UCF Physics PHZ 3150: Introduction to Numerical Computing
Midterm exam
Due October 5 2021; 2 pm!!!

For this Midterm exam we will be coding a little navigation program. Typically, a good navigation program is pretty complex with thousands of variables that need to be taken into account. We will do a simplified case where a traveler can only travel between 12 cities, on predetermined routes. Your task will be to let the driver know how long their trip will be and what the best route is when they want to go from A to B.
As we will slowly build the code to add more complexity, you will need to keep a good log of all the changes you do! Make sure you print out each output requested and add it to your log. This way we can check that you have done all steps and where something went wrong if it did.

Problems to Hand In: Your log is part of your Midterm exam. In one of the entries, it should identify the start and end of the Midterm and list the problem numbers in order. Keep notes about what you are doing for each problem, as well as the answers to the problems.

Problem 1 (**5 points total**).

A) (**2 points**) Start by making a new folder in your handin/ and homework folders called Midterm/. Your main homework file is a Python file named midterm_<username>.ipynb (or .py…). Save it. Remember to commit your file often and push it to GitHub (also, great backup!). Your name, assignment number, and the date should appear as comments at the top. Put the problems numbers in markdown comments, as well as any remarks or written answers you may make. If you need to comment something in the code (for clarity) do so with a normal comment. **Print the problem number (as in "Problem 1:") on a line by itself before each problem's output.** Use the print() function to print, don't just type the expression.

B) (**3 points**). For this program you will need the numerical module of Python. Do the appropriate import.

Problem 2 (**20 points total**). Let's start our program!

A) (**4 points**) We need to inform the user of our navigation system that we need from them a start and ending point for their trip. Print an informative sentence that lets the user know that we need them to input two cities (`start_city, target_city`). Their available options should be: Atlanta, Baltimore, Boston,

Charlotte, Dallas, Denver, Miami, Minneapolis, Orlando, Sacramento, Tampa and Washington (let the user know this!).

B) (**4 points**) Ask the input `start_city` and `target_city` for your trip

C) (**4 points**) Make a list `cities_list` with all the names of the possible cities the traveler can go from/to.

D) (**4 points**) Check if the input start and target cities are among the cities the traveler is allowed to travel to/from. If not, print an error statement that asks them to input another, appropriate city.

E) (**4 points**) Test that your code so far works. Run it with a start city: Paris and end city: Orlando. When the code complains for your start city, give new start city: Atlanta. Make a screenshot of your test and save it with the appropriate name to your Midterm folder.

Problem 3 (**20 points total**).

A) (**3 points**) From Webcourses/Files/Midterm get file "`distances_midterm.dat`". This file contains the total distance on the route between city A and B. If you open the file, you will see that it contains a 12 x 12 table (as we have 12 cities we can travel between) with the distances between cities. The i^{th} line and i^{th} column are a given city, so that the diagonal is 0s (as the distance from A to A is zero). The distance between Atlanta and Baltimore, e.g., is 678 miles so that line 0 (#5 of the file) column 1 of your data (i.e., the distance from Atlanta to Baltimore) is 678 (note that the same is true for line 1 [-Baltimore], column 0[-Atlanta] as this is the distance from Baltimore to Atlanta). Use the appropriate numpy function to read the data into variable `travel_data`.

B) (**4 points**) Next you will need to find which cities the traveler wants to travel from/ to when they give you their navigation start and end points (as in Problem 2.4) . Use your preferred method to get the index of the cities (name the indexes `first_city` and `second_city`) from your list.

C) (**4 points**) Using the `travel_data` and indexes `first_city` and `second_city`, find the distance the traveler will travel. Print an informative statement to let the driver know what the distance between his start and end city are (e.g., "between start_city and end_city you will travel 1000 miles.").

D) (**5 points**) Assume that the traveler can travel at an average speed of 75mph. Calculate the average travel time it will take them to go from their `start_city` to their `target_city`. Calculate how many days (`travel_time_days`), hours (`travel_time_hours`) and minutes (`travel_time_minutes`) the trip will take. Print an informative statement that lets the traveler know how long their trip

will be in days, hours and minutes (e.g., “your trip from start_city to end_city will take 1 day, 2 hours and 3 minutes. ”).

E) (**2 points**) Test that your program so far works. Run it from the `start_city`: Tampa to `target_city` : Orlando. Your trip should take 1hr and some minutes (note that our assumed average speed is higher than the legal speed limit, so you will get there faster than what, e.g., google maps suggests).

F) (**2 points**) Turn the program into a function that takes as input a list that contains the two cities (start, target) and outputs the necessary information (distance, time..). Make sure you include an appropriate informative docstring!

Problem 4 (**15 points total**). We will now change the function we made in Problem 3 to add an extra city as a pitstop.

A) (**2 points**) Modify your Problem 3 function to now ask the input `start_city`, a `pitstop_city` and a `target_city` for your trip. Let’s make a cross-country road trip. Run it for `start_city`: Boston, `pitstop_city`: Denver and `target_city`: Sacramento.

B) (**4 points**) Find which cities the traveler wants to travel from, pitstop at, and to in your list of cities. Use your preferred method to get the index of the cities (`firs_city`, `second_city` and now added `between_city`) from your list.

C) (**5 points**) Use the above indices and your `travel_data` to find the distance between `first_city` - `between_city` and `between_city`-`third_city`. Assuming that your traveler’s car can do 24.9 mpg on the highway and their tank holds16gallons, calculate how many refueling stops they will need to do on the first and the second leg of their trip (round the numbers; e.g., needing 3.2 stops doesn’t make sense - 4 does). Print an informative statement that lets the traveler know how many stops they will need to do on the two legs of their trip.

D) (**4 points**) Assume an average price of $2.4 per gallon and calculate how much the trip will cost in fuel. Print an informative statement that lets the traveler know the total (gas-only) value of their trip.

Problem 5 (**20 points total**) Let’s try to make a more proper navigation code and rum it for a random new trip. You will need to modify the function again so that your navigation finds out what your best pitstop is on its own, as follows:

A) (**3 points**) Ask the traveler to give you a start city and destination city only. You will need to then call the function from Problem 3, but change it as follows:

B) (**15 points total**) The basis of a navigation program is to get you from A to B in the shortest distance/time possible. Let's try to do this!

1. (**12 points**) Get the indexes of the cities you start from (`start_city`) and want to end at (`second_city`). You now need to scan the `travel_data` table to find the right pitstop city that takes you from A to B in the minimum total distance. Make sure that the straight A to B option is **not** taken into account, as we **need** to do a pitstop in some city C. Scan all possible combinations A→C, C→B and keep the index (`min_tot_dist_ind`) of pitstop city C that minimizes the total distance (A→C) + (C→B). Print an informative statement that lets the traveler know that they will need to stop in city C, and what the total distance traveled will be.

2. (**3 points**) Run this program for a trip from Baltimore to Orlando. What is your pitstop city?

3. (**2 points**) Let the traveler know how many stops for gas they will need to do in every leg of the trip (from Baltimore to Orlando) and how much the total gas cost of the trip will be.

Problem 6 (**10 points**) Prepare and submit your homework. Write what you did to make and submit the zip file into your log. Don't forget to commit your finalized code and push it to GitHub. When satisfied, close the log, copy it to your handin directory and run the commands to make and submit the zip file. Turn the file in on WebCourses.