

COMP 1516 – Programming Fundamentals with Python – Assignment 1

Instructor	Mike Mulder (mmulder10@bcit.ca or Slack)
Total Marks	25
Due Dates	June 5 th , 2021 at midnight in the Assignment 1 dropbox on the Learning Hub.

Overview

For this assignment you will be generating one of three different reports for one of three different inputs.

Input Types:

- Phone Data
- Tablet Data
- Laptop Data

Report Types:

- Text
- CSV
- JSON

The input type and report type will be specified as command line parameters to the Python script (main.py) that you will be creating. You are provided with a Python module (data.py) that has functions that return a list of data for each of the above input types. You will also create a Python module (reports.py) with functions for generating each of the above report types.

Your assignment must have no syntax errors and must successfully run (i.e., no runtime errors) for all combinations of the above inputs and outputs. **If the script or provided unit test cannot be run, you will receive a mark of zero on this assignment.**

Marks will be based largely on the output from the provided unit test file (test_main.py) when run. Full marks will only be given to implementations that are reasonably efficient, i.e., there should NOT be a lot of copy and paste code.

Requirements

The following table identifies the requirements for this assignment.

Requirement	Marks
Create a script in a main.py that takes in command line arguments.	4
The command line arguments are as follows:	

<p>main.py <input type> <report type></p> <p>Where:</p> <ul style="list-style-type: none"> • Input Type is phone, tablet or laptop • Report Type is text, csv or json <p>The following errors should be printed to the console for invalid parameters:</p> <ul style="list-style-type: none"> • Invalid number of command line arguments. • Input type must be either phone, tablet or laptop. • Report type must be either text, csv or json. <p><i>Make sure you follow best practices for having a main function in this script.</i></p> <p>You may also have other functions in this script that calculate the various statistics for the input data. For example:</p> <pre>def calculate_statistics(input_type): """ Calculates the statistics for the given input type and data """ # Return values such as: # current_datetime, device_name, num_devices, # avg price, min price, max price, median ram, oses</pre>	
<p>Create a module called reports in a reports.py file. Creates functions for the following three reports in this module: text report, csv report and json report.</p> <p>The function signatures should look like this:</p> <pre>def text_report(date, type, num_devices, avg_price, min_price, max_price, median_ram, oses): pass def csv_report(date, type, num_devices, avg_price, min_price, max_price, median_ram, oses): pass def json_report(date, type, num_devices, avg_price, min_price, max_price, median_ram, oses): pass</pre>	3
<p>The output format for the text report should be as follows:</p> <p>Date: <date time or report></p> <p>Device: <device type – Mobile Phone, Tablet or Laptop></p>	3

<p>Number: <number of unique devices in list> Median RAM: <median ram> GB Average Price: \$<calculated average price> Minimum Price: \$<minimum price in list> Maximum Price: \$<maximum price in list> Operating Systems: <comma separated list of operating systems></p> <p>Used formatted strings to generate the text output. <i>The operating system should be the operating system value AND the version (i.e., Android 5.1.1 Lollipop)</i> <i>The datetime format should be YYYY-MM-DD HH:MM</i></p>	
<p>The output format for the csv report should be as follows:</p> <p><datetime>,<device type>,<number of devices>,<mediuan ram>,<calculated average price>,<minimum price in list>,<maximum price in list> ,<::: separated list of operating systems></p> <p>Use the <u>join</u> function to generate the csv output. <i>The operating system should be the operating system value AND the version (i.e., Android 5.1.1 Lollipop)</i> <i>The format of the list of operating systems should look like:</i> <i>Android 5.1.1 Lollipop:::Android 7.1 Nougat:::Android 9.0 Pie</i> <i>The datetime format should be YYYY-MM-DD HH:MM</i></p>	3
<p>The output format for the json report should be as follows:</p> <pre>{ "data_time": "<date time of report>", "device_type": "<device type>", "number": <number of devices>, "median_ram": <median ram in gb>, "average_price": <calculated average price>, "min_price": <minimum price in list>, "max_price": <maximum price in list>, "operating_systems": "<comma separated list of operating systems>" }</pre> <p>Use formatted strings to generate the json output. <u>Do NOT use the Python json modules for this, it will not produce the same results as the unit test expects.</u> <i>The operating system should be the operating system value AND the version (i.e., Android 5.1.1 Lollipop)</i> <i>The datetime format should be YYYY-MM-DD HH:MM.</i></p>	3
<p>Correct Output for each input type and report type (see <i>Example Run Configurations to Test</i> below and the sample_output.txt file provided).</p>	9

Full marks will be given for correct output and efficient implementation (limited redundant code).	
--	--

Example Run Configurations to Test

There are nine run configurations that must be tested:

- main.py phone text
- main.py phone csv
- main.py phone json
- main.py tablet text
- main.py tablet csv
- main.py tablet json
- main.py laptop text
- main.py laptop csv
- main.py laptop json

Sample Output and Unit Tests

The **sample_output.txt** file provides the expected output for each of the above run configurations. Your script must produce the same output.

The **test_main.py** unit test will verify that the output of your script matches the requirements above. This will be used to mark your submission against the requirements. In any discrepancies between the requirements above the unit tests, the unit tests will be considered the correct output against which your submission will be marked.

Submission

Submit the following Python files in a **zipfile** called **assignment1.zip** to the Assignment 1 dropbox in D2L:

- main.py – Your implementation of the main script
- reports.py – Your implementation of a reports module
- data.py – Provided with the lab
- *Note: You will NOT need to make any changes to data.py or test_main.py for this lab.*

Make sure you followed naming best practices and included DocString in all functions.

Grading Summary

Implementation As per the requirements described above AND the test_main.py unit test.	25 marks
Marks will be subtracted for violations of naming standards or missing DocString. (-0.5 mark each)	
Total	25 marks

Best Practices

- Variable names should be lower_snake_case
- Function names should be lower_snake_case
- All functions should include a DocString comment