Assignment 1 (10 Marks)

Due: Friday 16 March 2018 at 6:00pm

Introduction

With the current Winter Olympics and upcoming Commonwealth Games, the first assignment will process some sporting event data[1]. The data you will be processing is the results from events. An event management system records the results for each participant in an event. It maintains the data in the official event database. The data is sent to the games website for public display.

The program you are implementing for this assignment is the intermediary between the event data and the website. The process your program will implement is called "data cleansing". The purpose is to identify errors in the data and to either correct or remove the errors before the data is sent to the website.

Data

The data is stored in a comma-separated values (CSV) file. You are provided with a `for` loop in the `main()` function that will read the data from the file. The raw data file that you need to clean is called `athlete_data.csv`. The data can be thought of as a matrix, with rows and columns. There are twelve columns in the data. Each row represents the result for one athlete in a competition. The format rules for each column are:

Num.	Data	Max Characters	Format Rules
1	Athlete Identifier	6	whole number
2	Event Name	30	letters, digits, space, hyphen & apostrophe
3	Athlete's First Name	30	letters, digits, space, hyphen & apostrophe
4	Athlete's Surname	30	letters, digits, space, hyphen & apostrophe
5	Country Code	3	uppercase letters
6	Place (current or final)	3	whole number or one of three strings (DNS[2], DNF[3], PEN[4])
7	Score	6	whole or floating-point number
8	Time	8	whole or floating-point number
9	Medal	6	Gold, Silver or Bronze
10	Olympic Record (if set)	8	whole or floating-point number
11	World Record (if set)	8	whole or floating-point number
12	Track Record (if set)	8	whole or floating-point number

Columns 1 to 5 must have data in every row. If column 6 contains a whole number then there must be a legal value in columns 7 or 8, but not both. If column 9 contains a legal value then column 6 must have an appropriate value (e.g. Gold must be for first place, Silver for second place and Bronze for third place). Columns 10 to 12 do not need to correspond to first place in column 6 (e.g. it is possible for an athlete to set a world record in a qualifying round and then not win a medal in the final rounds). Similarly, the value in columns 10 to 12 does not need to be the same as the value in columns 7 or 8 (e.g. the athlete does worse in the final than when setting the record in a qualifying round). If column 11 contains a legal value then it must equal the value in column 10.

[1] For the curious, the inspiration for this assignment comes from http://goldmedalsystems.com. Students in more advanced courses have designed modules corresponding to some of the systems offered by the company.

[2] Did Not Start

[3] Did Not Finish

[4] Penalty

Processing

The website does not use the athlete's identifier number. Your program must remove the first column from the data from every row, including rows with corrupt data. Invalid data for the athlete's identifier number can be ignored.

Your program must ensure that the data in a column does not exceed the maximum specified number of characters. For the event name, and athlete's first name and surname, if they exceed the maximum specified number of characters the name is to be truncated to the maximum allowed number of characters. For all other columns, if the data exceeds the maximum specified number of characters then the data in the row is considered to be corrupt.

Your program must also ensure that the data in a column is correct according to the specified format rules. If the country code is three letters but they are not all in uppercase then your program should convert them to uppercase. If the medal column contains the strings Gold, Silver or Bronze, but with different letter case, then your program should convert the strings to be exactly Gold, Silver or Bronze (i.e. first letter is uppercase and all others are lowercase). Any other illegal values, according to a column's format rules, mean that the row is considered to be corrupt.

Your program should also ensure that the rules regarding data being in a particular column are also correctly implemented. If these rules are broken the row is considered to be corrupt.

Every row that is identified as being corrupt is to be output to the result file, as it was read in from the raw data file (minus the athlete id column), and with an extra column added to that row containing the text CORRUPT.

For example, if the raw data file contained the following data:

```
1,Men's Moguls,Rohan, Charleman-Davies Charleman-Davies,AuS,17,73.96,,,,,
2,Men's Moguls,Matt,Graham,AUS,2,82.57,,silver,,,
3,Men's Moguls,Brodie,Summers,AUS,DNS,,,,,,
4,Men's Moguls,James,Matheson,Aus,fourteen,75.98,,,,,
```

Your program should process it and save the following data to the clean result file.

```
Men's Moguls,Rohan, Charleman-Davies Charleman-Dav,AUS,17,73.96,,,,,
Men's Moguls,Matt,Graham,AUS,2,82.57,,Silver,,,
Men's Moguls,Brodie,Summers,AUS,DNS,,,,,,
Men's Moguls,James,Matheson,Aus,fourteen,75.98,,,,,,CORRUPT
```

Design

You are provided with a `main()` function inside the file `assign1.py`. This function provides the main loop in which you are to place your code to clean the data. The `for` loop reads one row from the file and provides it as a string called `row`. Your code needs to process the data in `row`. You are provided with code at the end of the `for` loop that saves the cleaned data to the output file called `athlete_data_clean.csv`.

The `main()` function declares a boolean variable called `corrupt` that you are to use to flag when you determine that the data in a row is corrupt. There are two string variables, `row` and `row_to_process` that contain the data from the row being processed. The variable `row_to_process` is to be used for processing. The variable `row` is to keep the original data read from the file (minus the athlete id column). This allows the logic at the end of the loop to correctly save the valid, cleaned row, or the original row data with the flag CORRUPT.

Note that the program does not have any user interaction. All input and output is via files.

Hints

The focus of this assignment is on demonstrating your ability to implement a simple program using basic procedural programming constructs. It is possible to pass the assignment if you only implement some of the specified functionality. You should design and implement your program in stages, so that after the first stage you will always have a working previous partial implementation.

In thinking about the stages to implement the program, consider that some features are easier to implement than others. Focus on getting the easier features working first. (e.g. Removing the athlete identifier, checking for maximum character lengths of columns and checking for numeric values are probably relatively easy.) You may wish to leave comparing values between columns until after you have some of the easier parts of the program working correctly.

When checking the column data against the format rules there are some built-in functions in Python that you may find useful. It is possible to determine if a string or character is a letter or digit using the functions `isalpha()` and `isdigit()`. You use these functions by calling them from a string variable (e.g. `string_variable.isalpha()` and `string_variable.isdigit()`). It is also possible to convert a string to uppercase characters via `string_variable.upper()`. You can determine the length of a string by using the `len()` function (e.g. `len(string_variable)`).

You are provided with three utility functions in the file `assign1_utilities.py`. It is already imported into the provided code in `assign1.py` and is ready to be used. The three utility functions are `truncate_string`, `get_column` and `replace_column`. You may find these useful in helping to process the data in a row. Read the docstrings for these functions to see how they are to be used.

You should make use of functions to make the structure of your logic easier to understand. A function should perform a single task. Do not try to write a large function that does many things. It is better to have several functions, where each one does a single task in your overall logic.

Extension

With placing and medals, it is possible for athletes to tie for a medal. They both win the medal for their place and then the next place skips to a lower medal. (e.g. An event might have two gold medals, no silver and one bronze. Alternatively, an event might have one gold medal and two silver medals, and no bronze.) Your program could check for these conditions.

If medals are awarded for an event, three medals must be awarded, unless there are less than three athletes with a place. (DNF, DNS and PEN are not considered to be a place in an event.) Not all athletes in the raw data file will have a place, as some events will not be completed at the time of receiving the data. Your program could enforce this three medal rule. (Note that the provided raw data file does not contain all data from the Olympics. This means that the place positions will not be a contiguous range. [i.e. For an event you might only have athletes in places 1, 2, 3, 7, 9 and 14.] The raw data file will have athletes with places equivalent to all the medals awarded in the event, unless the data is corrupt.)

You could also have one or more master files that contain all the valid country codes, event names and athlete names. These could be used to check that the data in those columns in the raw data file matches names for those master files.

Submission

You must submit your assignment electronically through Blackboard. For information on submitting through Blackboard, please read:

https://web.library.uq.edu.au/library-services/it/learnuq-blackboard-help/learnuq-assessment/blackboard-assignments

You should submit your assignment as a single Python file called `assign1.py` (use this name – all lower case). You do **not** need to submit the `assign1_utilities.py` utility file or the raw data CSV file. If you have created any master files for the extension to the assignment, then these must be submitted as well.

You may submit your assignment multiple times before the deadline – only the **last** submission will be marked.

Late submission of the assignment will **not** be accepted. Do not wait until the last minute to submit your assignment, as the time to upload it may make it late. In the event of exceptional personal or medical circumstances that prevent you from handing in the assignment on time, you may submit a request for an extension. See the course profile for details of how to apply for an extension:
http://www.courses.uq.edu.au/student_section_loader.php?section=5&profileId=92705

Requests for extensions **must** be made no later than 48 hours prior to the submission deadline. The application and supporting documentation (e.g. medical certificate) **must** be submitted via my.UQ. You **must** retain the original documentation for a minimum period of six months to provide as verification should you be requested to do so.

Assessment and Marking Criteria

This assignment assesses course learning objectives:

1. apply program constructs such as variables, selection, iteration and sub-routines,
3. read and analyse code written by others,
5. read and analyse a design and be able to translate the design into a working program,
6. apply techniques for testing and debugging

Criteria	Mark
Programming Constructs	
• Program is well structured and readable	1
• Identifier names are meaningful and informative	1
• Algorithmic logic is appropriate	1
• Functions are used to split logic into some meaningful and useful blocks, with data passed as parameters and returned appropriately	1
• Functions are well-designed, simple cohesive blocks of logic	1
Sub-Total	5
Functionality	
• Column size limits implemented correctly	1
• Column format rules implemented correctly	1
• Conditional constraints between columns implemented correctly	0.5
• Two or more extension rules implemented correctly	0.5
Sub-Total	3
Documentation	
• Comments are clear and concise, without excessive or extraneous text	0.5
• All functions having informative docstring comments	1
• Significant blocks of program logic are clearly explained by comments	0.5
Sub-Total	2
Total	**/ 10**

Your mark will be limited to a maximum possible value if your program does not execute or crashes when executing. Your assignment will be limited to a maximum mark of

- 3, if the program contains syntax or semantic errors that prevent it from executing;
- 4, if the program crashes when executing using the provided raw data file;
- 5, if the program crashes when executing using the test data file;
- 5, if the program if none of the required functionality is implemented.

It is your responsibility to ensure that you have adequately tested your program to ensure that it is working correctly. You have been provided with a sample file of data in `athlete_data.csv` and with the results that your program should produce in `athlete_data_clean.csv`, if you have implemented all functionality, including the extension functionality. Your submitted assignment will be tested with a different raw data file that will contain other issues that your program should detect. You should create your own data files to test all features of your implementation.

In addition to providing a working solution to the assignment problem, the assessment will involve discussing your code submission with a tutor. This discussion will take place in week 5, in the practical session to which you have signed up. You **must** attend that session in order to obtain marks for the assignment.

In preparation for your discussion with a tutor you may wish to consider:

- any parts of the assignment that you found particularly difficult, and how you overcame them to arrive at a solution; or, if you did not overcome the difficulty, what you would like to ask the tutor about the problem;
- whether you considered any alternative ways of implementing a given function;
- where you have known errors in your code, their cause and possible solutions (if known).

It is also important that you can explain to the tutor how each of the functions that you have written operates (for example, if you have used a for loop or a while loop in a function, why this was the right choice).

Marks will be awarded based on a combination of the correctness of your code and on your understanding of the code that you have written. **A technically correct solution will not achieve a pass mark unless you can demonstrate that you understand its operation.**

A partial solution will be marked. If your partial solution causes problems in the Python interpreter please comment out the code causing the issue and we will mark that. Python 3.6.4 will be used to test your program. If your program works correctly with an earlier version of Python but does not work correctly with Python 3.6.4, you will lose **at least** all of the marks for the functionality criteria.

Please read the section in the course profile about plagiarism. Submitted assignments will be electronically checked for potential plagiarism.

Detailed Marking Criteria

Criteria	Mark		
Programming Constructs	**1**	**0.5**	**0**
• Program is well structured and readable	Code structure highlights logical blocks and is easy to understand. Code does not employ global variables. Constants clarify code meaning.	Code structure corresponds to some logical intent and does not make the code too difficult to read. Code does not employ global variables.	Code structure makes the code difficult to read.
• Identifier names are meaningful and informative	All variable and function names are clear and informative, increasing readability of the code.	Most identifier names are informative, aiding code readability to some extent.	Most identifier names are not clear or informative, detracting from code readability.
• Algorithmic logic is appropriate	Algorithm design is simple, appropriate, and has no logical errors. Control structures are well used to implement expected logic.	Algorithm design is not too complex or has minor logical errors. A few control structures are a little convoluted.	Algorithm design is overly complex or has significant errors. Many control structures are used in a convoluted manner (e.g. unnecessary nesting, multiple looping, …).
• Functions are used to split logic into some meaningful and useful blocks, with data passed as parameters and returned appropriately	Functions represent useful logical functionality and parameters and return values are appropriate.	Functions represent some useful logical functionality and parameters and return values are usually appropriate (e.g. too functions or they are too large).	Functions are not used or are not useful logical blocks. Parameters or return values are not used or are not appropriate.
• Functions are well-designed, simple cohesive blocks of logic	Program is well-designed, splitting the logic into an appropriate number of general functions, where each function performs a single cohesive logical task.	Program is split into a reasonable number of general functions.	Program may use functions correctly, but some functions are large blocks of logic that implement multiple tasks.

Functionality	**1**	**0.5**	**0**
• Column size limits implemented correctly	All size limit rules are implemented correctly.	Size limit rules are implemented correctly for at least event and athlete names.	Size limit rules are not implemented correctly.
• Column format rules implemented correctly	All format rules are implemented correctly.	At least three of the five different format rules are implemented correctly.	Less than three of the five different format rules are implemented correctly.
• Conditional constraints between columns implemented correctly		At least two of the four different constraints between columns are implemented correctly.	Less than two of the four different constraints between columns are implemented correctly.
• Two or more extension rules implemented correctly		At least two of the rules listed under the heading "Extension" are implemented correctly.	Less than two of the rules listed under the heading "Extension" are implemented correctly.
Documentation	**1**	**0.5**	**0**
• Comments are clear and concise, without excessive or extraneous text		Comments provide useful information that elaborates on the code. These are useful in understanding the logic and are not too wordy.	Many comments irrelevant or do not provide any detail beyond what is already obvious in the code. The excessive length of some comments obscures the program logic.
• All functions having informative docstring comments	All docstrings provide a complete, unambiguous, description of how to use the function.	Most docstrings provide a complete description of how to use the function.	Some docstrings provide an inaccurate description of how to use the function, or there are functions without docstrings.
• Significant blocks of program logic are clearly explained by comments		In-line comments are used to explain logical blocks of code (e.g. significant loops or conditionals).	In-line comments are missing in places where they would have been useful. Or, in-line comments are irrelevant or repeat what is already clear in the code.