|  |  |
| --- | --- |
| Course Name | ITD 3773 – Software Systems Integration |
| Instructor | Mark L. Pranger |
| Student Name | Matthew Whitehead, Crystal Reimche, Aaron Laster |
| Due date | August 21, 2022 |
| Grade | 100 |
| Grading Comments | Group 2 Comments: Great Video, good documentation, the HTML file format was surprising and well done |

Fixed Width to CSV / CSV to Fixed Width User Manual

**Table of Contents**

[Introduction](#_Introduction)

[Installation](#_Script_and_Dependencies)

[Configuration](#Config_Files)

[BoolFixedWidth](#BoolFixedWidth)

[Delimiter](#Delimiter)

[FullName](#FullName)

[FirstName](#FirstName)

[LastName](#LastName)

[Street](#Street)

[City](#City)

[State](#State)

[Zipcode](#Zipcode)

[Phone](#Phone)

[Classes and Usage](#Classes_Usage)

[Helper\_Functions](#Helper_Function)

[CSV\_to\_FW](#CSV_to_FW)

[FW\_to\_CSV](#FW_to_CSV)

[Constants](#Constants)

[DocStrings](#Docstrings)

### Introduction

This program was designed to take in a file, manipulate the data, and then create an output file in the format the user requires. The input file can be either in a Comma Separated Value (CSV) format or a Fixed Width format. By simply modifying the configuration file, the user can tell the program what format the input file is in, what delimiter to use, and how to set up the columns for the output file. If the input file is in CSV format, the user selects if headers are to be added to the output file, if so, what the header names are. Along with the header option, the user can enter how wide each column should be and what character to use to fill in the space if the data does not fill it in. If the input file is in a Fixed Width format, the user can again select if they want headers added to the output file or not. They also can select what character to use for the delimiter when the output CSV is created.

### Script and Dependencies Setup

The first step to run the program is to install the dependencies. The only non-standard dependency is Pandas, which is a data analysis library that is used in this project to read and write rows and columns to a CSV.

Install Pandas using PIP. Open a command prompt window and type: **pip install pandas**

Text

Description automatically generated

It will download and install all Pandas, along with all its dependencies.

Text

Description automatically generated

Once you see the **Successfully installed** message, you’re now ready to import Pandas at the top of your driver script.

Our project utilizes a driver script named **Main.py**, this is recommended. The project contains 4 classes in total: **Constants.py**, **csv\_to\_fw.py**, **fw\_to\_csv.py**, and **helper\_functions.py**, along with a configuration file named **inputconfig.ini** Their usages are defined below. They just need to be included in the project folder, then added to the imports at the top of the driver script to be called. Their usage is described below, with the docstrings summary.

The driver script will need to include a way to check the config file to determine if the input file is fixed width or CSV. You’ll then do your data processing from there. The **inputconfig.ini** file includes a Header section that can be used to write the configured column headers before processing the data.

### Configuration Files

The **inputconfig.ini** config file is how the user customizes how the input file is laid out and how the output file is expected to look after the program runs.

**[BOOLFIXEDWIDTH]**

This indicates if the incoming file is a fixed width formatted document or not. It can only be set to true or false.

[BOOLFIXEDWIDTH]fixed\_width = true

**[DELIMITER]**

If the input file is in CSV format, set this value to the delimiter that separates each column of data.

If the input file is in a fixed width format, set this value to the delimiter you want that separates each column of data.

Delimiter examples: comma ( , ), semi-colon ( ; ), hashtag ( # ), pipe ( | ), slashes( / \ )

[DELIMITER]delimiter = ,

**[HEADER]**

Set CREATE\_HEADER to true if you want your output file to have column headers generated on the output file. Set to false if you do not. You can also modify each of the values you want to be written for each header directly below.

[HEADER]CREATE\_HEADER = true  
*; These are the header names, modify the right side as needed*FULL\_NAME = Full Name  
FIRST\_NAME = First Name  
LAST\_NAME = Last Name  
STREET = Street  
CITY = City  
STATE = State  
ZIP = Zip  
PHONE = Phone

**[FULLNAME]**

Set SPLIT\_FULL\_NAME to true if the input file has a first and last combined in one column and the output file needs to separate the name into two columns. Otherwise, set it to false to not separate the name.

If the output file will be in a fixed width format, set the following configurations:

Set FULL\_NAME to how many characters are allowed in the column.

Set PADDING\_SIDE to left if the padding will be to the left of the data, or right if the padding will be to the right of the data.

Set PADDING\_CHAR to a single character. If padding with a space, type in the word space.

E.g., space, #, \*, -, |

[FULLNAME]SPLIT\_FULL\_NAME = trueFULL\_NAME = 35PADDING\_SIDE = rightPADDING\_CHAR = space

**[FIRSTNAME]**

\*\*\* The following values only need to be set if [FULLNAME] is set to false. \*\*\*

Set FIRST\_NAME to how many characters are allowed in the column.

Set PADDING\_SIDE to left if the padding will be to the left of the data, or right if the padding will be to the right of the data.

Set PADDING\_CHAR to a single character. If padding with a space, type in the word space.

E.g., space, #, \*, -, |

[FIRSTNAME]FIRST\_NAME = 15PADDING\_SIDE = rightPADDING\_CHAR = space

**[LASTNAME]**

\*\*\* The following values only need to be set if **[FULLNAME]** is set to false. \*\*\*

Set LAST\_NAME to how many characters are allowed in the column.

Set PADDING\_SIDE to left if the padding will be to the left of the data, or right if the padding will be to the right of the data.

Set PADDING\_CHAR to a single character. If padding with a space, type in the word space.

E.g., space, #, \*, -, |

[LASTNAME]LAST\_NAME = 20PADDING\_SIDE = rightPADDING\_CHAR = space

**[STREET]**

Set STREET to how many characters are allowed in the column.

Set PADDING\_SIDE to left if the padding will be to the left of the data, or right if the padding will be to the right of the data.

Set PADDING\_CHAR to a single character. If padding with a space, type in the word space.

E.g., space, #, \*, -, |

[STREET]STREET = 30PADDING\_SIDE = rightPADDING\_CHAR = space

**[CITY]**

Set CITY to how many characters are allowed in the column.

Set PADDING\_SIDE to left if the padding will be to the left of the data, or right if the padding will be to the right of the data.

Set PADDING\_CHAR to a single character. If padding with a space, type in the word space.

E.g., space, #, \*, -, |

[CITY]CITY = 20PADDING\_SIDE = rightPADDING\_CHAR = space

**[STATE]**

Set STATE to how many characters are allowed in the column.

Set PADDING\_SIDE to left if the padding will be to the left of the data, or right if the padding will be to the right of the data.

Set PADDING\_CHAR to a single character. If padding with a space, type in the word space.

E.g., space, #, \*, -, |

[STATE]STATE = 2PADDING\_SIDE = rightPADDING\_CHAR = X

**[ZIPCODE]**

Set ZIP to how many characters are allowed in the column.

Set PADDING\_SIDE to left if the padding will be to the left of the data, or right if the padding will be to the right of the data.

Set PADDING\_CHAR to a single character. If padding with a space, type in the word space.

E.g., space, #, \*, -, |

[ZIPCODE]ZIP = 5PADDING\_SIDE = rightPADDING\_CHAR = *#*

**[PHONE]**

Set PHONE to how many characters are allowed in the column.

Set PADDING\_SIDE to left if the padding will be to the left of the data, or right if the padding will be to the right of the data.

Set PADDING\_CHAR to a single character. If padding with a space, type in the word space.

Set STRIP\_CHAR to a single character that will be removed from the phone number. E.g., ( ) -

Set REPLACE\_CHAR\_WITH to a single character that will be replacing the character that was stripped. E.g., \*, -, 'no space'

[PHONE]PHONE = 10PADDING\_SIDE = rightPADDING\_CHAR = spaceSTRIP\_CHAR = -REPLACE\_CHAR\_WITH = no space

### Classes and usage

***helper\_functions.py***

Write\_all\_rows() is the method that does the data processing for each line and it can be called in a for loop to iterate over each row of a CSV document. It expects an output file object passed into it to write to as well as the class name to use (csv\_to\_fw.py or fw\_to\_csv.py). It then uses those class functions to process the data.

Write\_fullname\_header() is a method that’s called from the driver script before write\_all\_rows() if you wish to write headers to the top of the document.

There exists a function in Helper\_functions.py for each of these headers and they all do the same thing as write\_fullname\_header(), with identical usage. It’s recommended to call each one in the order that you intend for them to be shown, which is the order that the data is represented in the input document.

Setup\_fixedwidth\_columns() is another method that’s called from the driver script when the config file reflects that the input file is a fixed width document. It’s called to indicate where each column starts and ends within the line.

The remainder of the functions in helper\_functions.py are calculating functions for the fw\_to\_csv.py and csv\_to\_fw.py classes and are not user-facing.

***csv\_to\_fw.py***

This class is used to apply the indicated column formats from the config file and is called from within the helper\_functions.py class to return each of the formatted datapoints. It needs to be initialized in the driver script to be passed into the write\_all\_rows() function as the second parameter. This is the only way that it interacts with the driver script and otherwise is a backend class.

***fw\_to\_csv.py***

This class is used to apply the indicated column formats from the config file and is called from within the helper\_functions.py class to return each of the formatted datapoints. It needs to be initialized in the driver script to be passed into the write\_all\_rows() function as the second parameter. This is the only way that it interacts with the driver script and otherwise is a backend class.

***constants.py***

Constants.py is used to read the configs and apply the configured formatting to the columns. It is used in the driver script in the calls to the write functions from helper\_functions.py as well as any readers to indicate what the configured delimiter is.

### Docstrings





