rank

A CLI application that produces a point table based on matches.
Supported OS: **Linux** only
Detailed instruction with images at: https://github.com/m4tice/rank

Project architecture

```
.
├── ...
├── rank
│   ├── ...
│   ├── rank  <-- main cli
│   ├── test  <-- test scripts
│   ├── venv  <-- virtual environment
│   ├── data  <-- directory for input and output text files
│   └── requirements.txt
└── ...
```

how to run

Clone the repository and run the following lines:
* Activate the virtual environment:

```
source ./venv/bin/activate
```

- Generate the ranked table:

```
rank -i <input_file.txt>
```

Example:

```
rank -i data/sample-input.txt
```

Application architecture

The application is designed to be an ETL pipeline with 3 phases:
* *Extract*: extract data from the input text file and put them into a list.
* *Transform*: transform the extracted data to the required format.
* *Load*: load the result into a text file.

Extract

The phase mainly deals with string manipulation. The goal is to transform the raw data into structured data (in form of rows and columns).
The sub-functions of this phase:
* `extract_match_info()` : used to split teams' names and scores and put into a list.
* `extract_team_info()` : used to support `extract_match_info()` .

Transform

The phase mainly deals with data transformation. The goal is to create a point table and add the points based on matches and predefined rule, and rank them from high to low, alphabetically.

The workflow is as follow:
* First, we gather the unique teams' names based on matches and create a point_table dictionary. The dictionary is chosen for the purpose of quick access of a specific item.
* Second, we adding points to point_table dictionary based on input matches (3pts win and 1pt draw).
* Third, we sort the point_table based on points, alphabetically.
* Finally, we add rank to teams based on points.

The sub functions of this phase:

* `create_table_dict()` : create point_table dictionary.
* `point_analysis()` : add points to point_table.
* `sort_point_table()` : sort point_table.
* `rank_table()` : add ranks to point_table.

Load

This is the final phase, used to export the sorted rank table into a text file named `generated_output.txt` in the `data` directory.

Tests

The application is able to handle the following exception:
* *Non-exsisted file*: when user tries to input a non-exsisted file.
Test:

```
rank -i data/unknown.txt
```

- *Empty file*: when user tries to input an empty text file.
  Test:

```
rank -i data/sample-input-empty.txt
```

- *Wrong data format*: when user tries to input text file with wrong data format.
  Test:

```
rank -i data/sample-input-wrong-format.txt
```

OR
You can run the following, which covers all test cases :)

```
python tests/test_cli.py
```

The tests directory contains the test scripts for all the functions and sub-functions of this application.
Example:

```
python tests/test_extract.py
python tests/test_transform.py
python tests/test_load.py
```