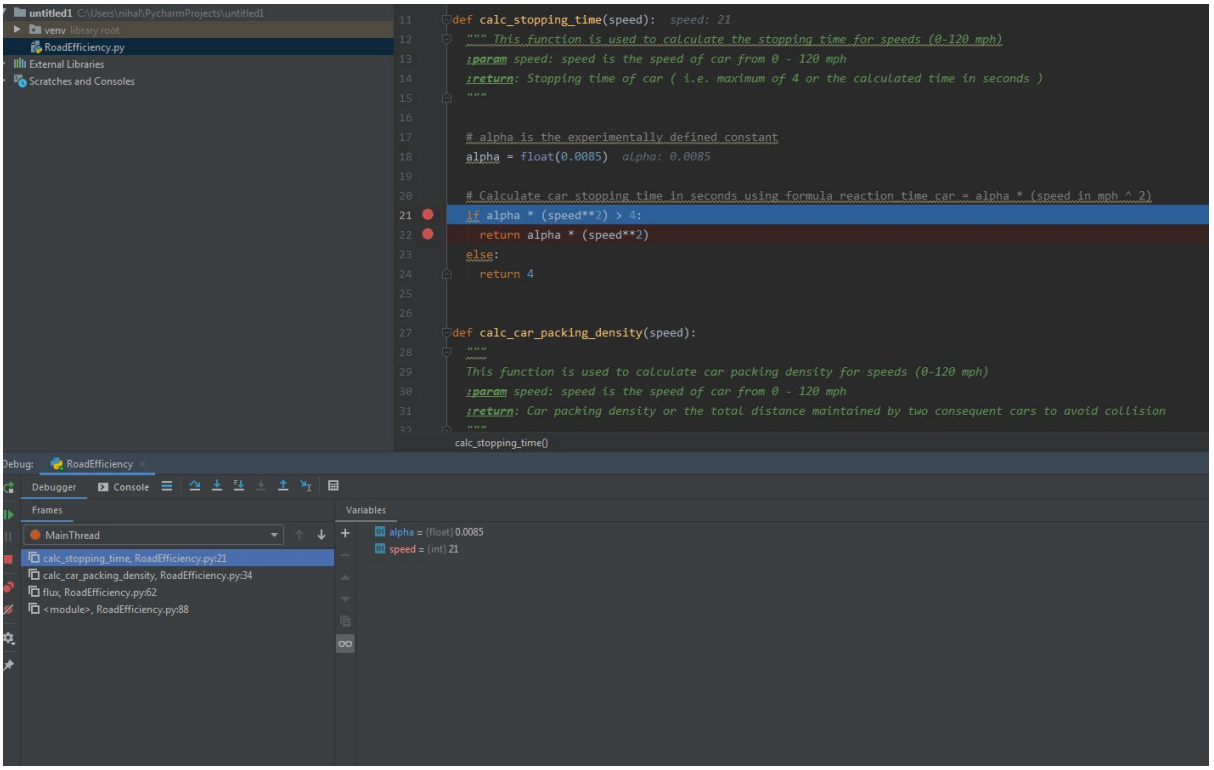
Big Data Analytics

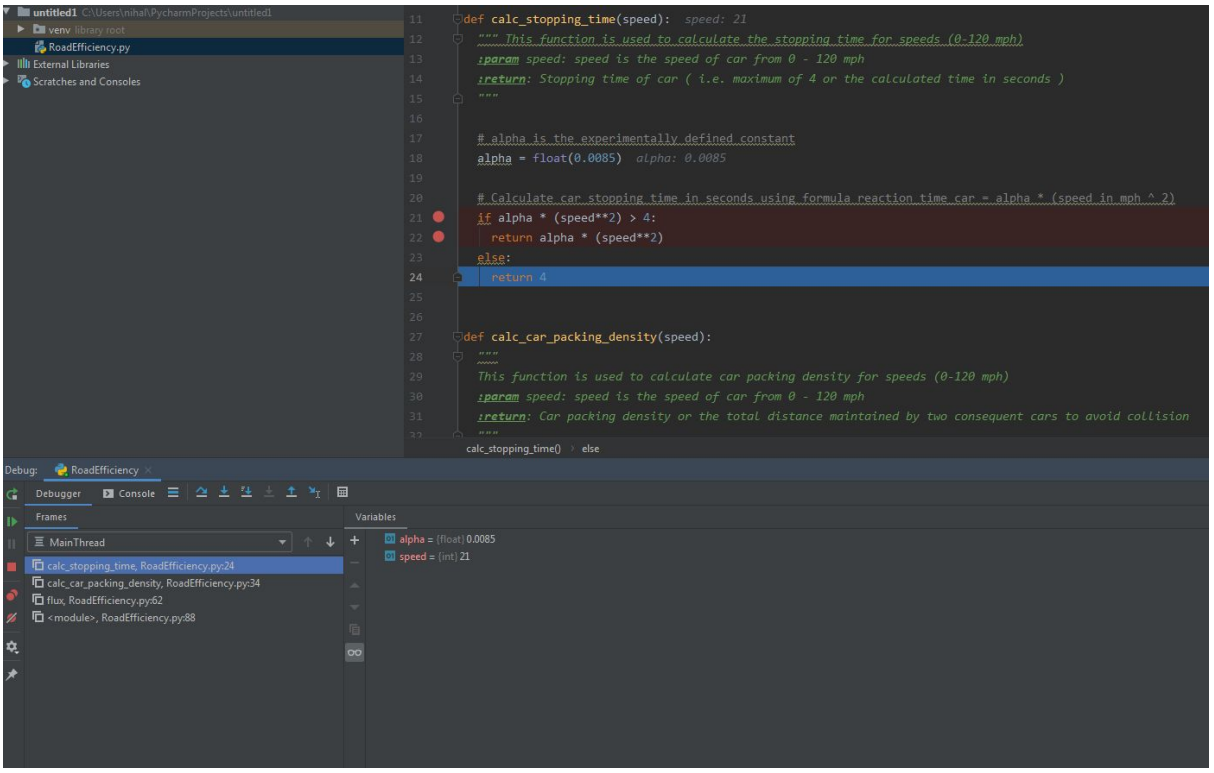
Nihal Surendra Parchand

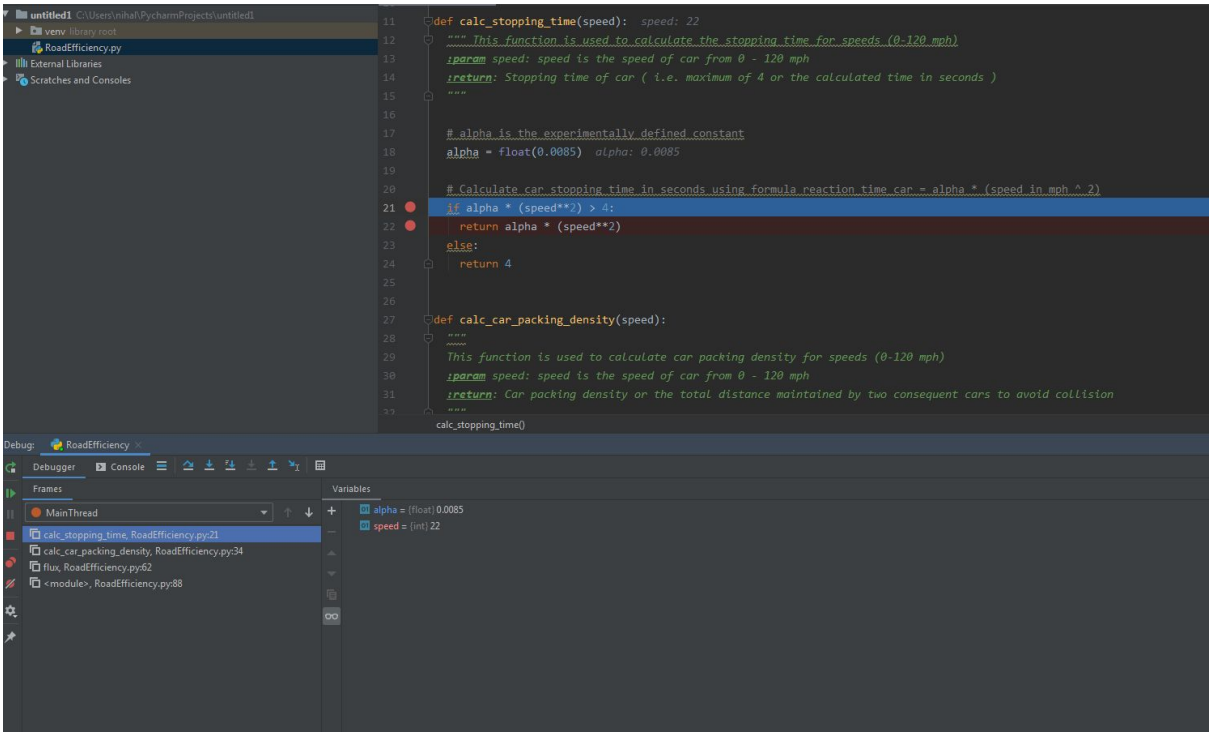
np9603@rit.edu

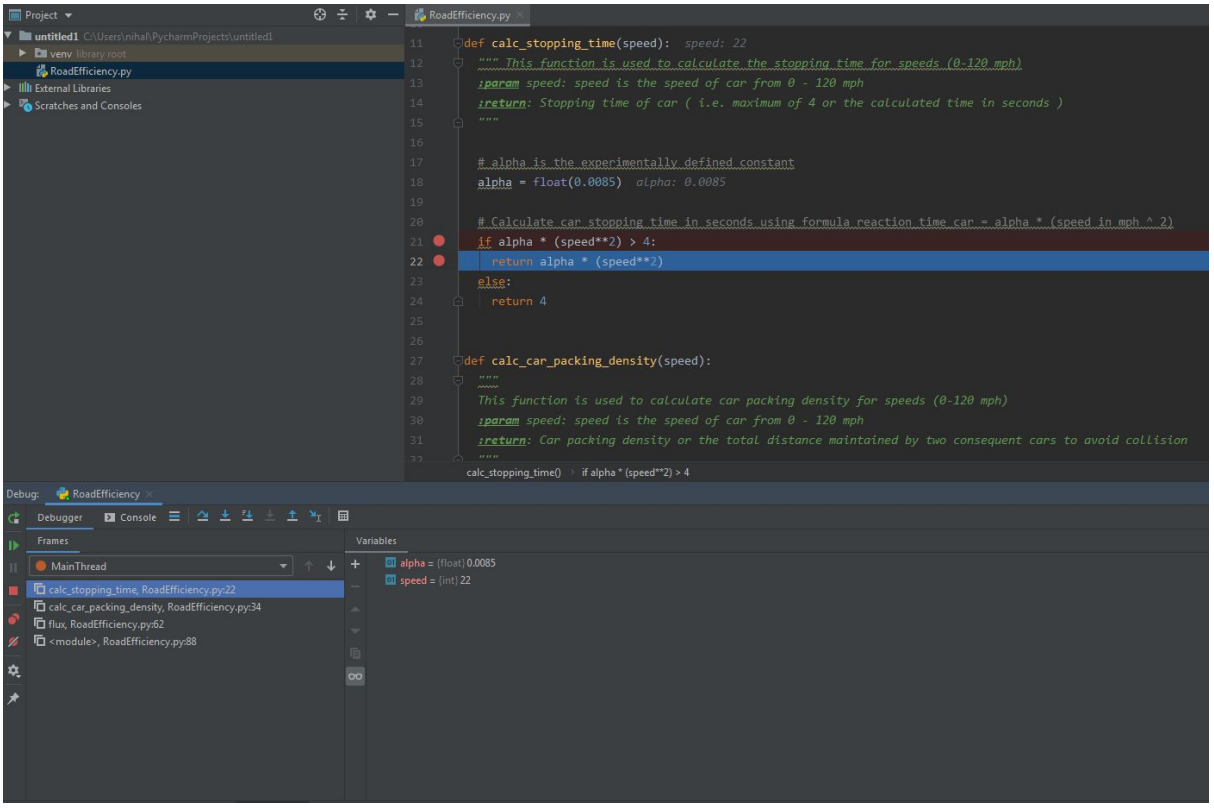
Homework 1

A. Screenshots of the debugger on Pycharm for finding out if I am getting the correct results for different speeds. I chose the speeds 21mph and 22mph for screenshots because for 21mph the stopping time calculated is less than 4 and for 22mph the stopping time goes above 4 which shows that the results are correct and debugging at this specific point in my code helped to finish my code quickly.

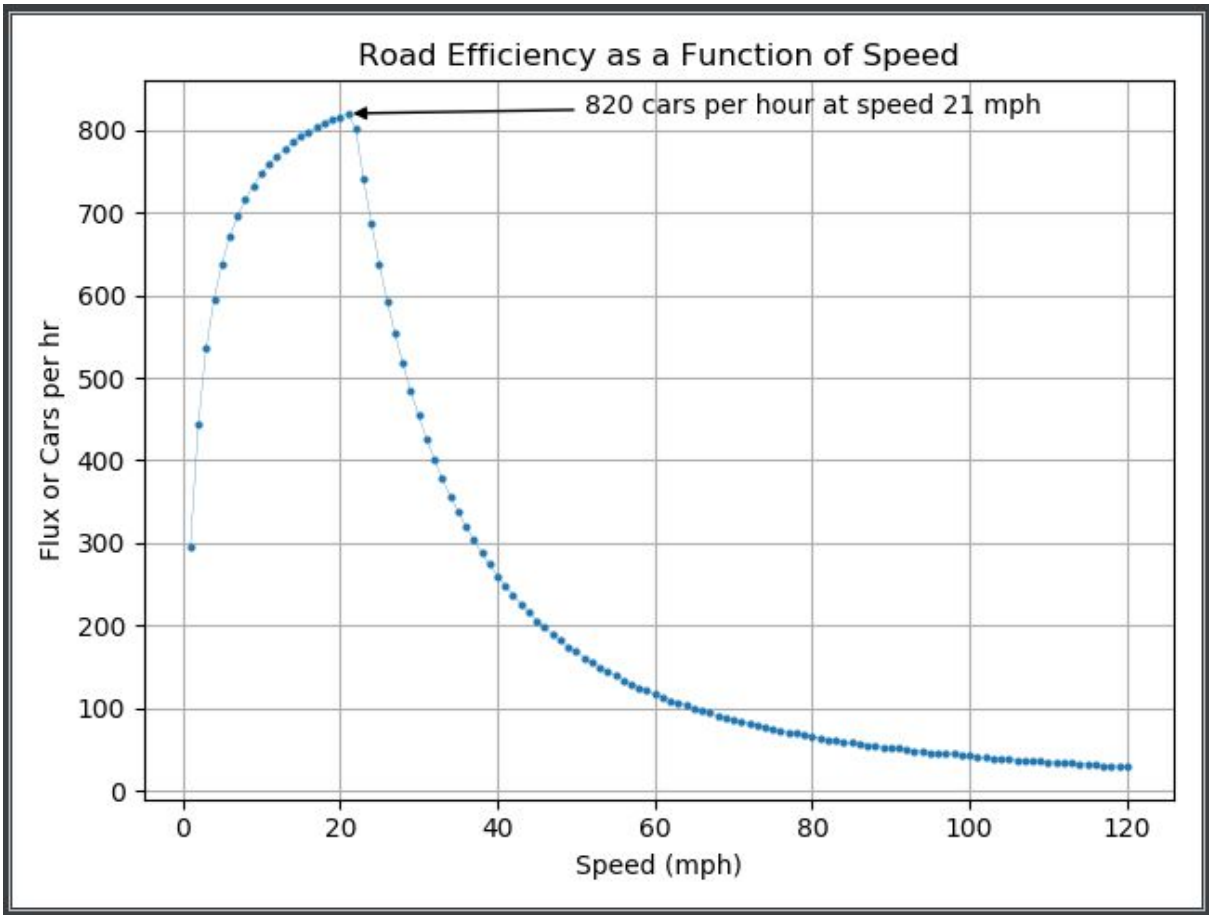



untitled1 C:\Users\nihal\PycharmProjects\untitled1
venv library root
RoadEfficiency.py
External Libraries
Scratches and Consoles
def calc_stopping_time(speed): speed: 21
""" This function is used to calculate the stopping time for speeds (0-120 mph)
:param speed: speed is the speed of car from 0 - 120 mph
:return: Stopping time of car (i.e. maximum of 4 or the calculated time in seconds)
"""
# alpha is the experimentally defined constant
alpha = float(0.0085) alpha: 0.0085
# Calculate car stopping time in seconds using formula reaction time car = alpha * (speed in mph ^ 2)
if alpha * (speed**2) > 4:
return alpha * (speed**2)
else:
return 4
def calc_car_packing_density(speed):
"""
This function is used to calculate car packing density for speeds (0-120 mph)
:param speed: speed is the speed of car from 0 - 120 mph
:return: Car packing density or the total distance maintained by two consequent cars to avoid collision
"""
calc_stopping_time() › else
Debug: RoadEfficiency
Debugger
Console
Frames
Variables
MainThread
calc_stopping_time, RoadEfficiency.py:24
calc_car_packing_density, RoadEfficiency.py:34
flux, RoadEfficiency.py:62
<module>, RoadEfficiency.py:88
alpha = {float} 0.0085
speed = {int} 21


untitled1 C:\Users\nihal\PycharmProjects\untitled1
venv library root
RoadEfficiency.py
External Libraries
Scratches and Consoles
def calc_stopping_time(speed): speed: 22
""" This function is used to calculate the stopping time for speeds (0-120 mph)
:param speed: speed is the speed of car from 0 - 120 mph
:return: Stopping time of car (i.e. maximum of 4 or the calculated time in seconds)
"""
# alpha is the experimentally defined constant
alpha = float(0.0085) alpha: 0.0085
# Calculate car stopping time in seconds using formula reaction time car = alpha * (speed in mph ^ 2)
if alpha * (speed**2) > 4:
return alpha * (speed**2)
else:
return 4
def calc_car_packing_density(speed):
"""
This function is used to calculate car packing density for speeds (0-120 mph)
:param speed: speed is the speed of car from 0 - 120 mph
:return: Car packing density or the total distance maintained by two consequent cars to avoid collision
"""
calc_stopping_time()
Debug: RoadEfficiency
Debugger
Console
Frames
Variables
MainThread
calc_stopping_time, RoadEfficiency.py:21
calc_car_packing_density, RoadEfficiency.py:34
flux, RoadEfficiency.py:62
<module>, RoadEfficiency.py:88
alpha = {float} 0.0085
speed = {int} 22


Project
untitled1 C:\Users\nihal\PycharmProjects\untitled1
venv library root
RoadEfficiency.py
External Libraries
Scratches and Consoles
RoadEfficiency.py
def calc_stopping_time(speed): speed: 22
"""This function is used to calculate the stopping time for speeds (0-120 mph)
:param speed: speed is the speed of car from 0 - 120 mph
:return: Stopping time of car (i.e. maximum of 4 or the calculated time in seconds)
"""
# alpha is the experimentally defined constant
alpha = float(0.0085) alpha: 0.0085
# Calculate car stopping time in seconds using formula reaction time car = alpha * (speed in mph ^ 2)
if alpha * (speed**2) > 4:
return alpha * (speed**2)
else:
return 4
def calc_car_packing_density(speed):
"""
This function is used to calculate car packing density for speeds (0-120 mph)
:param speed: speed is the speed of car from 0 - 120 mph
:return: Car packing density or the total distance maintained by two consequent cars to avoid collision
"""
calc_stopping_time() if alpha * (speed**2) > 4
Debug: RoadEfficiency
Debugger
Console
Frames
Variables
MainThread
calc_stopping_time, RoadEfficiency.py:22
calc_car_packing_density, RoadEfficiency.py:34
flux, RoadEfficiency.py:62
<module>, RoadEfficiency.py:88
alpha = {float} 0.0085
speed = {int} 22

B. A plot



C. The most efficient road speed was 21 mph.

D. The best efficiency was 820 cars per hour.

E.
The first part of the question was to calculate the stopping time of car in seconds which is the maximum of 4 seconds or the result calculated by the given formula (i.e. alpha * (mph ^ 2) where alpha = 0.0085 which is the experimentally defined constant).

Next, we calculate the car packing density which is the distance between the start of one car and the start of the next car + length of the car (12 feet). To calculate the total distance, we use the formula

Distance = Speed * Time

Since, the length of car is in feet, we convert the speed from miles per hour to feet per second by multiplying it with 1.4667. Thus, the formula for calculating the total distance becomes

Distance = Speed (miles per hour) * 1.4667 * stopping time + 12

Now, we calculate the total travel time (in seconds) for covering the total distance (safety gap + length of car) for every speed using the formula

Total travel time = Distance / speed (in mph) * 1.4667

To calculate flux or cars per hour, we divide 3600 by the total travel time (in seconds).

number_of_cars_per_hour_list and speed_list was used to maintain the track of flux for every speed (in mph) and for plotting the road efficiency graph using the plot function of Matplotlib library of Python.

NOTE - The for loop goes from 1-120mph because it does not make any sense to calculate flux for 0mph.

After completing this assignment I can conclude that sometimes the results after computation are quite significantly different as compared to our intuitions about a problem. I was expecting the most efficient speed would be around 50 mph but it turned out to be 21 mph. I also got to explore the Matplotlib library which helped me to plot the graph of speed vs flux. Data visualization helps to understand the results in a better way and draw conclusions from the plots built from the results.