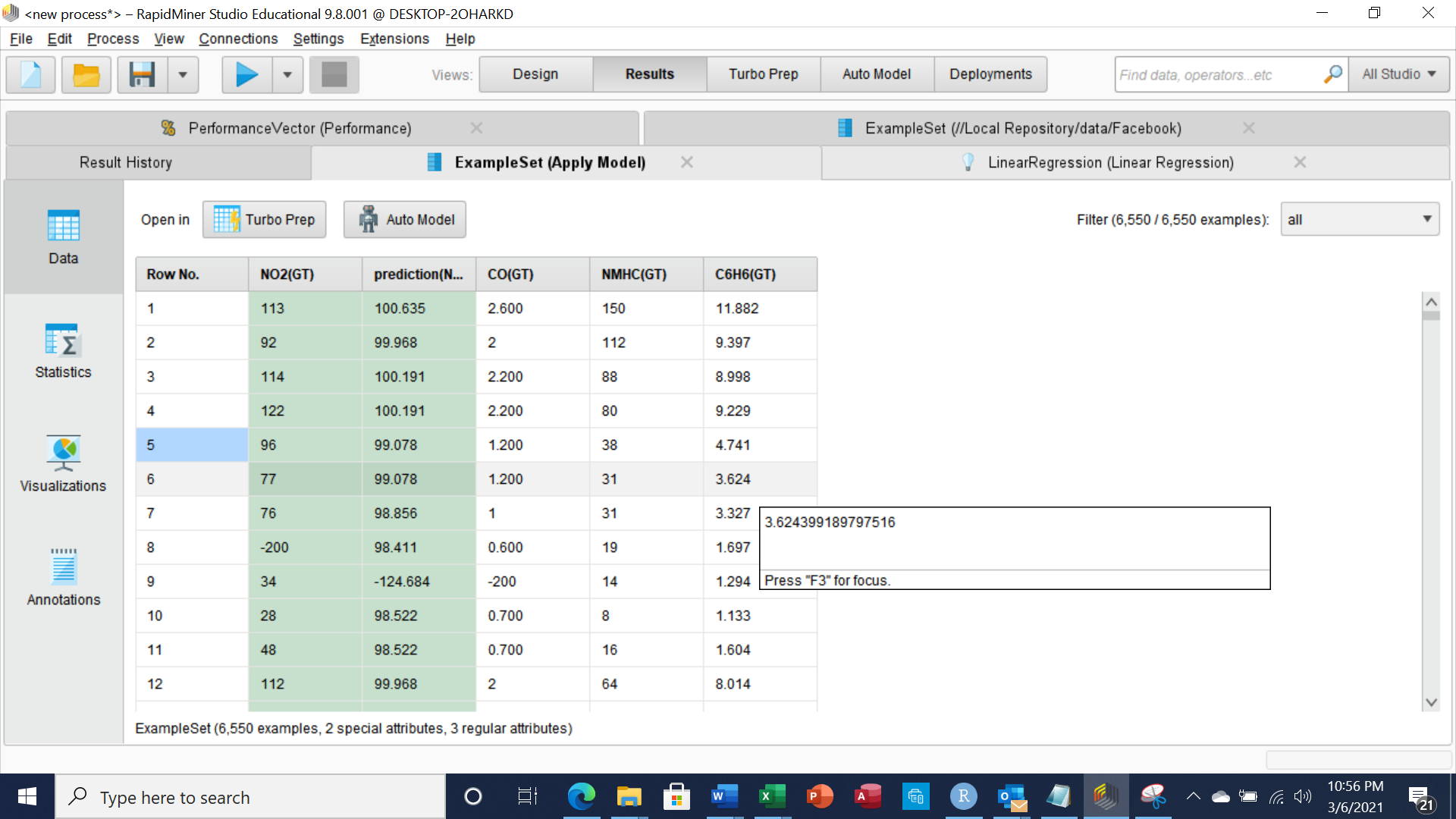
Step 2: Pick a dataset you find interesting, input dataset into regression tools you’ve chosen.

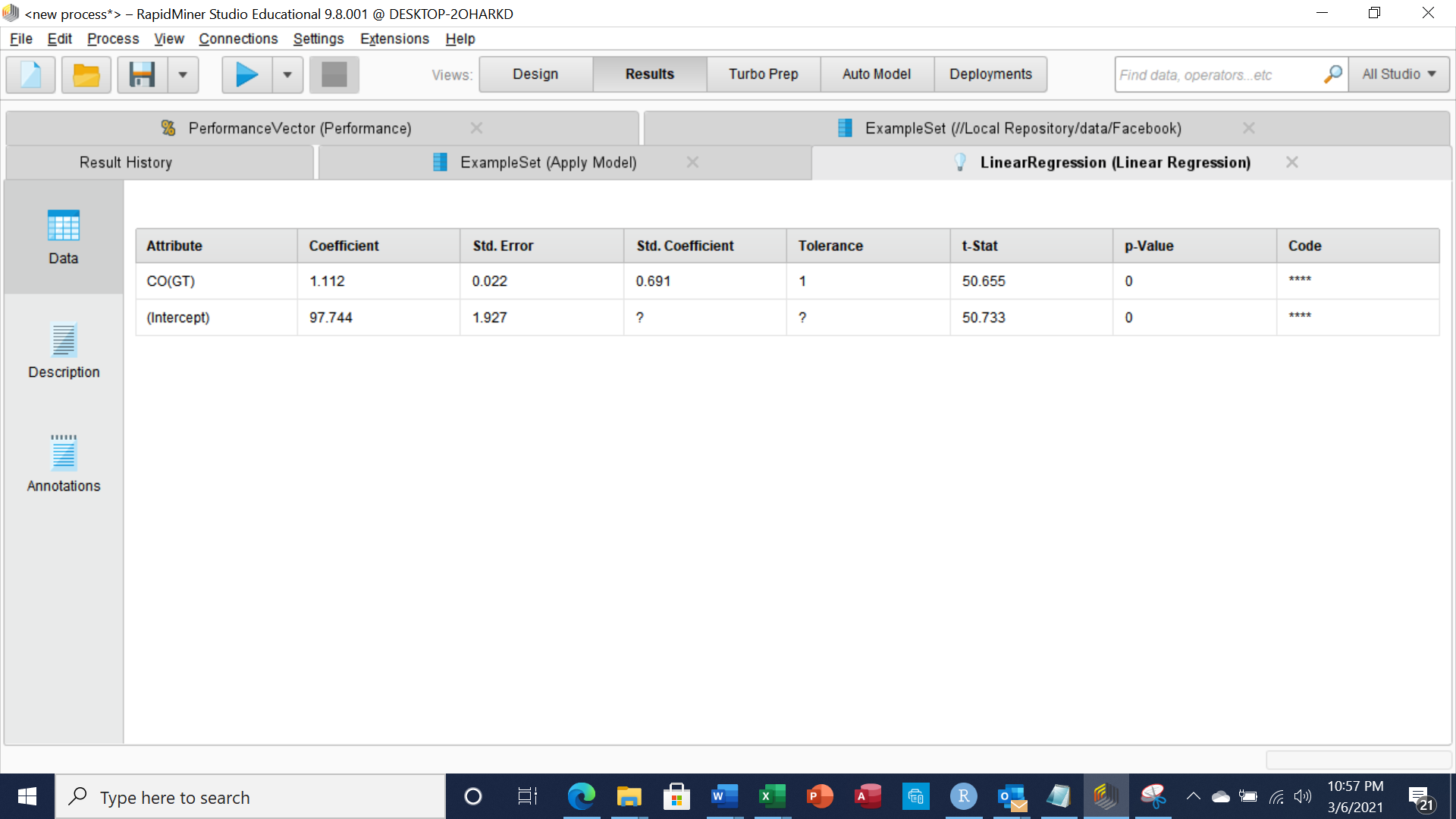
Step 3: Run regression, .and use visualizations to demonstrate the conceptual answers you provided for 5.(a).

**Multiple Linear Regression** models the linear relationship between one quantitative response variable and one or more predictor variables. The response variable is the variable being modeled or predicted, while the predictor variables are the variables used to predict the response variable. For example, the predictor variables calories eaten per day, steps per day, and minutes exercising a week and the response variable is Body Mass Index. Predictor variables are represented by X and response variables are represented by Y.

Multiple Linear Regression may result in poor results such as in the case of binary response variables where the outcome can only result in two different results. If linear regression is used with binary response variables, if 0 is no and 1 is yes the result may be over 1. When binary response variables are used, **logistic regression** tends to be more commonly used.

I performed the following logistic regression in RapidMiner using a Air Quality sample dataset returning the following results using the predictor variables CO, NMHC, and C6H6 to predict the response variable NO2. My results are below. CO had a positive coefficient of 1.112. The performance had a root mean squared error of 94.179.





For the logistic Regression Model, I had issues getting my data set to run in rapid miner.

When attempting to use the deals sample dataset I kept getting the following error regarding the a numerical label not being supported in the regression model section.