Task 1:
In this task we made a class called Team with input parameters name and trigram. In addition each team holds: number of games won, number of games lost, number of games drawn, total points, number of goals for, number of goals against, goal difference. All the values have functions to both set and to get the values.

The functions setGoalDifference and setTotalpoints uses the other values to calculate goal difference and total points, and are therefore called to update these values when other values are changed with their set function.

Task 2:
In this task we made a class called Game with input parameters trigramHometeam, hometeamScore, trigramVisitorteam and visitorteamScore that are stored into a list. All the values have functions to both set and to get the values.

Task 3:
In this task we made a class called Championship.
For instance of a championship object, a dictionary called teams, list called games and an empty gametable is created.

- The dictionary has team trigram as key and a list containing trigram and teamname.
- The list contains game-objects from Task 1

In addition the championship contains functions to look for a team, create a new team and so on.

Task 4:
Here we have created a championship called “cup”. The championship contains three teams, that's added to the ‘cup’ using the “.newTeam”-function. Six games are added to the championship using the “newGame”-function. Run codeline # to print out the list containing the six Game-objects. Run codeline #450 to check it.

Task 5:
In each of the three classes mentioned above(Team, Game, Championship) we have created a print function that writes all information associated with the class into a text file. Run codeline #459 to test it.

Task 6:
Here we have created the importChampionship which reads all data regarding a championship in a file and stores it into the championship-class from task 3. We have implemented a try/except validation to make sure the file is correctly imported. To import the correct information from the file we have chosen to split the strings on "#Name" and "#Home code". As we assume the file has the same setup as PremierLeague2019-2020-Description.tsv, we have made the function dependent of the file starting with “#Name Code” and having a transition with the line #Home code Home score Visitor code Visitor score. Run codeline #469 to test it.

Task 7:
In this task we traverse the data structure that encodes the championship and update the current teams values such as games drawn, total points etc. Run codeline #474.

Task 8:
Here we have created the function getRanking(). The function appends teams with their score into a list, then sorts this new list on points and returns it.
Run codeline #478 to test it.

Task 9:
This task writes a championship ranking into a .tsv-file. Run codeline #482 to test it.

Task 10:
Here we have created the function updateGameList which returns a dictionary with team trigram as key and a list as a value. The list contains 2 elements; games played at home and games played as visitors. Run codeline #487 to test it.

Task 11:
Here we have created two auxiliary-functions to help get the info. The function 'getGame' returns a Game-object from a hometeam and visitorteam. The 'getGameTable'-function gets all home-matches for all teams and appends this into a 2d-list. The function 'printGameTable' writes the 2d-list into a .tsv file. Run codeline #491 to test it.

Task 12:
Here we have created the function printRankingHomeAndAwayAndCombined which prints General ranking, Home ranking and Visior ranking into a text file.

The functions are based on the two functions getRankingByHome and getRankingByVisitor. They both work the same except they count points differently.

The getRankingByHome function uses the updateGameList function from Task 10. Then looks for games where the current team plays at home sums all points for the hometeams in the games found, before appending the team and its total points into an empty list(homeranking). Before returning the list it is sorted by total points. Run codeline #495 to test it.

Task 13:
Here we created two auxiliary-functions to help us. 'rankingByGoalsFor' gives us the ranking represented in a list, considered the For-goals only. 'rankingByGoalsAgainst' gives us the ranking represented in a list also, considered the Against-goals only. It sorts by least goals against first. The 'printedRankingGeneralForAndAgainst' writes to file a normal ranking (all parameters considered), a ranking based on number of goals for and a ranking based on number of goals against. Run codeline #500 to test it.