APS106 – Lab #8

Preamble

This week you will practice file I/O by writing a program to analyze experimental results and select parameters that maximize an objective function.

Deliverables

For this lab, you must submit the five functions listed below within a single file named 'lab8.py' to MarkUS by the posted deadline.

Functions to implement for this lab:

- `parse_material_costs`
- `parse_test_results`
- `analyze_test_results`

Use appropriate variable names and place comments throughout your program.

The name of the source file must be "lab8.py".

Five test cases are provided on MarkUs to help you prepare your solution. **Passing all these test cases does not guarantee your code is correct.** You will need to develop your own test cases to verify your solution works correctly. Your programs will be graded using ten secret test cases. These test cases will be released after the assignment deadline.

IMPORTANT:

- Do not change the file name or function names
- Do not use input() inside your program

Problem

This week we will imagine we are working on developing cables that will be used in cable-stayed bridges. Your objective to identify the cable material and diameter that achieves the highest maximum tensile strength per unit cost of the cable. That is, you are trying to find the combination of material and cable diameter that maximize the following expression:

$$\frac{Maximum\ Tensile\ Strength}{(material\ cost + diameter\ cost)}$$

To find the optimal selection, your firm is performing tests with different combinations of materials and cable diameters. The results of these tests are recorded in csv files. Because there are new tests being ordered and reported each day, you have been asked to write a program that can read these csv files and report the top three material-diameter combination from each batch of tests.

To do this, you will need to complete three functions:

- `parse_material_costs`
- `parse_test_results`
- `analyze_test_results`

a colleague of yours has also provided you with two functions that you can use in your programs to compute the cost and maximum tensile strength per unit cost. These functions are named

- `cable_cost`
- `cable_strength_per_cost`

Descriptions of these functions and their input parameters are provided within their respective docstrings.

Part 1 – Parse Material Costs

In this part of the lab, you will complete the parse_material_costs function which reads a csv file specifying material names and their associated costs and returns a dictionary of material-cost key-value pairs. The input to this function is a string specifying the name of a csv to read. The format of this file is shown in the table below.

Material	Cost
material-X	11
material-Y	8.5
material-Z	20
composite-1	31
composite-2	55

The first column contains the names of materials and the second column contains the cost of each material. The function should return a dictionary with the material names (strings) as keys and the costs (floats) as values. For the table above, the returned dictionary would contain the following key-value pairs:

```
{"material-X" : 11.0, "material-Y" : 8.5, "material-Z" : 20.0, "composite-1"
: 31.0, "composite-2" : 55.0}
```

Note the returned dictionary should not contain the "Material" and "Cost" headers in the first row. A file, material_costs1.csv, is provided to help you test your function.

Part 2 – Parse Test Results

In this part of the lab, you will complete the parse_test_results function which extracts maximum tensile strength test results for different materials and diameters from a csv file and returns a nested dictionary that contains the maximum tensile strength for each material and diameter. The input to this function is a string specifying the name of the file containing the maximum tensile strength test results

The format of the tensile strength test results is shown below

Material\Diameter	1	3	5.5	10	100
material-X	24.3	50.3	70.9	99.3	203.5
material-Y	13.3	33.3	55.9	81.8	185.4
material-Z	30.3	35.3	40.8	57.3	110.1
composite-1	40.2	45.3	49.3	58.3	70.9
composite-2	28.2	33.9	38.6	50.3	89.4

Each cell outside of the first row and column represents the maximum tensile strength achieved using a material-diameter combination. The material used for each test are given in the first column and the diameter for each test is given in the first row. For example, from the above table, we can see that a cable made from 'material-Z' with a diameter of 10, achieved a maximum tensile strength of 57.3[1]. Similarly, a cable made with 'composite-1' and a diameter of 3 had a maximum tensile strength of 45.3. Reading the table, the values within a row are tensile strengths for cables made with the same material but different diameters. The values within a column are tensile strengths for cables made from different materials with the same diameter.

Your function will need to extract the maximum tensile strength for each material-diameter combination and then store the maximum tensile strength in a nested dictionary. The dictionary will have the following format

```
{material_name0 : {diameter0 : max_strength,
                   diameter1 : max_strenght,
                   diameter2 : max_strength,
                   ...
                  },
material_name1 : {diameter0 : max_strength,
                  diameter1 : max_strenght,
                  diameter2 : max_strength,
                  ...
                 },
 ...
 }
```

That is, the function returns a nested dictionary where the results for each material are stored within an inner dictionary of diameter-max tensile strength per unit cost pairs.

For the example table above, the function should return the following dictionary:

[1] We won't worry about the units for all of these measures for the stake of this assignment

```
{"material-X" : {1.0   : 24.3,
                 3.0   : 50.3,
                 5.5   : 70.9,
                 10.0  : 99.3,
                 100.0 : 203.5
                },
 "material-Y" : {1.0   : 13.3,
                 3.0   : 33.3,
                 5.5   : 55.9,
                 10.0  : 81.8,
                 100.0 : 185.4
                },
 "material-Z" : {1.0   : 30.3,
                 3.0   : 35.3,
                 5.5   : 40.8,
                 10.0  : 57.3,
                 100.0 : 110.1
                },
 "composite-1" : {1.0   : 40.2,
                  3.0   : 45.3,
                  5.5   : 49.3,
                  10.0  : 58.3,
                  100.0 : 70.9
                 },
 "composite-2" : {1.0   : 28.2,
                  3.0   : 33.9,
                  5.5   : 38.6,
                  10.0  : 50.3,
                  100.0 : 89.4
                 }
}
```

Note you may not assume that the diameters tested will always be the same. That is, your function will need to read the diameter values in the first row of the file and use these values to organize the data in the dictionary. Two files, test_results1.csv and test_results2.csv are provided to help you test your code.

Part 3 – Analyze Test Results

In this final part of the lab, you will use the functions from parts 1 & 2 as well as the cable_strength_per_cost function to find the three material-diameter combinations with the highest maximum tensile strength per unit cost. The inputs to this function are:

- test_result_file_name – A string specifying a csv file containing maximum tensile strength test results
- material_cost_filename – A string specifying a csv file containing material names and their associated costs
- diameter_fixed_cost – A float representing the fixed cost to manufacture a cable with any diameter
- diameter_marginal_cost – A float representing an additional per unit cost to manufacture cables with a diameter greater than a manufacturer specified threshold
- diameter_cost_threshold – A float representing the manufacturer specified maximum cable diameter before an additional marginal cost is applied.

For more details on the last three parameters, see the cable_strength_per_cost function docstring.

The function should return a three-element tuple specifying the three material-diameter combinations with the highest maximum tensile strength per unit cost. The elements of this tuple should be sorted according to the maximum tensile strength per unit cost from highest to lowest. Each element in the returned tuple should be a tuple containing the following information: a string representing the material, a float representing the diameter, and a float representing the maximum tensile strength per unit cost.

Developing test cases for this function is left as an exercise. You may consider creating your own material cost and test result files as well as trying different values for the diameter cost related parameters.