CS 8, Winter 2015

Homework Assignment #? (draft)

Assignment Overview

This project allows you to practice more data retrieval, summarization, and tabulation tasks and sharpens your skills with all kinds of data structures: string, list, and dictionary.

Background

The National Basketball Association (NBA) is the USA's premier professional men's basketball league. It has 30 teams; 29 in the United States and one in Canada. It is an active member of USA Basketball (USAB), which is recognized by the International Basketball Federation as the National Governing Body (NGB) for basketball in the United States. The NBA is one of the four major North American professional sports leagues which are NBA, NHL, NFL and MLB.

Project Specifications

You are to get the raw NBA stats from a file, compute the efficiency of 3924 players, and compare and output the most efficient players, as well as some other interesting statistics. Your program is to take no other input (except for reading the file) and produce an internal database that can be queried to obtain many kinds of results.

The format of this file is easy to understand. The first line tells you the names of all columns. From the second line, each line's data corresponds to one player's regular season's statistics. Each field is separated by "|"

To understand the meanings of each of the abbreviations, look at the page:
http://www.databasebasketball.com/about/aboutstats.htm

Efficiency

How do many NBA coaches quickly evaluate a player's game performance? They check his efficiency. NBA.com evaluates all players based on the efficiency formula indicated below (and shown on the aboutstats.htm page). In this project, we will follow this efficiency formula. Since we are not evaluating a player based on one game, we need to divide the average efficiency by the number of games the player played. So the formula is:

$$Efficiency = [\ (\ pts + reb + asts + stl + blk) - (\ (\ fga - fgm) + (\ fta - ftm) + turnover\ )\ ] \ /\ gp$$

The technical words on the right side correspond to the fields in the statistics file.

Other Stats

Besides efficiency, also collect statistics like:

1. The player who played the most minutes
2. The player who played the most games
3. The player who scored the most points
4. The player who got the most rebounds
5. The player who got the most penalties

6. The player who made the most free throws

Similar to the query of the movie databases, there are multiple ways a query can be posed. In the movie search, you can either query an actor to find all the movies that he/she played in or you can query a movie to find all the actors in it. Similarly, you can query a player and find all his stats or you can query a stat and find all the top players in that particular stat category (accumulatively over the career of a player).

Deliverables

The deliverable for this assignment is the following file:

NBA.py – the source code for your Python program

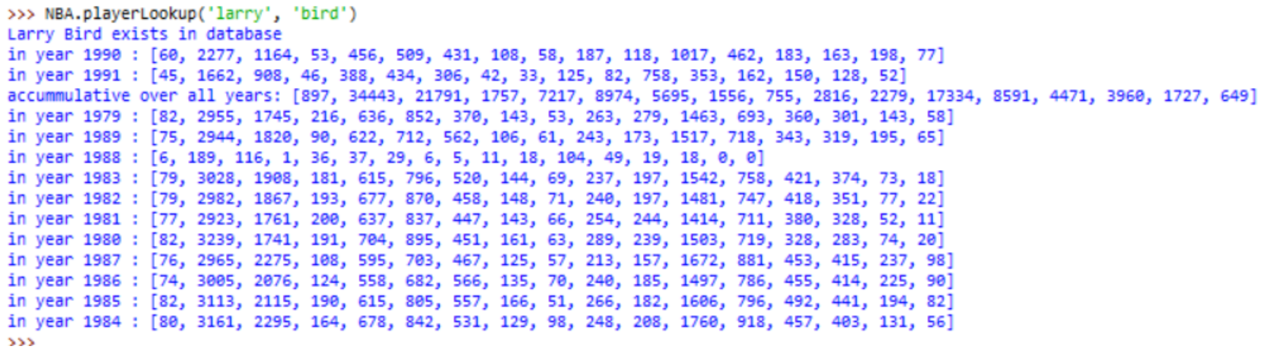
Be sure to use the specified file name and submit it for grading via the **turnin** system before the project deadline.

Assignment Notes:

1. When read the input file, you should be careful about the first line which does not contain any data. That line is important for you to tabulate and print the query results though.
2. Don’t forget to convert the string to number.
3. You cannot use a player’s name as key, as there are multiple players with the same names in NBA. Instead, use player ID as the key, but be warned that the ID must be normalized against random upper/lower case change.
4. Be warned that this database project is considerably more complex than either the sunspots or Apple stock ones. You need the following functions:
    a. A function “initDB (filename)” that takes one argument (the input file) and builds multiple dictionary structures (how many and what are they are up to you). You should also compute the efficiency of a player and add that as a field in your dictionary.
    b. A function “playerLookup (fname, lname)” that looks up a player with all his available stats tabulated yearly. Note that as the dictionary is not organized by names, but by ID, before you can get to a player’s stats you must call the function below
    c. A function “keyLookup(fname, lname)” that looks up a players unique ID. This function should not be called from outside your package, as user of your system should not know the player ID but only the player’s name. However, this call is important as the given player name might not be unique. If there are multiple players of the same names, this function should retrieve multiple IDs (keys) for you to index into the database.
    d. A function “statLookup(stat, numberItems) that looks up the player stats, sort them, and print out the top numberItems.
    e. A function “efficiency(numberItems) that looks up the most efficient numberItems players.

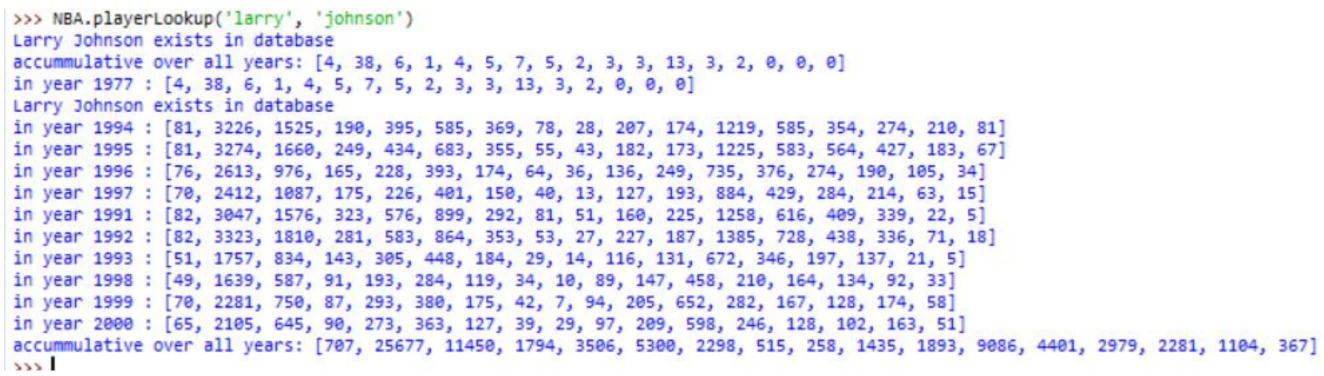
Sample Outputs:

If you look up a player “Larry Bird”, as the name in unique, you should get only one hit. You need to tabulate all available stats by year and add a line of average stats over the player’s career. Note that in the sample implementation, the player’s yearly record is stored in a dictionary, so the display may not put the accumulative record at the end.



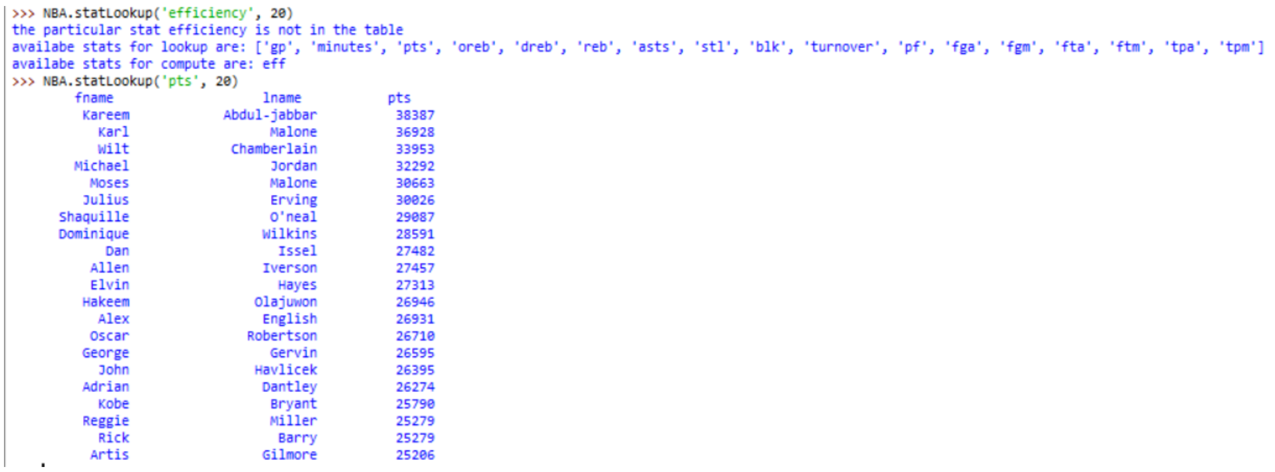
```
>>> NBA.playerLookup('larry', 'bird')
Larry Bird exists in database
in year 1990 : [60, 2277, 1164, 53, 456, 509, 431, 108, 58, 187, 118, 1017, 462, 183, 163, 198, 77]
in year 1991 : [45, 1662, 908, 46, 388, 434, 306, 42, 33, 125, 82, 758, 353, 162, 150, 128, 52]
accummulative over all years: [897, 34443, 21791, 1757, 7217, 8974, 5695, 1556, 755, 2816, 2279, 17334, 8591, 4471, 3960, 1727, 649]
in year 1979 : [82, 2955, 1745, 216, 636, 852, 370, 143, 53, 263, 279, 1463, 693, 360, 301, 143, 58]
in year 1989 : [75, 2944, 1820, 90, 622, 712, 562, 106, 61, 243, 173, 1517, 718, 343, 319, 195, 65]
in year 1988 : [6, 189, 116, 1, 36, 37, 29, 6, 5, 11, 18, 104, 49, 19, 18, 0, 0]
in year 1983 : [79, 3028, 1908, 181, 615, 796, 520, 144, 69, 237, 197, 1542, 758, 421, 374, 73, 18]
in year 1982 : [79, 2982, 1867, 193, 677, 870, 458, 148, 71, 240, 197, 1481, 747, 418, 351, 77, 22]
in year 1981 : [77, 2923, 1761, 200, 637, 837, 447, 143, 66, 254, 244, 1414, 711, 380, 328, 52, 11]
in year 1980 : [82, 3239, 1741, 191, 704, 895, 451, 161, 63, 289, 239, 1503, 719, 328, 283, 74, 20]
in year 1987 : [76, 2965, 2275, 108, 595, 703, 467, 125, 57, 213, 157, 1672, 881, 453, 415, 237, 98]
in year 1986 : [74, 3005, 2076, 124, 558, 682, 566, 135, 70, 240, 185, 1497, 786, 455, 414, 225, 90]
in year 1985 : [82, 3113, 2115, 190, 615, 805, 557, 166, 51, 266, 182, 1606, 796, 492, 441, 194, 82]
in year 1984 : [80, 3161, 2295, 164, 678, 842, 531, 129, 98, 248, 208, 1760, 918, 457, 403, 131, 56]
>>>
```

If you look up a player “Larry Johnson”, you should get two hits – as there are two NBA players with that name. So the keyLookup call should return two keys that allow you to retrieve records of both players and display them.



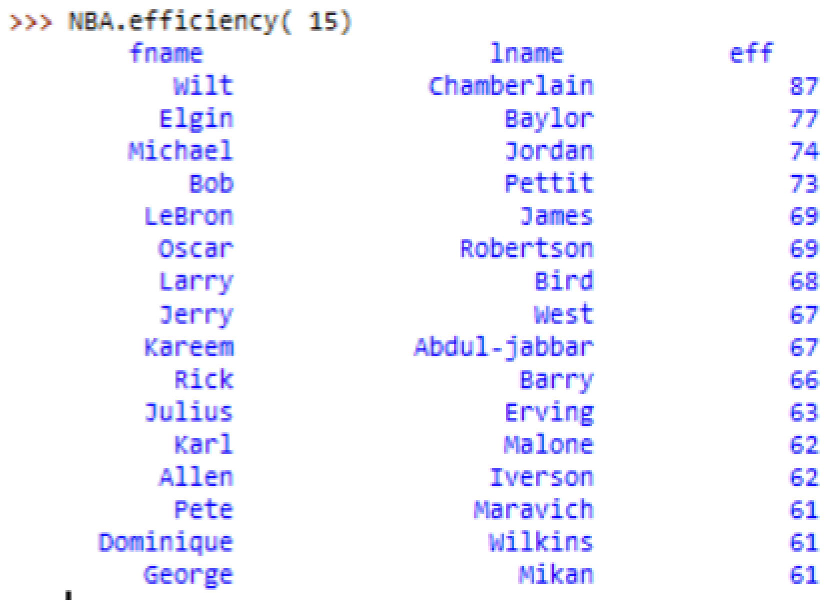
```
>>> NBA.playerLookup('larry', 'johnson')
Larry Johnson exists in database
accummulative over all years: [4, 38, 6, 1, 4, 5, 7, 5, 2, 3, 3, 13, 3, 2, 0, 0, 0]
in year 1977 : [4, 38, 6, 1, 4, 5, 7, 5, 2, 3, 3, 13, 3, 2, 0, 0, 0]
Larry Johnson exists in database
in year 1994 : [81, 3226, 1525, 190, 395, 585, 369, 78, 28, 207, 174, 1219, 585, 354, 274, 210, 81]
in year 1995 : [81, 3274, 1660, 249, 434, 683, 355, 55, 43, 182, 173, 1225, 583, 564, 427, 183, 67]
in year 1996 : [76, 2613, 976, 165, 228, 393, 174, 64, 36, 136, 249, 735, 376, 274, 190, 105, 34]
in year 1997 : [70, 2412, 1087, 175, 226, 401, 150, 40, 13, 127, 193, 884, 429, 284, 214, 63, 15]
in year 1991 : [82, 3047, 1576, 323, 576, 899, 292, 81, 51, 160, 225, 1258, 616, 409, 339, 22, 5]
in year 1992 : [82, 3323, 1810, 281, 583, 864, 353, 53, 27, 227, 187, 1385, 728, 438, 336, 71, 18]
in year 1993 : [51, 1757, 834, 143, 305, 448, 184, 29, 14, 116, 131, 672, 346, 197, 137, 21, 5]
in year 1998 : [49, 1639, 587, 91, 193, 284, 119, 34, 10, 89, 147, 458, 210, 164, 134, 92, 33]
in year 1999 : [70, 2281, 750, 87, 293, 380, 175, 42, 7, 94, 205, 652, 282, 167, 128, 174, 58]
in year 2000 : [65, 2105, 645, 90, 273, 363, 127, 39, 29, 97, 209, 598, 246, 128, 102, 163, 51]
accummulative over all years: [707, 25677, 11450, 1794, 3506, 5300, 2298, 515, 258, 1435, 1893, 9086, 4401, 2979, 2281, 1104, 367]
>>> |
```

The following shows stat look up for both non-existent and existing stats.



```
>>> NBA.statLookup('efficiency', 20)
the particular stat efficiency is not in the table
availabe stats for lookup are: ['gp', 'minutes', 'pts', 'oreb', 'dreb', 'reb', 'asts', 'stl', 'blk', 'turnover', 'pf', 'fga', 'fgm', 'fta', 'ftm', 'tpa', 'tpm']
availabe stats for compute are: eff
>>> NBA.statLookup('pts', 20)
       fname             lname          pts
      Kareem          Abdul-jabbar       38387
        Karl                Malone       36928
        Wilt           Chamberlain       33953
     Michael                Jordan       32292
       Moses                Malone       30663
      Julius                Erving       30026
   Shaquille                O'neal       29087
   Dominique               Wilkins       28591
         Dan                 Issel       27482
       Allen               Iverson       27457
       Elvin                 Hayes       27313
      Hakeem              Olajuwon       26946
        Alex               English       26931
       Oscar             Robertson       26710
      George                Gervin       26595
        John              Havlicek       26395
      Adrian               Dantley       26274
        Kobe                Bryant       25790
      Reggie                Miller       25279
        Rick                 Barry       25279
       Artis               Gilmore       25206
```

The following shows the top 15 efficient players using the stats calculated over their careers.



```
>>> NBA.efficiency( 15)
        fname                   lname           eff
           Wilt             Chamberlain             87
          Elgin                  Baylor             77
        Michael                  Jordan             74
            Bob                  Pettit             73
         LeBron                   James             69
          Oscar               Robertson             69
          Larry                    Bird             68
          Jerry                    West             67
         Kareem            Abdul-jabbar             67
           Rick                   Barry             66
         Julius                  Erving             63
           Karl                  Malone             62
          Allen                 Iverson             62
           Pete                Maravich             61
      Dominique                 Wilkins             61
         George                   Mikan             61
```