:: Span League Score ::

User and Technical documentation

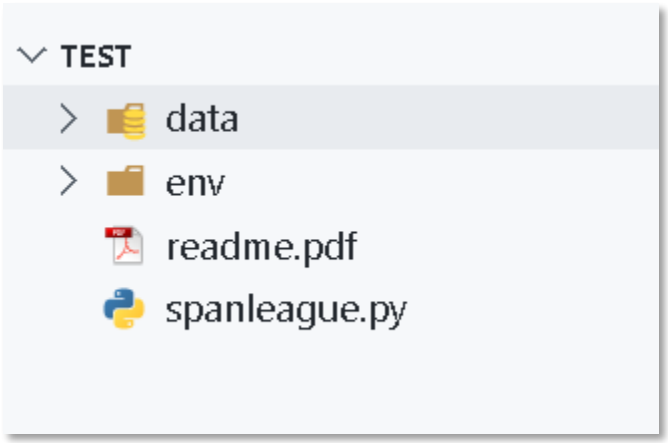
Mexico City. Aug 26, 2022

Index

	Page
User documentation	3
Technical documentation	6
Append – Requirements	8

User Documentation

This is a command-line application developed in Python that covers the requirements described in the appendix. The compressed test.zip file is delivered. You must unzip the file; from there it will display the test folder with the following files:



- Test folder, there are the test case files.
- Env folder, there is the generated environment for Python (*py -m venv env*).
- Readme file is the instructions file.
- Spanleague.py file contains the Python program, see the technical description of each module below.

Execution.

To run the program, it is recommended to activate the python environment.

```
\test> .\env\Scripts\activate
```

```
(env) PS D:\MIGUEL\Miguel\MIGUEL\2022\FullStack\test>
```

Run the program in Python.

```
(env) PS D:\MIGUEL\Miguel\MIGUEL\2022\FullStack\test> py spanleague.py
```

It displays a menu with the following options:

```
(env) PS D:\MIGUEL\Miguel\MIGUEL\2022\FullStack\test> py spanleague.py
SPAN Digital ::Score League::
Capture the score {Team1 Score1, Team2 Score2}. Instructions: 0.- Exit | 1.- Show results | 2.- Load file ::
```

0.- Exit the application

1.- Show the results

2.- Load a data file

Capture from the command line.

In the case of capturing the score, type directly on the command line and enter.

```
(env) PS D:\MIGUEL\Miguel\MIGUEL\2022\FullStack\test> py spanleague.py
SPAN Digital ::Score League::
Capture the score {Team1 Score1, Team2 Score2}. Instructions: 0.- Exit | 1.- Show results | 2.- Load file ::Lions 3, Snakes 3
```

The application will validate if there are no duplicate teams or if the marker is a non-integer data, for example:

```
Capture the score {Team1 Score1, Team2 Score2}. Instructions: 0.- Exit | 1.- Show results | 2.- Load file ::Lions 3, Lions 3
Validation::  Lions 3, Lions 3 :: Teams equals arent valid
```

```
Capture the score {Team1 Score1, Team2 Score2}. Instructions: 0.- Exit | 1.- Show results | 2.- Load file ::Lions 3.5, Snakes 3
Validation::  Lions 3.5, Snakes 3 :: The scores arent whole number
```

In any case: valid or incorrect, the application redisplays the line with the menu to be able to capture more information, for example:

```
(env) PS D:\MIGUEL\Miguel\MIGUEL\2022\FullStack\test> py spanleague.py
SPAN Digital ::Score League::
Capture the score {Team1 Score1, Team2 Score2}. Instructions: 0.- Exit | 1.- Show results | 2.- Load file ::Lions 3, Snakes 3
Capture the score {Team1 Score1, Team2 Score2}. Instructions: 0.- Exit | 1.- Show results | 2.- Load file ::Tarantulas 1, FC Awesome 0
Capture the score {Team1 Score1, Team2 Score2}. Instructions: 0.- Exit | 1.- Show results | 2.- Load file ::Lions 1, FC Awesome 1
Capture the score {Team1 Score1, Team2 Score2}. Instructions: 0.- Exit | 1.- Show results | 2.- Load file ::Tarantulas 3, Snakes 1
Capture the score {Team1 Score1, Team2 Score2}. Instructions: 0.- Exit | 1.- Show results | 2.- Load file ::Lions 4, Grouches 0
```

At any time by capturing option ”1” you can view the results

```
Capture the score {Team1 Score1, Team2 Score2}. Instructions: 0.- Exit | 1.- Show results | 2.- Load file ::1
SPAN Digital ::Results::
1. Tarantulas, 6 pts
2. Lions, 5 pts
3. FC Awesome, 1 pt
4. Snakes, 1 pt
5. Grouches, 0 pts
```

Load from a file.

To load the data from a file, option “2” must be captured in the menu

```
Capture the score {Team1 Score1, Team2 Score2}. Instructions: 0.- Exit | 1.- Show results | 2.- Load file ::2
Input the file name {e.g. test.txt}::
```

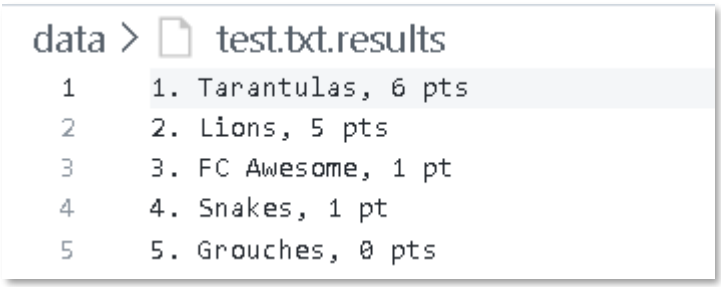
Get the path and name of the file.

```
Input the file name {e.g. test.txt}:: data/test.txt
```

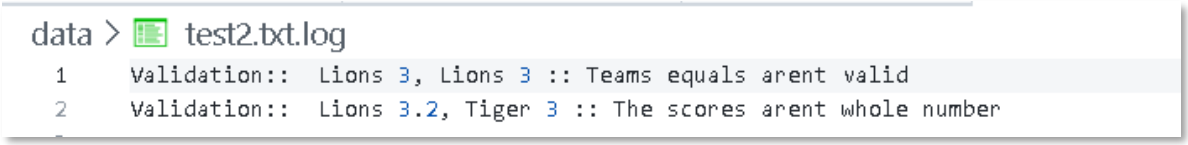
If the file is correct, the application will generate two output files:

```
Capture the score {Team1 Score1, Team2 Score2}. Instructions: 0.- Exit | 1.- Show results | 2.- Load file ::2
Input the file name {e.g. test.txt}:: data/test.txt
SPAN Digital ::Results::
->See the result in the file:data/test.txt.results
->See the log in the file:data/test.txt.log
```

- *Results, which contains the results-



- *Log, where are the possible identified errors-



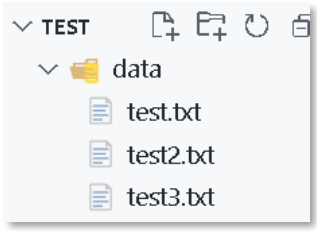
Exit application.

To exit the application, you must capture the option "0" from the command line-

```
Capture the score {Team1 Score1, Team2 Score2}. Instructions: 0.- Exit | 1.- Show results | 2.- Load file ::0
Exit
(env) PS D:\MIGUEL\Miguel\MIGUEL\2022\FullStack\test>
```

Test files.

In the data folder there are three test files, which can be executed and are the cases analyzed-



Technical Documentation

The application is inside the spanleague.py file. It was developed using functions and without any additional library other than the one in the environment. The application is divided into the following functions:

```
def results(teams):
```

Format the result to present it this way:

1. Destroyers, 6 pts
2. Lions, 3 pts
3. Rogues, 3 pts
4. Snakes, 3 pts

```
def update(teams,team,points):
```

Find the team and update the data with the points.

```
def calculate(teams,data,fileFlag):
```

Calculate the point allocation based on the following rules.

- If team1 1 score > Team2 3 3 points are added to the winning team, zero to the losing team.
- If there is a tie, 1 point is assigned to each team.

```
def validation(data,team1,team2):
```

Validates the data and displays or sends it to a file.

- The teams are different.
- Th score is integer.

```
def writefile(results,filename,label,extension):
```

Create and write the data in the defined file (Results or error log).

```
def readfile(teams,filecontent,filename):
```

Reads the data from the bookmarks file and from there generates the results and error log files.

```
def loadfile(teams):
```

Open the bookmark data file and call the read from file function, additionally handle read from file exceptions.

Main function

Manages the main menu waiting for data capture, file selection or display of results.

Append - Requirements

Read the problem statement, code a working solution (valid input and output will be provided) and supporting tests using a language of your choice. Be prepared to explain your solution during a review.

The Problem

We want you to create a production ready, maintainable, testable command-line application that will calculate the ranking table for a league.

Input/output

The input and output will be text. Either using stdin/stdout or taking filenames on the command line is fine.

The input contains results of games, one per line. See “Sample input” for details.

The output should be ordered from most to least points, following the format specified in “Expected output”.

You can expect that the input will be well-formed. There is no need to add special handling for malformed input files.

The rules

In this league, a draw (tie) is worth 1 point and a win is worth 3 points. A loss is worth 0 points. If two or more teams have the same number of points, they should have the same rank and be printed in alphabetical order (as in the tie for 3rd place in the sample data).

Guidelines

This should be implemented in a language with which you are familiar. We would prefer that you use python, java, Golang, or Scala, if you are comfortable doing so. If none of these are comfortable, please choose a language that is both comfortable for you and suited to the task like, node, ruby, or C.

If you use other libraries installed by a common package manager (ruby gems/bundler, npm,

pip), it is not necessary to commit the installed packages.

We write automated tests, and we would like you to do so as well.

If there are any complicated setup steps necessary to run your solution, please document them.

Platform support

This will be run in a unix-ish environment (OS X). If you choose to use a compiled language, please keep this in mind. Please use platform-agnostic constructs where possible (line-endings and file-path-separators are two problematic areas).

Sample input:

```
Lions 3, Snakes 3
Tarantulas 1, FC Awesome 0
Lions 1, FC Awesome 1
Tarantulas 3, Snakes 1
Lions 4, Grouches 0
```

Expected output:

```
1. Tarantulas, 6 pts
2. Lions, 5 pts
3. FC Awesome, 1 pt
3. Snakes, 1 pt
5. Grouches, 0 pts
```