DAEN 500- DL2 – Data Analytics Fundamentals

Fall 2020 Final Examination Exercise Package

10/7 – 10/13/2020

Final Submission Deadline: NLT 11:59PM (EST). Tuesday, 13 Oct 2020

*Failure to submit will result in DAEN COURSE FAIL/NoCredit*

Name: Patrick J. Doumont GMU G# G01291522

Student Signature (Honor Certification): Signature provided through electronic submission

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 -- within a specified range -- which are divisible by 7 but are not a multiple of 5*; and b) demonstrate the program works by running the program for the range: *numbers between 2000 and 3200*.**
* **INSERT (cut&paste) your Python code in space below and *then insert a screen shot in space below, showing your successful run and output.***

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**.

input1 = int(input("Enter the lower boundary for the range: "))

input2 = int(input("Enter the upper boundary for the range: "))

input\_range = list(range(input1,input2+1))

divis\_by\_7\_list = []

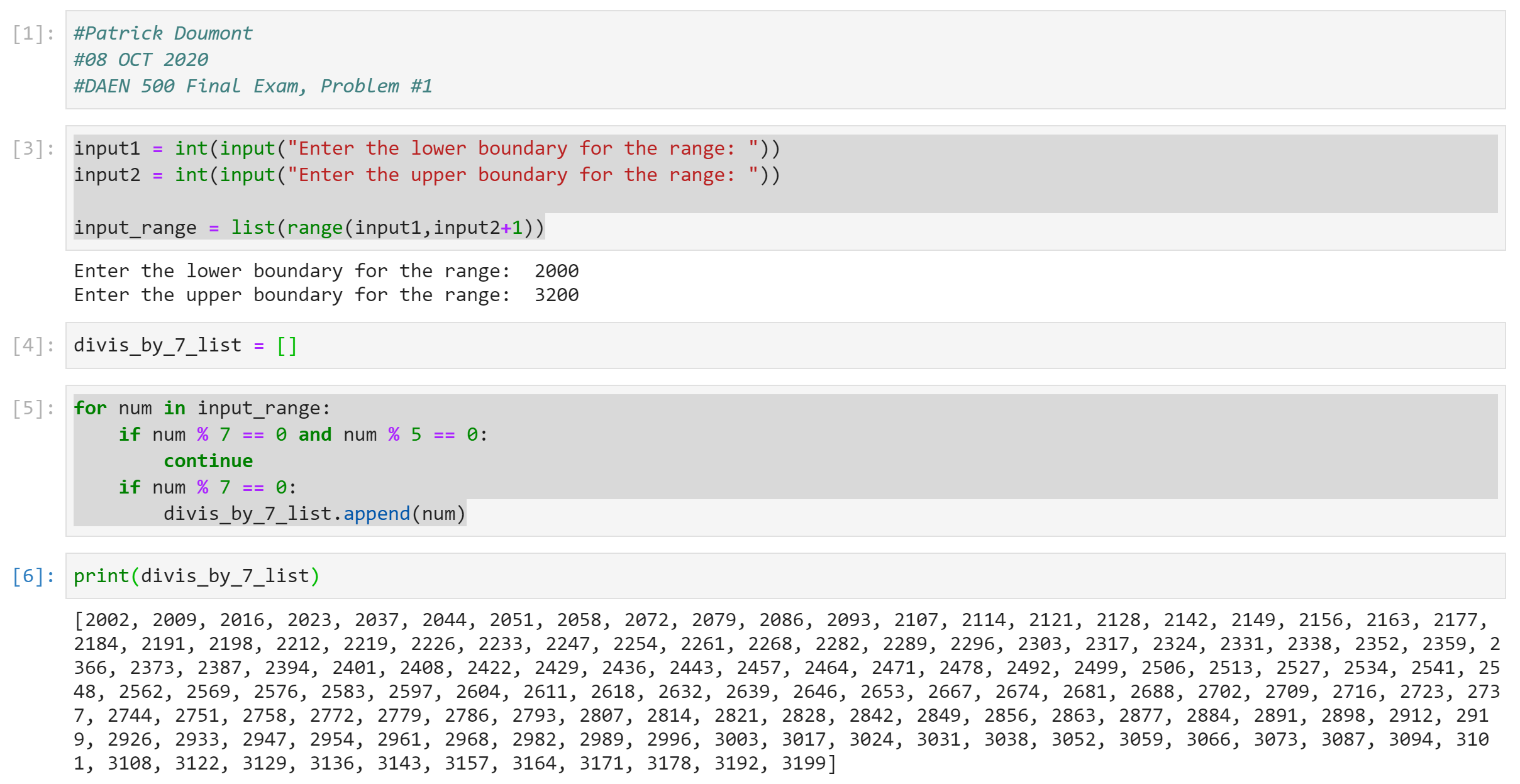
for num in input\_range:

if num % 7 == 0 and num % 5 == 0:

continue

if num % 7 == 0:

divis\_by\_7\_list.append(num)

print(divis\_by\_7\_list)

This program was designed and implemented in JupyterLab, a part of the Anaconda package.

# 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).***

class String\_Manipulation:

"""Prompts the user for a string and returns the string in upper case."""

def \_\_init\_\_(self, user\_input=''):

self.user\_input = user\_input

def getString(self):

self.user\_input = input("Enter a string to be returned in upper case: ")

def printString(self):

print(self.user\_input.upper())

new\_input = String\_Manipulation()

new\_input.getString()

new\_input.printString()

new\_input2 = String\_Manipulation()

new\_input2.getString()

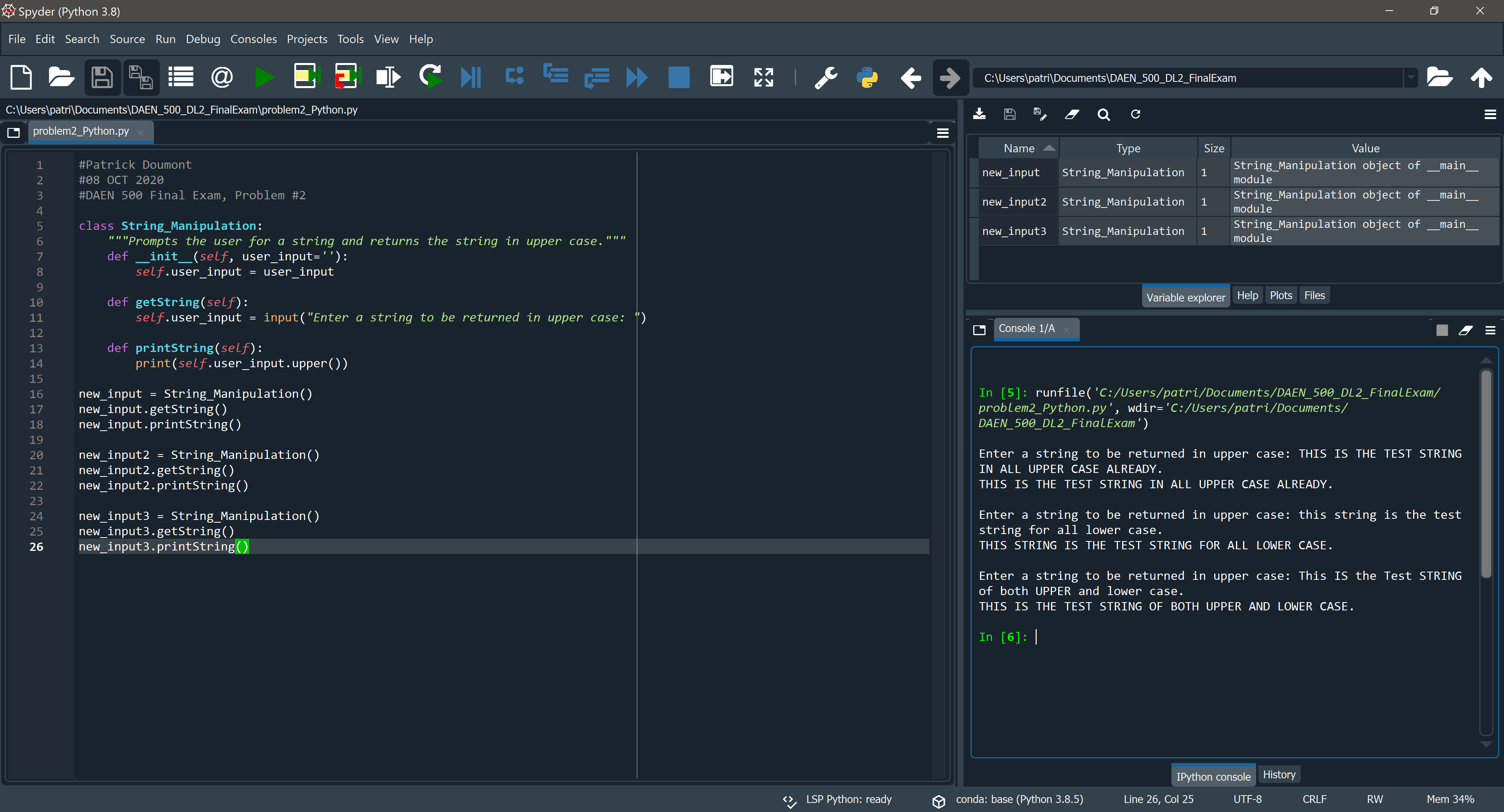
new\_input2.printString()

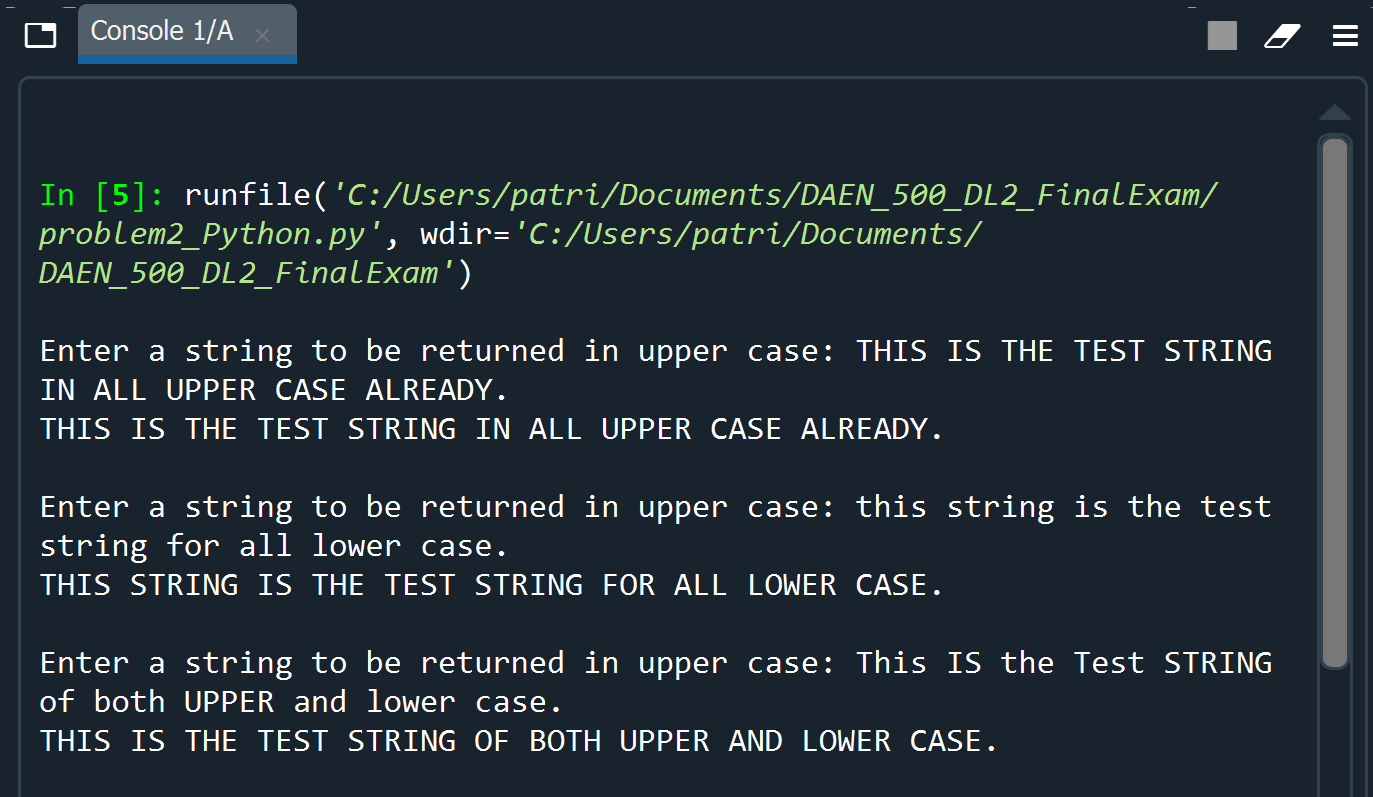
new\_input3 = String\_Manipulation()

new\_input3.getString()

new\_input3.printString()

The successful outputs to (a), (b), and (c) can be seen in the console on the bottom right corner. I have provided a “blown-up” version of just the bottom right below this main screen since the text is quite small in the picture below which includes the script, the variable explorer window, and terminal.





This script was created using Spyder, an integrated development environment, a part of the Anaconda package.



# 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.

#Create vector of previous courses

myCourses <- c("CH101", "EN101", "HI103", "MA103", "MA104",

"MA205", "PH201", "PH202")

#Find vector's length

length(x=myCourses)

#Return first two courses from myCourses

myCourses[c(1,2)]

#Return third and fourth courses from myCourses

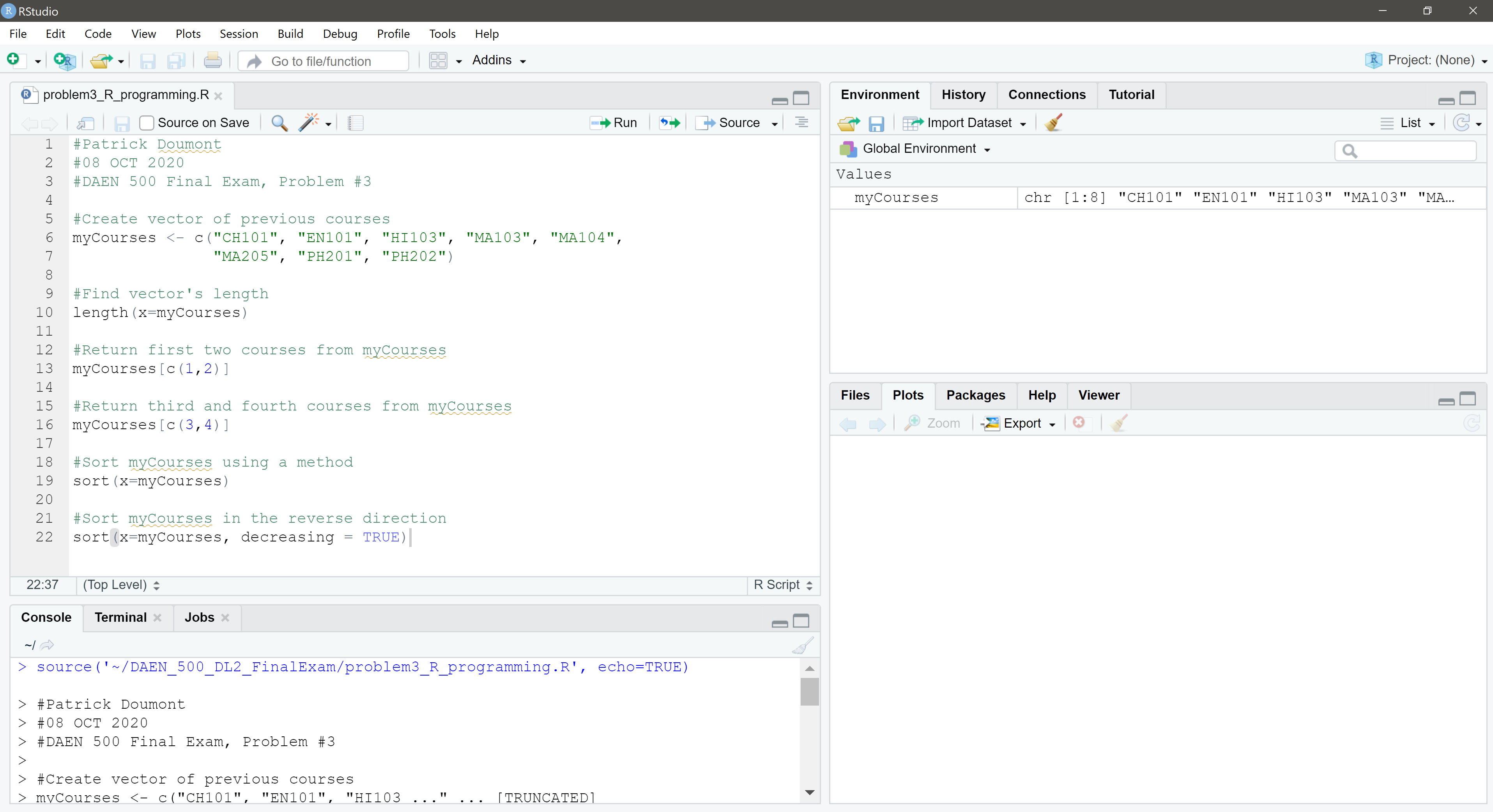
myCourses[c(3,4)]

#Sort myCourses using a method

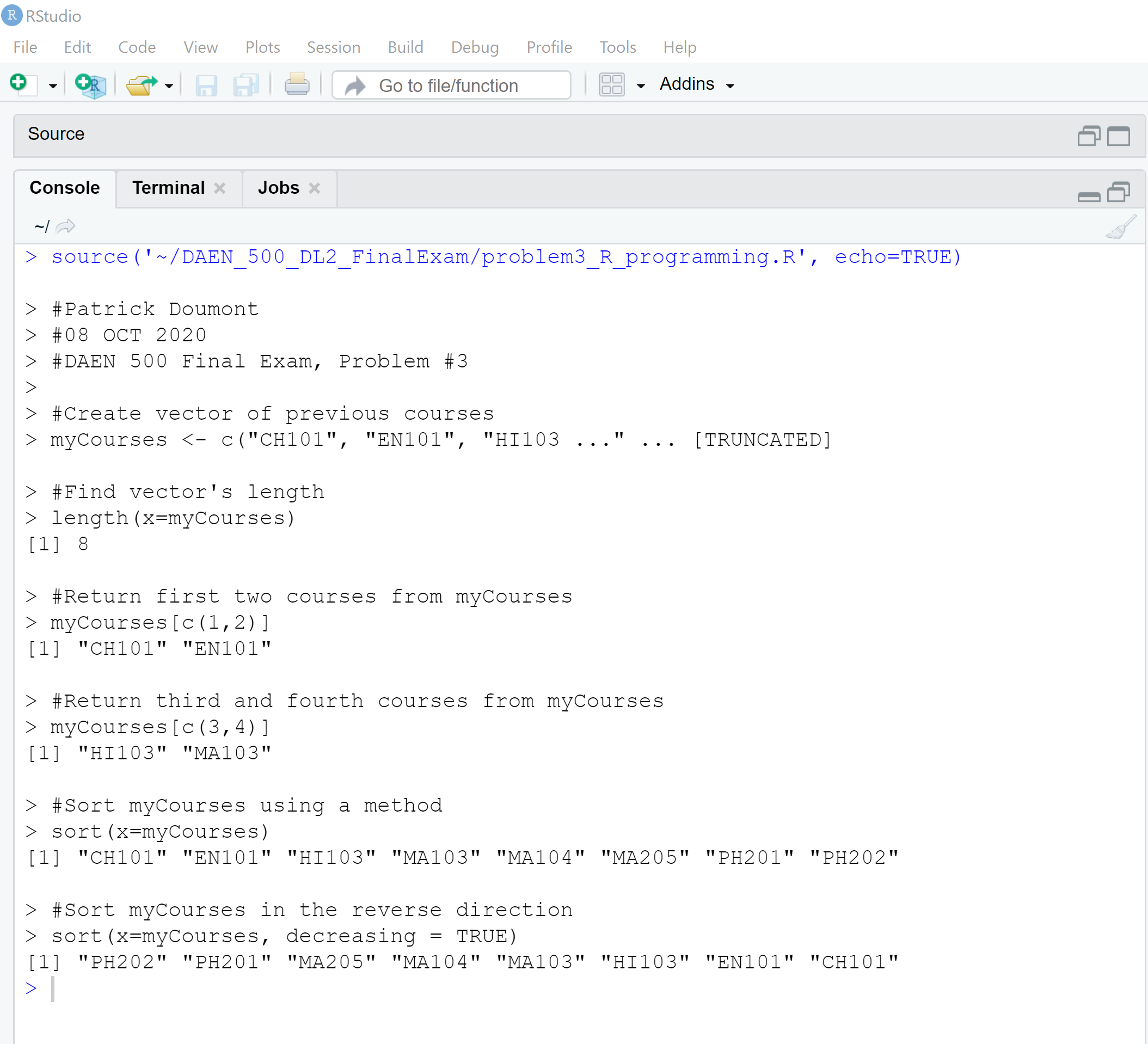
sort(x=myCourses)

#Sort myCourses in the reverse direction

sort(x=myCourses, decreasing = TRUE)



Here is a screenshot of the script in RStudio.



This a screenshot of the script running in the “console” pane of Rstudio with appropriate outcomes.



# 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 variable created which is the combination of many predictor variables. It can help determine what best explains the variability in data. It can also serve to reduce the number of parameters in your analysis. Though correlations are used to help calculate components, they are not just simply the grouping of correlated variables or data. They are the weighted combinations of the predictor variables. By convention, they are denoted as Z1, Z2, etc.
2. **Principal Component Analysis – Provide a description.(5 points)**
   * Principal component analysis seeks to transform variables into fewer possible components to simply analyses if possible. The end goal is to maximize the variable in the data along as few components as possible. This is achieved by “rotating” the data where the new axes are in the direction of the principal components. The first principal component, the one with the greatest variability will become the new x-axis.
3. **Provide and explain a specific example of a Principal Component Analysis(15 points)**
   * An example that could use Principal Component Analysis would be for determining the price of a car given multiple variables such as color, age, miles, engine size, automatic/manual, seating capacity, 2WD/4WD, etc. Hypothetically, in the conduct of the analysis you notice that the car’s age and the miles driven on the car are highly correlated. This makes sense since it is well known cars depreciate quickly soon after they are purchased. You therefore decide to create components where principal component 1 may be a weighted combination of those two variables (age and miles) that explain the majority of the change in price over time. Though maybe not as predictably as age and mileage, engine size and seating capacity could serve as a second principal component as their relationship may be highly correlate in the mainstream car market (not necessarily luxury or supercars) due to a larger engine being needed for a car that can hold more people.

# 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)

# Multiple linear regression is used to predict a quantitative response variable based upon multiple predictor variables. An example would be the price of a car given its age, seating capacity, engine size, automatic/manual, 2WD/4WD, etc. It is used to predict a number which represents the model’s prediction of price given the predictor variables. The price of the car could be $1500 if it has 250,000 miles, seats two, manual, 2WD, and 20 years old. Or if a car is new, 15 miles, seats 7, is automatic, and 4WD, the car may be $45,000. The response variable could also be any number in between (or outside of that range).

# Logistic regression is used to determine the probability of an outcome which only has two options, often called a binary response. In this model, your predictor variables help determine a probability or likelihood of being in one of two bins. An example would be (given a 2-party system) building a model whose predictor variables consist of demographic information like sex, age, location of residence (city/suburbs/rural), work-type, etc that provides which party, Democrat or Republican you are more likely to be. For example, a 65 year-old white male who lives in Birmingham, AL and works at a bank is more likely to be a Republican but a 23-year old black female who attends UC-Berkley as a student is more likely to be a Democrat. This is only a hypothetical to show how there are only two options Republican or Democrat.

# 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..

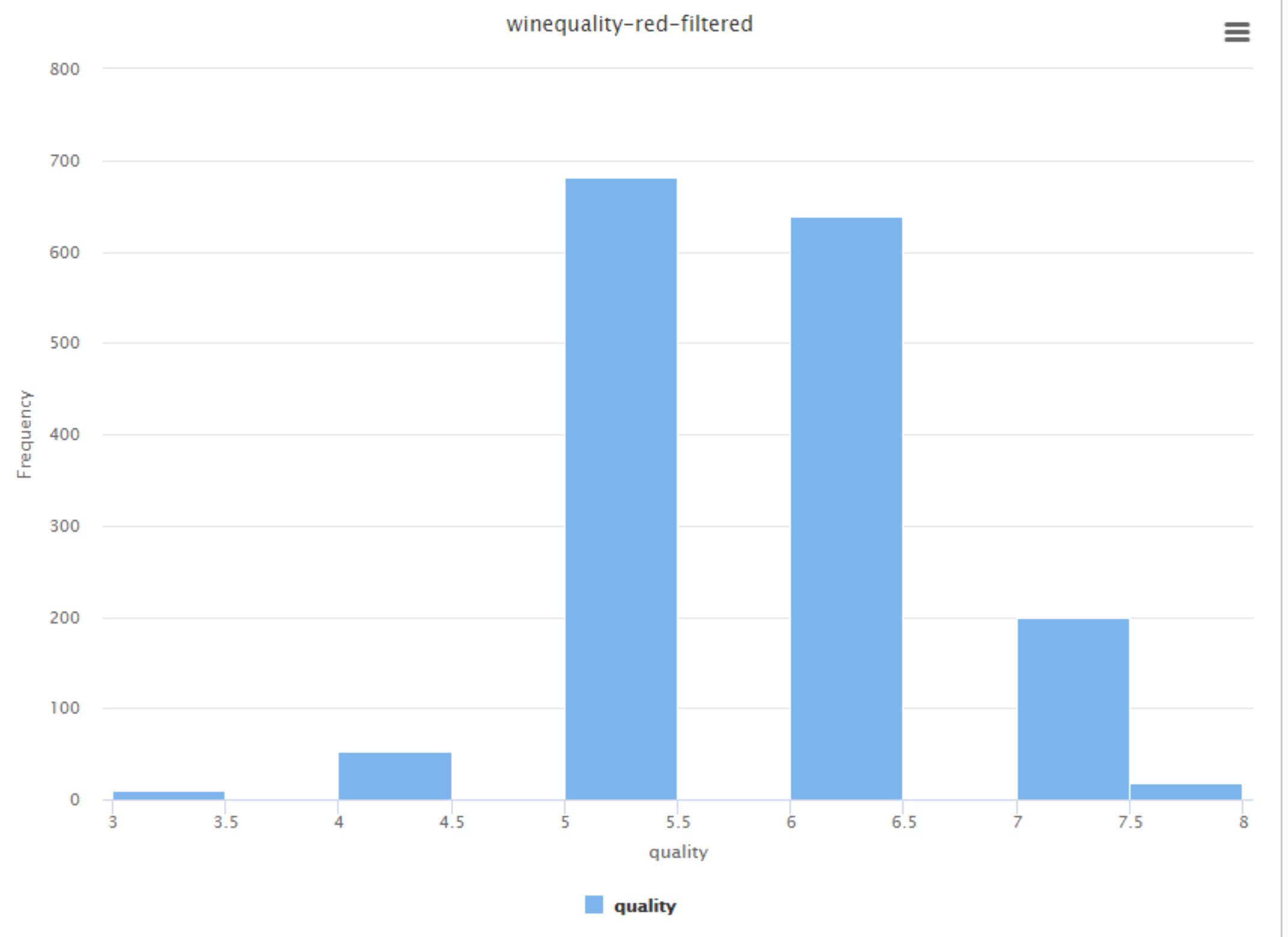
Step 1: Perform a quick search of the [UCIS public data archive](https://archive.ics.uci.edu/), a well-curated site which you already have seen as part of your introductory RapidMiner training.

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

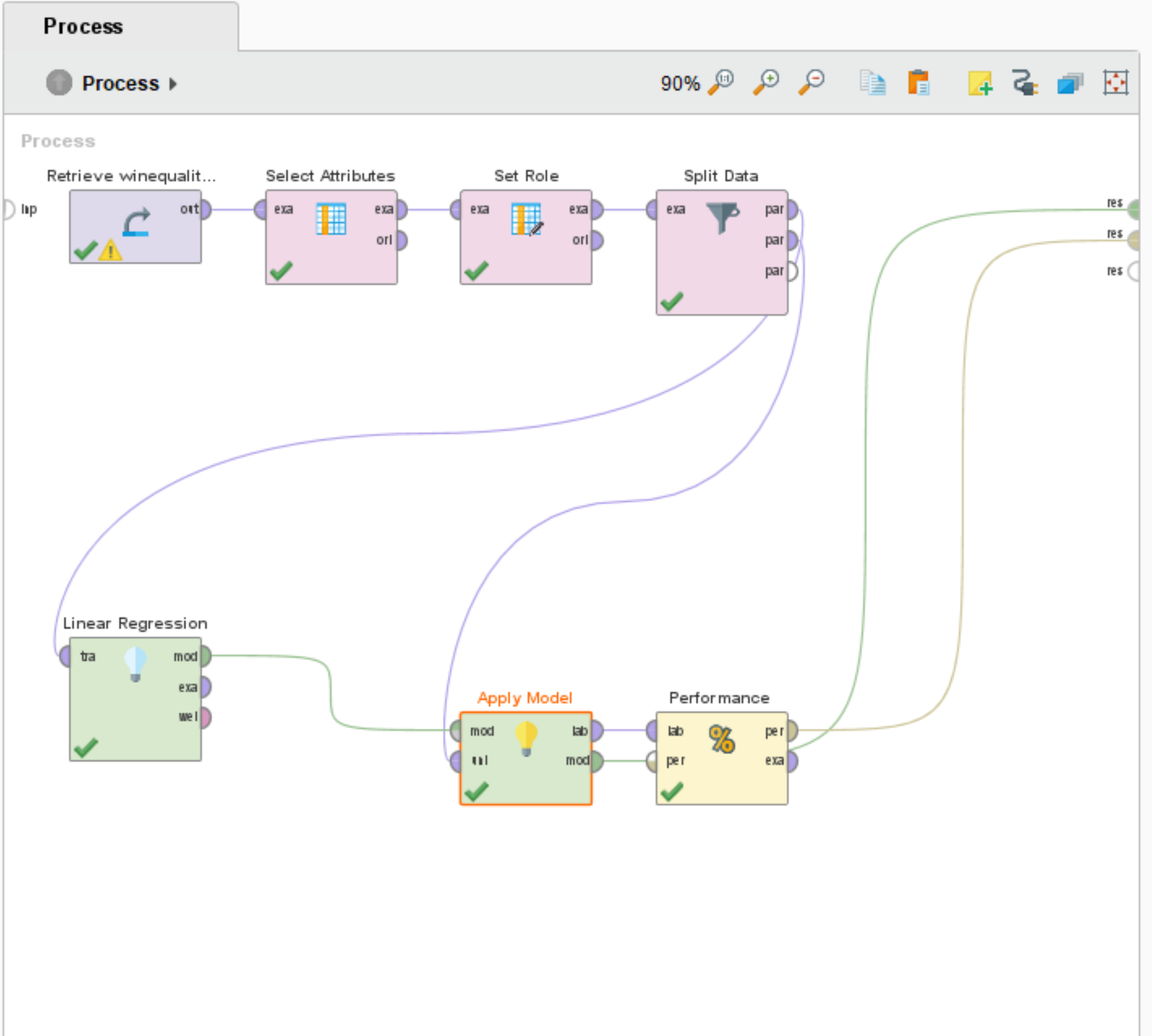
* I have chosen to use the Wine Quality Data Set - Red that can be found in the “Most Popular” section of the UCIS data archive. I chose this data set because it contained many quantitative fields in a format (CSV) that I have become familiar with over the course of the class. A link can be found here - <https://archive.ics.uci.edu/ml/datasets/Wine+Quality>
* Since the dataset consisted of only quantitative variable, and the outcome itself was a quantitative variable, I had to create a new variable which labeled a wine either good or bad based upon its rating being above a certain value (I chose greater than 6 for its review). I did this to only show the difference between what a multiple regression and logistic regression outcomes look like and needed a predicted variable whose outcome was one of only two categorical options (i.e. good or bad wine).
* I used RapidMiner as my tool of choice to conduct both regression analyses. I used the same variables in both regression choices after removing a couple whose p-values were considerably higher than 0.05 when calculated for the multiple regression. I decided to keep the predictor variables the same for the logistic regression so they are more easily comparable.
* The multiple regression uses as input multiple characteristics of the red wine such as ph, and alcohol content (as well as many more) to predict the quality of the wine on a 0-10 scale. The number can be any number in between.
* The logistic regression uses as input the same variables as the multiple regression but predicts instead whether the wine will be a “good” or “not good” wine.

Step 3: Run the dataset (*may be a significant subset, if the dataset is very large*) first. a Multiple Regression and then a Logistic Regression, .and use visualizations to demonstrate the conceptual answers you provided for 5.(a).

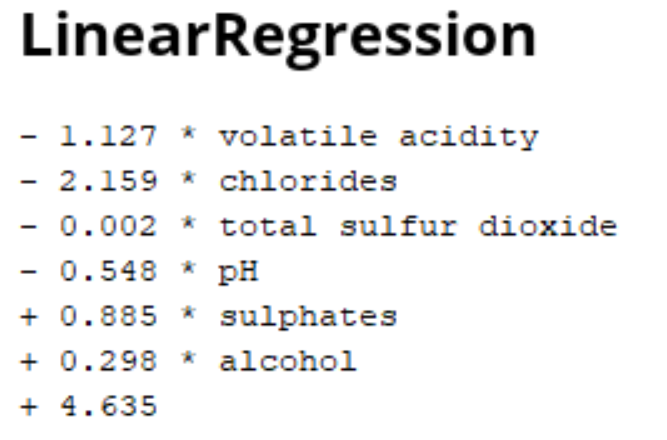
Below is a frequency chart of quality of wine for the entire data set. You can see how most wines (by far) were between 5 and 7.

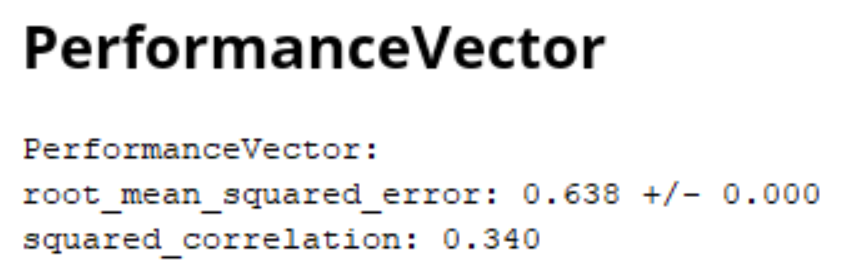


Below is the process I used for running the multiple regression to determine the quality of a red wine based on its physiochemical tests.



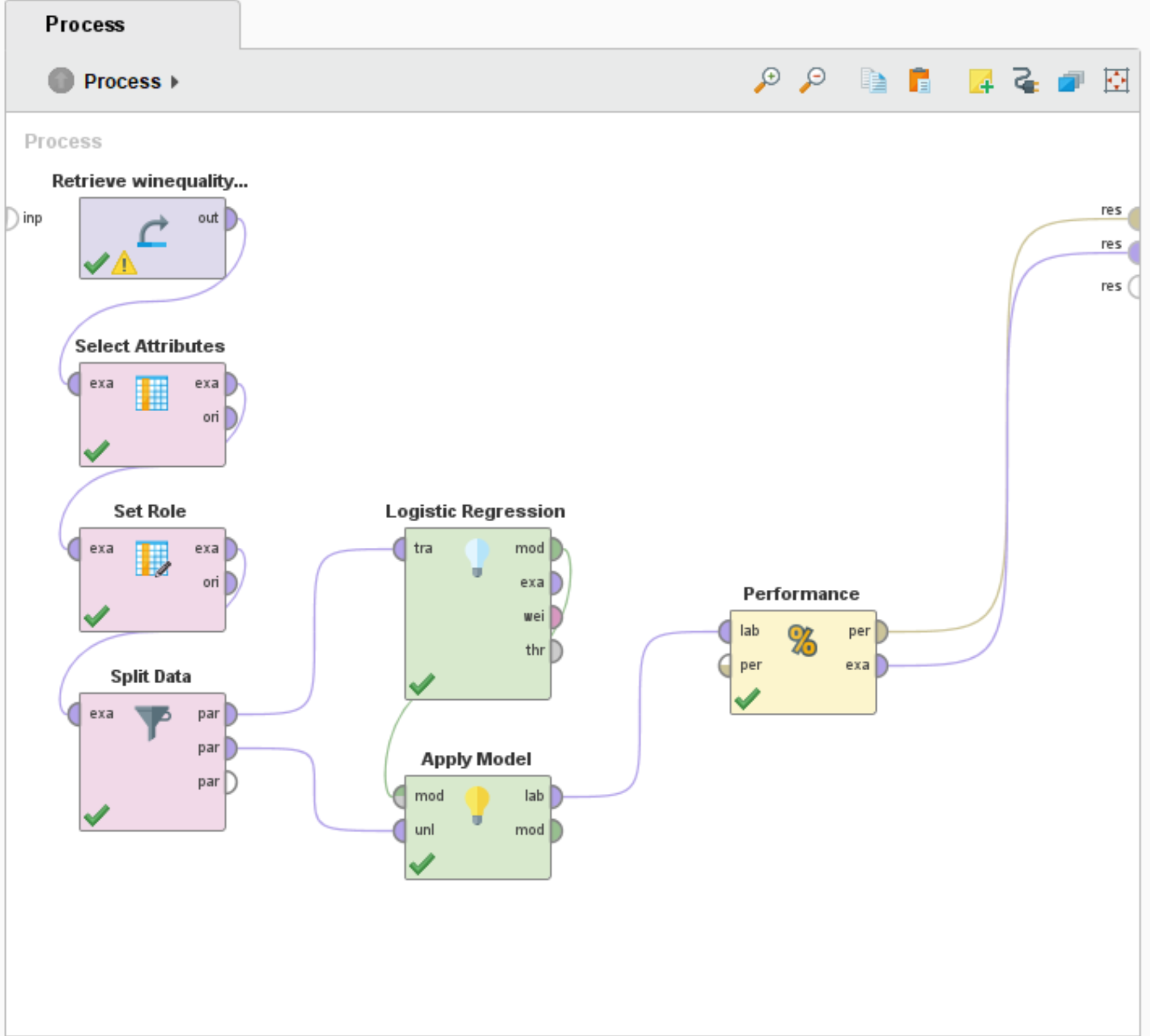
Below you can see the regression model with performance. The linear regression model can be used manually to predict the quality of the wine with the equation below.



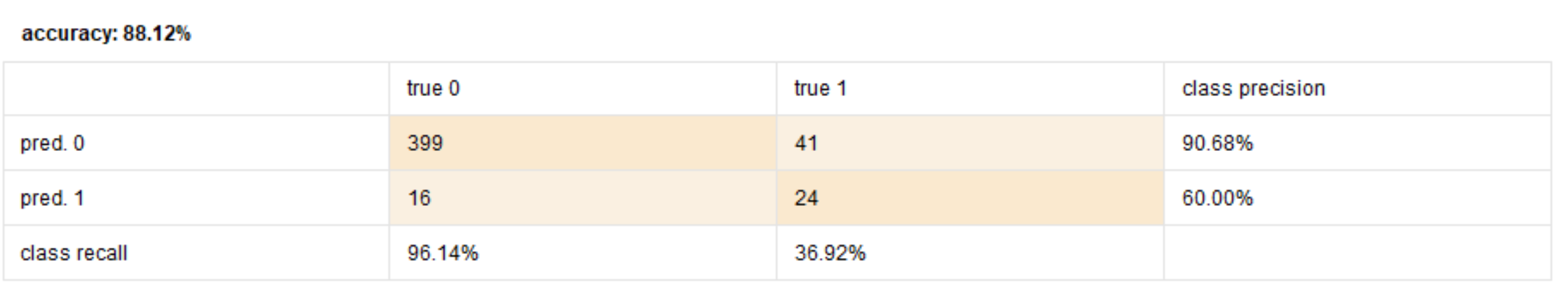


The model predicts within + or – 0.638 of the quality of the wine on a scale from 0-10 and the predictor variables only explain 34% of the variation in the quality of the wine. Given most values lied between 5 and 7, this is not a very good model for predicting the quality of the wine.

Below is the process I used to create the logistic regression model for the same data set. The only difference is the predicted variable is now a categorical variable with only two outcomes, either the wine is “good” or not.



Below is a table showing the performance and accuracy of the logistic regression model.



As you can see the model best predicted when a wine would be labeled a “not good” wine compared to being able to select a “good wine.