

```
In [2]: #Import numpy
import numpy as np

#Seasons
Seasons = ["2010", "2011", "2012", "2013", "2014", "2015", "2016", "2017", "2018", "2019"]
Sdict = {"2010":0, "2011":1, "2012":2, "2013":3, "2014":4, "2015":5, "2016":6, "2017":7, "2018":8, "2019":9}

#Players
Players = ["Sachin", "Rahul", "Smith", "Sami", "Pollard", "Morris", "Samson", "Dhoni", "Kohli", "Sky"]
Pdict = {"Sachin":0, "Rahul":1, "Smith":2, "Sami":3, "Pollard":4, "Morris":5, "Samson":6, "Dhoni":7, "Kohli":8, "Sky":9}

#Salaries
Sachin_Salary = [15946875, 17718750, 19490625, 21262500, 23034375, 24806250, 25244493, 278100000, 30518000, 33218000, 35918000, 38618000, 41318000, 44018000, 46718000, 49418000, 52118000]
Rahul_Salary = [12000000, 12744189, 13488377, 14232567, 14976754, 16324500, 18038573, 19750000, 21441800, 23141800, 24841800, 26541800, 28241800, 30041800, 31741800, 33441800, 35141800, 36841800]
Smith_Salary = [4621800, 5828090, 13041250, 14410581, 15779912, 14500000, 16022500, 1754500, 19061250, 20581250, 22101250, 23621250, 25141250, 26661250, 28181250, 29701250, 31221250, 32741250]
Sami_Salary = [3713640, 4694041, 13041250, 14410581, 15779912, 17149243, 18518574, 1945000, 20891770, 22321800, 23758000, 25202590, 26647180, 28091770, 295361800, 31022500, 32471800, 33921800, 35371800, 36821800, 38271800, 39721800, 41171200, 42621800, 44071800, 45521800, 46971800, 48421800, 49871800, 51321800, 52771800, 54221800, 55671800, 57121800, 58571800, 59921800, 61371800, 62821800, 64271800, 65721800, 67171800, 68621800, 69971800, 71421800, 72871800, 74321800, 75771800, 77221800, 78671800, 79121800, 80571800, 81021800, 82471800, 83921800, 85371800, 86821800, 88271800, 89721800, 91171800, 92621800, 94071800, 95521800, 96971800, 98421800, 99871800, 101321800, 102771800, 104221800, 105671800, 107121800, 108571800, 109921800, 111371800, 112821800, 114271800, 115721800, 117171800, 118621800, 119971800, 121421800, 122871800, 124321800, 125771800, 127221800, 128671800, 129121800, 130571800, 131921800, 133371800, 134821800, 136271800, 137721800, 139171800, 140621800, 142071800, 143521800, 144971800, 146421800, 147871800, 149321800, 150771800, 152221800, 153671800, 155121800, 156571800, 157921800, 159371800, 160821800, 162271800, 163721800, 165171800, 166621800, 168071800, 169521800, 171071800, 172521800, 174071800, 175521800, 177071800, 178521800, 179971800, 181421800, 182871800, 184321800, 185771800, 187221800, 188671800, 189121800, 190571800, 191921800, 193371800, 194821800, 196271800, 197721800, 199171800, 200621800, 202071800, 203521800, 204971800, 206421800, 207871800, 209321800, 210771800, 212221800, 213671800, 215121800, 216571800, 218021800, 219471800, 220921800, 222371800, 223821800, 225271800, 226721800, 228171800, 229621800, 231071800, 232521800, 233971800, 235421800, 236871800, 238321800, 239771800, 241221800, 242671800, 244121800, 245571800, 246921800, 248371800, 249821800, 251271800, 252721800, 254171800, 255621800, 257071800, 258521800, 259971800, 261421800, 262871800, 264321800, 265771800, 267221800, 268671800, 269121800, 270571800, 271921800, 273371800, 274821800, 276271800, 277721800, 279171800, 280621800, 282071800, 283521800, 284971800, 286421800, 287871800, 289321800, 290771800, 292221800, 293671800, 295121800, 296571800, 297921800, 299371800, 300821800, 302271800, 303721800, 305171800, 306621800, 308071800, 309521800, 310971800, 312421800, 313871800, 315321800, 316771800, 318221800, 319671800, 321121800, 322571800, 323921800, 325371800, 326821800, 328271800, 329721800, 331171800, 332621800, 334071800, 335521800, 336971800, 338421800, 339871800, 341321800, 342771800, 344221800, 345671800, 347121800, 348571800, 349921800, 351371800, 352821800, 354271800, 355721800, 357171800, 358621800, 359971800, 361421800, 362871800, 364321800, 365771800, 367221800, 368671800, 369121800, 370571800, 371921800, 373371800, 374821800, 376271800, 377721800, 379171800, 380621800, 382071800, 383521800, 384971800, 386421800, 387871800, 389321800, 390771800, 392221800, 393671800, 395121800, 396571800, 397921800, 399371800, 400821800, 402271800, 403721800, 405171800, 406621800, 408071800, 409521800, 410971800, 412421800, 413871800, 415321800, 416771800, 418221800, 419671800, 421121800, 422571800, 423921800, 425371800, 426821800, 428271800, 429721800, 431171800, 432621800, 434071800, 435521800, 436971800, 438421800, 439871800, 441321800, 442771800, 444221800, 445671800, 446121800, 447571800, 448921800, 449971800, 451421800, 452871800, 454321800, 455771800, 457221800, 458671800, 459971800, 461421800, 462871800, 464321800, 465771800, 467221800, 468671800, 469121800, 470571800, 471921800, 473371800, 474821800, 476271800, 477721800, 479171800, 480621800, 482071800, 483521800, 484971800, 486421800, 487871800, 489321800, 490771800, 492221800, 493671800, 495121800, 496571800, 497921800, 499371800, 500821800, 502271800, 503721800, 505171800, 506621800, 508071800, 509521800, 510971800, 512421800, 513871800, 515321800, 516771800, 518221800, 519671800, 521121800, 522571800, 523921800, 525371800, 526821800, 528271800, 529721800, 531171800, 532621800, 534071800, 535521800, 536971800, 538421800, 539871800, 541321800, 542771800, 544221800, 545671800, 546121800, 547571800, 548921800, 549971800, 551421800, 552871800, 554321800, 555771800, 557221800, 558671800, 559971800, 561421800, 562871800, 564321800, 565771800, 567221800, 568671800, 569121800, 570571800, 571921800, 573371800, 574821800, 576271800, 577721800, 579171800, 580621800, 582071800, 583521800, 584971800, 586421800, 587871800, 589321800, 590771800, 592221800, 593671800, 595121800, 596571800, 597921800, 599371800, 600821800, 602271800, 603721800, 605171800, 606621800, 608071800, 609521800, 610971800, 612421800, 613871800, 615321800, 616771800, 618221800, 619671800, 621121800, 622571800, 623921800, 625371800, 626821800, 628271800, 629721800, 631171800, 632621800, 634071800, 635521800, 636971800, 638421800, 639871800, 641321800, 642771800, 644221800, 645671800, 646121800, 647571800, 648921800, 649971800, 651421800, 652871800, 654321800, 655771800, 657221800, 658671800, 659971800, 661421800, 662871800, 664321800, 665771800, 667221800, 668671800, 669121800, 670571800, 671921800, 673371800, 674821800, 676271800, 677721800, 679171800, 680621800, 682071800, 683521800, 684971800, 686421800, 687871800, 689321800, 690771800, 692221800, 693671800, 695121800, 696571800, 697921800, 699371800, 700821800, 702271800, 703721800, 705171800, 706621800, 708071800, 709521800, 710971800, 712421800, 713871800, 715321800, 716771800, 718221800, 719671800, 721121800, 722571800, 723921800, 725371800, 726821800, 728271800, 729721800, 731171800, 732621800, 734071800, 735521800, 736971800, 738421800, 739871800, 741321800, 742771800, 744221800, 745671800, 746121800, 747571800, 748921800, 749971800, 751421800, 752871800, 754321800, 755771800, 757221800, 758671800, 759971800, 761421800, 762871800, 764321800, 765771800, 767221800, 768671800, 769121800, 770571800, 771921800, 773371800, 774821800, 776271800, 777721800, 779171800, 780621800, 782071800, 783521800, 784971800, 786421800, 787871800, 789321800, 790771800, 792221800, 793671800, 795121800, 796571800, 797921800, 799371800, 800821800, 802271800, 803721800, 805171800, 806621800, 808071800, 809521800, 810971800, 812421800, 813871800, 815321800, 816771800, 818221800, 819671800, 821121800, 822571800, 823921800, 825371800, 826821800, 828271800, 829721800, 831171800, 832621800, 834071800, 835521800, 836971800, 838421800, 839871800, 841321800, 842771800, 844221800, 845671800, 846121800, 847571800, 848921800, 849971800, 851421800, 852871800, 854321800, 855771800, 857221800, 858671800, 859971800, 861421800, 862871800, 864321800, 865771800, 867221800, 868671800, 869121800, 870571800, 871921800, 873371800, 874821800, 876271800, 877721800, 879171800, 880621800, 882071800, 883521800, 884971800, 886421800, 887871800, 889321800, 890771800, 892221800, 893671800, 895121800, 896571800, 897921800, 899371800, 900821800, 902271800, 903721800, 905171800, 906621800, 908071800, 909521800, 910971800, 912421800, 913871800, 915321800, 916771800, 918221800, 919671800, 921121800, 922571800, 923921800, 925371800, 926821800, 928271800, 929721800, 931171800, 932621800, 934071800, 935521800, 936971800, 938421800, 939871800, 941321800, 942771800, 944221800, 945671800, 946121800, 947571800, 948921800, 949971800, 951421800, 952871800, 954321800, 955771800, 957221800, 958671800, 959971800, 961421800, 962871800, 964321800, 965771800, 967221800, 968671800, 969121800, 970571800, 971921800, 973371800, 974821800, 976271800, 977721800, 979171800, 980621800, 982071800, 983521800, 984971800, 986421800, 987871800, 989321800, 990771800, 992221800, 993671800, 995121800, 996571800, 997921800, 999371800, 1000821800, 1002271800, 1003721800, 1005171800, 1006621800, 1008071800, 1009521800, 1010971800, 1012421800, 1013871800, 1015321800, 1016771800, 1018221800, 1019671800, 1021121800, 1022571800, 1023921800, 1025371800, 1026821800, 1028271800, 1029721800, 1031171800, 1032621800, 1034071800, 1035521800, 1036971800, 1038421800, 1039871800, 1041321800, 1042771800, 1044221800, 1045671800, 1046121800, 1047571800, 1048921800, 1049971800, 1051421800, 1052871800, 1054321800, 1055771800, 1057221800, 1058671800, 1059971800, 1061421800, 1062871800, 1064321800, 1065771800, 1067221800, 1068671800, 1069121800, 1070571800, 1071921800, 1073371800, 1074821800, 1076271800, 1077721800, 1079171800, 1080621800, 1082071800, 1083521800, 1084971800, 1086421800, 1087871800, 1089321800, 1090771800, 1092221800, 1093671800, 1095121800, 1096571800, 1097921800, 1099371800, 1100821800, 1102271800, 1103721800, 1105171800, 1106621800, 1108071800, 1109521800, 1110971800, 1112421800, 1113871800, 1115321800, 1116771800, 1118221800, 1119671800, 1121121800, 1122571800, 1123921800, 1125371800, 1126821800, 1128271800, 1129721800, 1131171800, 1132621800, 1134071800, 1135521800, 1136971800, 1138421800, 1139871800, 1141321800, 1142771800, 1144221800, 1145671800, 1146121800, 1147571800, 1148921800, 1149971800, 1151421800, 1152871800, 1154321800, 1155771800, 1157221800, 1158671800, 1159971800, 1161421800, 1162871800, 1164321800, 1165771800, 1167221800, 1168671800, 1169121800, 1170571800, 1171921800, 1173371800, 1174821800, 1176271800, 1177721800, 1179171800, 1180621800, 1182071800, 1183521800, 1184971800, 1186421800, 1187871800, 1189321800, 1190771800, 1192221800, 1193671800, 1195121800, 1196571800, 1197921800, 1199371800, 1200821800, 1202271800, 1203721800, 1205171800, 1206621800, 1208071800, 1209521800, 1210971800, 1212421800, 1213871800, 1215321800, 1216771800, 1218221800, 1219671800, 1221121800, 1222571800, 1223921800, 1225371800, 1226821800, 1228271800, 1229721800, 1231171800, 1232621800, 1234071800, 1235521800, 1236971800, 1238421800, 1239871800, 1241321800, 1242771800, 1244221800, 1245671800, 1246121800, 1247571800, 1248921800, 1249971800, 1251421800, 1252871800, 1254321800, 1255771800, 1257221800, 1258671800, 1259971800, 1261421800, 1262871800, 1264321800, 1265771800, 1267221800, 1268671800, 1269121800, 1270571800, 1271921800, 1273371800, 1274821800, 1276271800, 1277721800, 1279171800, 1280621800, 1282071800, 1283521800, 1284971800, 1286421800, 1287871800, 1289321800, 1290771800, 1292221800, 1293671800, 1295121800, 1296571800, 1297921800, 1299371800, 1300821800, 1302271800, 1303721800, 1305171800, 1306621800, 1308071800, 1309521800, 1310971800, 1312421800, 1313871800, 1315321800, 1316771800, 1318221800, 1319671800, 1321121800, 1322571800, 1323921800, 1325371800, 1326821800, 1328271800, 1329721800, 1331171800,
```

In [5]: `Salary # martrix format`

```
Out[5]: array([[15946875, 17718750, 19490625, 21262500, 23034375, 24806250,
   25244493, 27849149, 30453805, 23500000],
   [12000000, 12744189, 13488377, 14232567, 14976754, 16324500,
   18038573, 19752645, 21466718, 23180790],
   [ 4621800, 5828090, 13041250, 14410581, 15779912, 14500000,
   16022500, 17545000, 19067500, 20644400],
   [ 3713640, 4694041, 13041250, 14410581, 15779912, 17149243,
   18518574, 19450000, 22407474, 22458000],
   [ 4493160, 4806720, 6061274, 13758000, 15202590, 16647180,
   18091770, 19536360, 20513178, 21436271],
   [ 3348000, 4235220, 12455000, 14410581, 15779912, 14500000,
   16022500, 17545000, 19067500, 20644400],
   [ 3144240, 3380160, 3615960, 4574189, 13520500, 14940153,
   16359805, 17779458, 18668431, 20068563],
   [ 0, 0, 4171200, 4484040, 4796880, 6053663,
   15506632, 16669630, 17832627, 18995624],
   [ 0, 0, 0, 4822800, 5184480, 5546160,
   6993708, 16402500, 17632688, 18862875],
   [ 3031920, 3841443, 13041250, 14410581, 15779912, 14200000,
   15691000, 17182000, 18673000, 15000000]])
```

In [6]: `Games`

```
Out[6]: array([[80, 77, 82, 82, 73, 82, 58, 78, 6, 35],
   [82, 57, 82, 79, 76, 72, 60, 72, 79, 80],
   [79, 78, 75, 81, 76, 79, 62, 76, 77, 69],
   [80, 65, 77, 66, 69, 77, 55, 67, 77, 40],
   [82, 82, 79, 82, 78, 54, 76, 71, 41],
   [70, 69, 67, 77, 70, 77, 57, 74, 79, 44],
   [78, 64, 80, 78, 45, 80, 60, 70, 62, 82],
   [35, 35, 80, 74, 82, 78, 66, 81, 81, 27],
   [40, 40, 40, 81, 78, 81, 39, 0, 10, 51],
   [75, 51, 51, 79, 77, 76, 49, 69, 54, 62]])
```

In [7]: `Points`

```
Out[7]: array([[2832, 2430, 2323, 2201, 1970, 2078, 1616, 2133, 83, 782],
   [1653, 1426, 1779, 1688, 1619, 1312, 1129, 1170, 1245, 1154],
   [2478, 2132, 2250, 2304, 2258, 2111, 1683, 2036, 2089, 1743],
   [2122, 1881, 1978, 1504, 1943, 1970, 1245, 1920, 2112, 966],
   [1292, 1443, 1695, 1624, 1503, 1784, 1113, 1296, 1297, 646],
   [1572, 1561, 1496, 1746, 1678, 1438, 1025, 1232, 1281, 928],
   [1258, 1104, 1684, 1781, 841, 1268, 1189, 1186, 1185, 1564],
   [ 903, 903, 1624, 1871, 2472, 2161, 1850, 2280, 2593, 686],
   [ 597, 597, 597, 1361, 1619, 2026, 852, 0, 159, 904],
   [2040, 1397, 1254, 2386, 2045, 1941, 1082, 1463, 1028, 1331]])
```

In [8]: `mydata = np.arange(0,20)`
`print(mydata)`

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
```

In [9]: `np.reshape(mydata,(4,5)) # 5 rows & 4 columns`

```
Out[9]: array([[ 0,  1,  2,  3,  4],
   [ 5,  6,  7,  8,  9],
   [10, 11, 12, 13, 14],
   [15, 16, 17, 18, 19]])
```

```
In [10]: mydata
```

```
Out[10]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
   17, 18, 19])
```

```
In [11]: #np.reshape(mydata,(5,4), order = 'c') #'C' means to read / write the elements using
MATR1 = np.reshape(mydata, (5,4), order = 'c')
MATR1
```

```
Out[11]: array([[ 0,  1,  2,  3],
   [ 4,  5,  6,  7],
   [ 8,  9, 10, 11],
   [12, 13, 14, 15],
   [16, 17, 18, 19]])
```

```
In [12]: MATR1
```

```
Out[12]: array([[ 0,  1,  2,  3],
   [ 4,  5,  6,  7],
   [ 8,  9, 10, 11],
   [12, 13, 14, 15],
   [16, 17, 18, 19]])
```

```
In [13]: MATR1[4,3]
```

```
Out[13]: 19
```

```
In [14]: MATR1[3,3]
```

```
Out[14]: 15
```

```
In [15]: MATR1
```

```
Out[15]: array([[ 0,  1,  2,  3],
   [ 4,  5,  6,  7],
   [ 8,  9, 10, 11],
   [12, 13, 14, 15],
   [16, 17, 18, 19]])
```

```
In [16]: MATR1[-3,-1]
```

```
Out[16]: 11
```

```
In [17]: MATR1
```

```
Out[17]: array([[ 0,  1,  2,  3],
   [ 4,  5,  6,  7],
   [ 8,  9, 10, 11],
   [12, 13, 14, 15],
   [16, 17, 18, 19]])
```

```
In [18]: mydata
```

```
Out[18]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
   17, 18, 19])
```

```
In [19]: MATR2 = np.reshape(mydata, (5,4), order = 'F') # reshape behaviour are - 'C', 'F', '
MATR2
```

```
Out[19]: array([[ 0,  5, 10, 15],
   [ 1,  6, 11, 16],
   [ 2,  7, 12, 17],
   [ 3,  8, 13, 18],
   [ 4,  9, 14, 19]])
```

```
In [20]: MATR2[4,3]
```

```
Out[20]: 19
```

```
In [21]: MATR2[0,2]
```

```
Out[21]: 10
```

```
In [23]: MATR2[0:2]
```

```
Out[23]: array([[ 0,  5, 10, 15],
   [ 1,  6, 11, 16]])
```

```
In [25]: MATR2
```

```
Out[25]: array([[ 0,  5, 10, 15],
   [ 1,  6, 11, 16],
   [ 2,  7, 12, 17],
   [ 3,  8, 13, 18],
   [ 4,  9, 14, 19]])
```

```
In [26]: MATR2[1:2]
```

```
Out[26]: array([[ 1,  6, 11, 16]])
```

```
In [27]: MATR2[1,2]
```

```
Out[27]: 11
```

```
In [28]: MATR2
```

```
Out[28]: array([[ 0,  5, 10, 15],
   [ 1,  6, 11, 16],
   [ 2,  7, 12, 17],
   [ 3,  8, 13, 18],
   [ 4,  9, 14, 19]])
```

```
In [29]: MATR2[-2,-1]
```

```
Out[29]: 18
```

```
In [30]: MATR2[-3,-3]
```

```
Out[30]: 7
```

```
In [31]: MATR2
```

```
Out[31]: array([[ 0,  5, 10, 15],
   [ 1,  6, 11, 16],
   [ 2,  7, 12, 17],
   [ 3,  8, 13, 18],
   [ 4,  9, 14, 19]])
```

```
In [32]: MATR2[0:2]
```

```
Out[32]: array([[ 0,  5, 10, 15],
   [ 1,  6, 11, 16]])
```

```
In [33]: mydata
```

```
Out[33]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
   17, 18, 19])
```

```
In [34]: MATR3 = np.reshape(mydata, (5,4), order = 'A')
MATR3
```

```
Out[34]: array([[ 0,  1,  2,  3],
   [ 4,  5,  6,  7],
   [ 8,  9, 10, 11],
   [12, 13, 14, 15],
   [16, 17, 18, 19]])
```

```
In [35]: MATR2
```

```
Out[35]: array([[ 0,  5, 10, 15],
   [ 1,  6, 11, 16],
   [ 2,  7, 12, 17],
   [ 3,  8, 13, 18],
   [ 4,  9, 14, 19]])
```

```
In [36]: MATR1
```

```
Out[36]: array([[ 0,  1,  2,  3],
   [ 4,  5,  6,  7],
   [ 8,  9, 10, 11],
   [12, 13, 14, 15],
   [16, 17, 18, 19]])
```

```
In [37]: a1 = ['welcome', 'to','datascience']
a2 = ['required','hard','work' ]
a3 = [1,2,3]
```

```
In [38]: [a1,a2,a3]
```

```
Out[38]: [['welcome', 'to', 'datascience'], ['required', 'hard', 'work'], [1, 2, 3]]
```

```
In [39]: np.array([a1,a2,a3])
```

```
Out[39]: array([[ 'welcome', 'to', 'datascience'],
   ['required', 'hard', 'work'],
   ['1', '2', '3']], dtype='<U11')
```

```
In [40]: Games
```

```
Out[40]: array([[80, 77, 82, 82, 73, 82, 58, 78, 6, 35],
   [82, 57, 82, 79, 76, 72, 60, 72, 79, 80],
   [79, 78, 75, 81, 76, 79, 62, 76, 77, 69],
   [80, 65, 77, 66, 69, 77, 55, 67, 77, 40],
   [82, 82, 82, 79, 82, 78, 54, 76, 71, 41],
   [70, 69, 67, 77, 70, 77, 57, 74, 79, 44],
   [78, 64, 80, 78, 45, 80, 60, 70, 62, 82],
   [35, 35, 80, 74, 82, 78, 66, 81, 81, 27],
   [40, 40, 40, 81, 78, 81, 39, 0, 10, 51],
   [75, 51, 51, 79, 77, 76, 49, 69, 54, 62]])
```

```
In [41]: Games[0]
```

```
Out[41]: array([80, 77, 82, 82, 73, 82, 58, 78, 6, 35])
```

```
In [42]: Games[7]
```

```
Out[42]: array([35, 35, 80, 74, 82, 78, 66, 81, 81, 27])
```

```
In [44]: Games[0:8]
```

```
Out[44]: array([[80, 77, 82, 82, 73, 82, 58, 78, 6, 35],
   [82, 57, 82, 79, 76, 72, 60, 72, 79, 80],
   [79, 78, 75, 81, 76, 79, 62, 76, 77, 69],
   [80, 65, 77, 66, 69, 77, 55, 67, 77, 40],
   [82, 82, 82, 79, 82, 78, 54, 76, 71, 41],
   [70, 69, 67, 77, 70, 77, 57, 74, 79, 44],
   [78, 64, 80, 78, 45, 80, 60, 70, 62, 82],
   [35, 35, 80, 74, 82, 78, 66, 81, 81, 27]])
```

```
In [46]: Games[0:8]
```

```
Out[46]: 6
```

```
In [47]: Games
```

```
Out[47]: array([[80, 77, 82, 82, 73, 82, 58, 78, 6, 35],
   [82, 57, 82, 79, 76, 72, 60, 72, 79, 80],
   [79, 78, 75, 81, 76, 79, 62, 76, 77, 69],
   [80, 65, 77, 66, 69, 77, 55, 67, 77, 40],
   [82, 82, 82, 79, 82, 78, 54, 76, 71, 41],
   [70, 69, 67, 77, 70, 77, 57, 74, 79, 44],
   [78, 64, 80, 78, 45, 80, 60, 70, 62, 82],
   [35, 35, 80, 74, 82, 78, 66, 81, 81, 27],
   [40, 40, 40, 81, 78, 81, 39, 0, 10, 51],
   [75, 51, 51, 79, 77, 76, 49, 69, 54, 62]])
```

In [48]: Games[0:2]

Out[48]: array([[80, 77, 82, 82, 73, 82, 58, 78, 6, 35],
[82, 57, 82, 79, 76, 72, 60, 72, 79, 80]])

In [49]: Games

Out[49]: array([[80, 77, 82, 82, 73, 82, 58, 78, 6, 35],
[82, 57, 82, 79, 76, 72, 60, 72, 79, 80],
[79, 78, 75, 81, 76, 79, 62, 76, 77, 69],
[80, 65, 77, 66, 69, 77, 55, 67, 77, 40],
[82, 82, 82, 79, 82, 78, 54, 76, 71, 41],
[70, 69, 67, 77, 70, 77, 57, 74, 79, 44],
[78, 64, 80, 78, 45, 80, 60, 70, 62, 82],
[35, 35, 80, 74, 82, 78, 66, 81, 81, 27],
[40, 40, 40, 81, 78, 81, 39, 0, 10, 51],
[75, 51, 51, 79, 77, 76, 49, 69, 54, 62]])

In [50]: Games[1:2]

Out[50]: array([[82, 57, 82, 79, 76, 72, 60, 72, 79, 80]])

In [51]: Games[4]

Out[51]: array([82, 82, 82, 79, 82, 78, 54, 76, 71, 41])

In [52]: Games

Out[52]: array([[80, 77, 82, 82, 73, 82, 58, 78, 6, 35],
[82, 57, 82, 79, 76, 72, 60, 72, 79, 80],
[79, 78, 75, 81, 76, 79, 62, 76, 77, 69],
[80, 65, 77, 66, 69, 77, 55, 67, 77, 40],
[82, 82, 82, 79, 82, 78, 54, 76, 71, 41],
[70, 69, 67, 77, 70, 77, 57, 74, 79, 44],
[78, 64, 80, 78, 45, 80, 60, 70, 62, 82],
[35, 35, 80, 74, 82, 78, 66, 81, 81, 27],
[40, 40, 40, 81, 78, 81, 39, 0, 10, 51],
[75, 51, 51, 79, 77, 76, 49, 69, 54, 62]])

In [53]: Games[2,6]

Out[53]: 62

In [54]: Games

```
Out[54]: array([[80, 77, 82, 82, 73, 82, 58, 78, 6, 35],  
                 [82, 57, 82, 79, 76, 72, 60, 72, 79, 80],  
                 [79, 78, 75, 81, 76, 79, 62, 76, 77, 69],  
                 [80, 65, 77, 66, 69, 77, 55, 67, 77, 40],  
                 [82, 82, 82, 79, 82, 78, 54, 76, 71, 41],  
                 [70, 69, 67, 77, 70, 77, 57, 74, 79, 44],  
                 [78, 64, 80, 78, 45, 80, 60, 70, 62, 82],  
                 [35, 35, 80, 74, 82, 78, 66, 81, 81, 27],  
                 [40, 40, 40, 81, 78, 81, 39, 0, 10, 51],  
                 [75, 51, 51, 79, 77, 76, 49, 69, 54, 62]])
```

```
In [55]: Games[-3:-1]
```

```
Out[55]: array([[35, 35, 80, 74, 82, 78, 66, 81, 81, 27],  
                 [40, 40, 40, 81, 78, 81, 39, 0, 10, 51]])
```

```
In [56]: Games[-3,-1]
```

```
Out[56]: 27
```

```
In [57]: Points
```

```
Out[57]: array([[2832, 2430, 2323, 2201, 1970, 2078, 1616, 2133, 83, 782],  
                 [1653, 1426, 1779, 1688, 1619, 1312, 1129, 1170, 1245, 1154],  
                 [2478, 2132, 2250, 2304, 2258, 2111, 1683, 2036, 2089, 1743],  
                 [2122, 1881, 1978, 1504, 1943, 1970, 1245, 1920, 2112, 966],  
                 [1292, 1443, 1695, 1624, 1503, 1784, 1113, 1296, 1297, 646],  
                 [1572, 1561, 1496, 1746, 1678, 1438, 1025, 1232, 1281, 928],  
                 [1258, 1104, 1684, 1781, 841, 1268, 1189, 1186, 1185, 1564],  
                 [903, 903, 1624, 1871, 2472, 2161, 1850, 2280, 2593, 686],  
                 [597, 597, 597, 1361, 1619, 2026, 852, 0, 159, 904],  
                 [2040, 1397, 1254, 2386, 2045, 1941, 1082, 1463, 1028, 1331]])
```

```
In [58]: Points[0]
```

```
Out[58]: array([2832, 2430, 2323, 2201, 1970, 2078, 1616, 2133, 83, 782])
```

```
In [59]: Points
```

```
Out[59]: array([[2832, 2430, 2323, 2201, 1970, 2078, 1616, 2133, 83, 782],  
                 [1653, 1426, 1779, 1688, 1619, 1312, 1129, 1170, 1245, 1154],  
                 [2478, 2132, 2250, 2304, 2258, 2111, 1683, 2036, 2089, 1743],  
                 [2122, 1881, 1978, 1504, 1943, 1970, 1245, 1920, 2112, 966],  
                 [1292, 1443, 1695, 1624, 1503, 1784, 1113, 1296, 1297, 646],  
                 [1572, 1561, 1496, 1746, 1678, 1438, 1025, 1232, 1281, 928],  
                 [1258, 1104, 1684, 1781, 841, 1268, 1189, 1186, 1185, 1564],  
                 [903, 903, 1624, 1871, 2472, 2161, 1850, 2280, 2593, 686],  
                 [597, 597, 597, 1361, 1619, 2026, 852, 0, 159, 904],  
                 [2040, 1397, 1254, 2386, 2045, 1941, 1082, 1463, 1028, 1331]])
```

```
In [60]: Points[-6,-1]
```

```
Out[60]: 646
```

```
In [61]: ===== DICTIONARY =====  
  
# dict does not maintain the order  
  
dict1 = {'key1': 'val1', 'key2': 'val2', 'key3': 'val3'}
```

```
In [62]: dict1
```

```
Out[62]: {'key1': 'val1', 'key2': 'val2', 'key3': 'val3'}
```

```
In [63]: dict1['key2']
```

```
Out[63]: 'val2'
```

```
In [64]: dict2 = {'bang': 2, 'hyd': 'we are hear', 'pune': True}
```

```
In [65]: dict2
```

```
Out[65]: {'bang': 2, 'hyd': 'we are hear', 'pune': True}
```

```
In [66]: dict3 = {'Germany': 'I have been here', 'France': 2, 'Spain': True}
```

```
In [67]: dict3
```

```
Out[67]: {'Germany': 'I have been here', 'France': 2, 'Spain': True}
```

```
In [68]: dict3
```

```
Out[68]: {'Germany': 'I have been here', 'France': 2, 'Spain': True}
```

```
In [69]: dict3['Germany']
```

```
Out[69]: 'I have been here'
```

```
In [70]: Games
```

```
Out[71]: array([[80, 77, 82, 82, 73, 82, 58, 78, 6, 35],  
                 [82, 57, 82, 79, 76, 72, 60, 72, 79, 80],  
                 [79, 78, 75, 81, 76, 79, 62, 76, 77, 69],  
                 [80, 65, 77, 66, 69, 77, 55, 67, 77, 40],  
                 [82, 82, 82, 79, 82, 78, 54, 76, 71, 41],  
                 [70, 69, 67, 77, 70, 77, 57, 74, 79, 44],  
                 [78, 64, 80, 78, 45, 80, 60, 70, 62, 82],  
                 [35, 35, 80, 74, 82, 78, 66, 81, 81, 27],  
                 [40, 40, 40, 81, 78, 81, 39, 0, 10, 51],  
                 [75, 51, 51, 79, 77, 76, 49, 69, 54, 62]])
```

```
In [72]: Pdict
```

```
Out[72]: {'Sachin': 0,  
          'Rahul': 1,  
          'Smith': 2,  
          'Sami': 3,  
          'Pollard': 4,  
          'Morris': 5,  
          'Samson': 6,  
          'Dhoni': 7,  
          'Kohli': 8,  
          'Sky': 9}
```

```
In [73]: Pdict['Sachin']
```

```
Out[73]: 0
```

```
In [74]: Games[0]
```

```
Out[74]: array([80, 77, 82, 82, 73, 82, 58, 78, 6, 35])
```

```
In [75]: Games
```

```
Out[75]: array([[80, 77, 82, 82, 73, 82, 58, 78, 6, 35],  
                 [82, 57, 82, 79, 76, 72, 60, 72, 79, 80],  
                 [79, 78, 75, 81, 76, 79, 62, 76, 77, 69],  
                 [80, 65, 77, 66, 69, 77, 55, 67, 77, 40],  
                 [82, 82, 82, 79, 82, 78, 54, 76, 71, 41],  
                 [70, 69, 67, 77, 70, 77, 57, 74, 79, 44],  
                 [78, 64, 80, 78, 45, 80, 60, 70, 62, 82],  
                 [35, 35, 80, 74, 82, 78, 66, 81, 81, 27],  
                 [40, 40, 40, 81, 78, 81, 39, 0, 10, 51],  
                 [75, 51, 51, 79, 77, 76, 49, 69, 54, 62]])
```

```
In [76]: Pdict['Rahul']
```

```
Out[76]: 1
```

```
In [77]: Games[1]
```

```
Out[77]: array([82, 57, 82, 79, 76, 72, 60, 72, 79, 80])
```

Games

```
In [78]: Games[Pdict['Rahul']]
```

```
Out[78]: array([82, 57, 82, 79, 76, 72, 60, 72, 79, 80])
```

```
In [79]: Points
```

```
Out[79]: array([[2832, 2430, 2323, 2201, 1970, 2078, 1616, 2133, 83, 782],  
[1653, 1426, 1779, 1688, 1619, 1312, 1129, 1170, 1245, 1154],  
[2478, 2132, 2250, 2304, 2258, 2111, 1683, 2036, 2089, 1743],  
[2122, 1881, 1978, 1504, 1943, 1970, 1245, 1920, 2112, 966],  
[1292, 1443, 1695, 1624, 1503, 1784, 1113, 1296, 1297, 646],  
[1572, 1561, 1496, 1746, 1678, 1438, 1025, 1232, 1281, 928],  
[1258, 1104, 1684, 1781, 841, 1268, 1189, 1186, 1185, 1564],  
[ 903,  903, 1624, 1871, 2472, 2161, 1850, 2280, 2593, 686],  
[ 597,  597,  597, 1361, 1619, 2026,  852,    0, 159, 904],  
[2040, 1397, 1254, 2386, 2045, 1941, 1082, 1463, 1028, 1331]])
```

```
In [80]: Salary
```

```
Out[80]: array([[15946875, 17718750, 19490625, 21262500, 23034375, 24806250,  
25244493, 27849149, 30453805, 23500000],  
[12000000, 12744189, 13488377, 14232567, 14976754, 16324500,  
18038573, 19752645, 21466718, 23180790],  
[ 4621800, 5828090, 13041250, 14410581, 15779912, 14500000,  
16022500, 17545000, 19067500, 20644400],  
[ 3713640, 4694041, 13041250, 14410581, 15779912, 17149243,  
18518574, 19450000, 22407474, 22458000],  
[ 4493160, 4806720, 6061274, 13758000, 15202590, 16647180,  
18091770, 19536360, 20513178, 21436271],  
[ 3348000, 4235220, 12455000, 14410581, 15779912, 14500000,  
16022500, 17545000, 19067500, 20644400],  
[ 3144240, 3380160, 3615960, 4574189, 13520500, 14940153,  
16359805, 17779458, 18668431, 20068563],  
[      0,      0, 4171200, 4484040, 4796880, 6053663,  
15506632, 16669630, 17832627, 18995624],  
[      0,      0,      0, 4822800, 5184480, 5546160,  
6993708, 16402500, 17632688, 18862875],  
[ 3031920, 3841443, 13041250, 14410581, 15779912, 14200000,  
15691000, 17182000, 18673000, 15000000]])
```

```
In [81]: Salary[3,6]
```

```
Out[81]: 18518574
```

```
In [82]: Salary
```

```
Out[82]: array([[15946875, 17718750, 19490625, 21262500, 23034375, 24806250,
   25244493, 27849149, 30453805, 23500000],
   [12000000, 12744189, 13488377, 14232567, 14976754, 16324500,
   18038573, 19752645, 21466718, 23180790],
   [ 4621800, 5828090, 13041250, 14410581, 15779912, 14500000,
   16022500, 17545000, 19067500, 20644400],
   [ 3713640, 4694041, 13041250, 14410581, 15779912, 17149243,
   18518574, 19450000, 22407474, 22458000],
   [ 4493160, 4806720, 6061274, 13758000, 15202590, 16647180,
   18091770, 19536360, 20513178, 21436271],
   [ 3348000, 4235220, 12455000, 14410581, 15779912, 14500000,
   16022500, 17545000, 19067500, 20644400],
   [ 3144240, 3380160, 3615960, 4574189, 13520500, 14940153,
   16359805, 17779458, 18668431, 20068563],
   [ 0, 0, 4171200, 4484040, 4796880, 6053663,
   15506632, 16669630, 17832627, 18995624],
   [ 0, 0, 4822800, 5184480, 5546160,
   6993708, 16402500, 17632688, 18862875],
   [ 3031920, 3841443, 13041250, 14410581, 15779912, 14200000,
   15691000, 17182000, 18673000, 15000000]])
```

```
In [84]: Salary[Pdict['Sky']][Sdict['2019']]
```

```
Out[84]: 15000000
```

```
In [85]: Salary
```

```
Out[85]: array([[15946875, 17718750, 19490625, 21262500, 23034375, 24806250,
   25244493, 27849149, 30453805, 23500000],
   [12000000, 12744189, 13488377, 14232567, 14976754, 16324500,
   18038573, 19752645, 21466718, 23180790],
   [ 4621800, 5828090, 13041250, 14410581, 15779912, 14500000,
   16022500, 17545000, 19067500, 20644400],
   [ 3713640, 4694041, 13041250, 14410581, 15779912, 17149243,
   18518574, 19450000, 22407474, 22458000],
   [ 4493160, 4806720, 6061274, 13758000, 15202590, 16647180,
   18091770, 19536360, 20513178, 21436271],
   [ 3348000, 4235220, 12455000, 14410581, 15779912, 14500000,
   16022500, 17545000, 19067500, 20644400],
   [ 3144240, 3380160, 3615960, 4574189, 13520500, 14940153,
   16359805, 17779458, 18668431, 20068563],
   [ 0, 0, 4171200, 4484040, 4796880, 6053663,
   15506632, 16669630, 17832627, 18995624],
   [ 0, 0, 4822800, 5184480, 5546160,
   6993708, 16402500, 17632688, 18862875],
   [ 3031920, 3841443, 13041250, 14410581, 15779912, 14200000,
   15691000, 17182000, 18673000, 15000000]])
```

```
In [87]: Games
```

```
Out[87]: array([[80, 77, 82, 82, 73, 82, 58, 78, 6, 35],  
   [82, 57, 82, 79, 76, 72, 60, 72, 79, 80],  
   [79, 78, 75, 81, 76, 79, 62, 76, 77, 69],  
   [80, 65, 77, 66, 69, 77, 55, 67, 77, 40],  
   [82, 82, 82, 79, 82, 78, 54, 76, 71, 41],  
   [70, 69, 67, 77, 70, 77, 57, 74, 79, 44],  
   [78, 64, 80, 78, 45, 80, 60, 70, 62, 82],  
   [35, 35, 80, 74, 82, 78, 66, 81, 81, 27],  
   [40, 40, 40, 81, 78, 81, 39, 0, 10, 51],  
   [75, 51, 51, 79, 77, 76, 49, 69, 54, 62]])
```

```
In [88]: Salary/Games
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_2544\3709746658.py:1: RuntimeWarning: divide by zero encountered in divide  
Salary/Games
```

```
Out[88]: array([[ 199335.9375 ,  230113.63636364,  237690.54878049,
   259298.7804878 ,  315539.38356164,  302515.24390244,
   435249.87931034,  357040.37179487,  5075634.16666667,
   671428.57142857],
 [ 146341.46341463,  223582.26315789,  164492.40243902,
  180159.07594937,  197062.55263158,  226729.16666667,
  300642.88333333,  274342.29166667,  271730.60759494,
  289759.875     ],
 [ 58503.79746835,  74719.1025641 ,  173883.33333333,
  177908.40740741,  207630.42105263,  183544.30379747,
  258427.41935484,  230855.26315789,  247629.87012987,
  299194.20289855],
 [ 46420.5       ,  72216.01538462,  169366.88311688,
  218342.13636364,  228694.37681159,  222717.44155844,
  336701.34545455,  290298.50746269,  291006.15584416,
  561450.      ],
 [ 54794.63414634,  58618.53658537,  73917.97560976,
  174151.89873418,  185397.43902439,  213425.38461538,
  335032.77777778,  257057.36842105,  288918.      ,
  522835.87804878],
 [ 47828.57142857,  61380.        ,  185895.52238806,
  187150.4025974 ,  225427.31428571,  188311.68831169,
  281096.49122807,  237094.59459459,  241360.75949367,
  469190.90909091],
 [ 40310.76923077,  52815.        ,  45199.5       ,
  58643.44871795,  300455.55555556,  186751.9125       ,
  272663.41666667,  253992.25714286,  301103.72580645,
  244738.57317073],
 [ 0.        ,  0.        ,  52140.        ,
  60595.13513514,  58498.53658537,  77611.06410256,
  234948.96969697,  205797.90123457,  220155.88888889,
  703541.62962963],
 [ 0.        ,  0.        ,  0.        ,
  59540.74074074,  66467.69230769,  68471.11111111,
  179325.84615385,  inf,  1763268.8       ,
  369860.29411765],
 [ 40425.6       ,  75322.41176471,  255710.78431373,
  182412.41772152,  204933.92207792,  186842.10526316,
  320224.48979592,  249014.49275362,  345796.2962963 ,
  241935.48387097]])
```

```
In [89]: np.round(Salary/Games)
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_2544\3232172828.py:1: RuntimeWarning: divide by zero encountered in divide
np.round(Salary/Games)
```

```
Out[89]: array([[ 199336.,  230114.,  237691.,  259299.,  315539.,  302515.,
   435250.,  357040.,  5075634.,  671429.],
   [ 146341.,  223582.,  164492.,  180159.,  197063.,  226729.],
   [ 300643.,  274342.,  271731.,  289760.],
   [ 58504.,  74719.,  173883.,  177908.,  207630.,  183544.],
   [ 258427.,  230855.,  247630.,  299194.],
   [ 46420.,  72216.,  169367.,  218342.,  228694.,  222717.],
   [ 336701.,  290299.,  291006.,  561450.],
   [ 54795.,  58619.,  73918.,  174152.,  185397.,  213425.],
   [ 335033.,  257057.,  288918.,  522836.],
   [ 47829.,  61380.,  185896.,  187150.,  225427.,  188312.],
   [ 281096.,  237095.,  241361.,  469191.],
   [ 40311.,  52815.,  45200.,  58643.,  300456.,  186752.],
   [ 272663.,  253992.,  301104.,  244739.],
   [ 0.,  0.,  52140.,  60595.,  58499.,  77611.],
   [ 234949.,  205798.,  220156.,  703542.],
   [ 0.,  0.,  0.,  59541.,  66468.,  68471.],
   [ 179326.,  inf,  1763269.,  369860.],
   [ 40426.,  75322.,  255711.,  182412.,  204934.,  186842.],
   [ 320224.,  249014.,  345796.,  241935.]])
```

```
In [90]: import warnings
warnings.filterwarnings('ignore')
#np.round(FieldGoals/Games)
#FieldGoals/Games # this matrix is lot of decimal points yo can not round
#round()
```

```
In [91]: ## --- First visualization ----##
```

```
In [92]: import numpy as np
import matplotlib.pyplot as plt
```

```
In [93]: %matplotlib inline # keep the plot inside jupyter nots insted of getting in other s
UsageError: unrecognized arguments: # keep the plot inside jupyter nots insted of ge
tting in other screen
```

```
In [94]: Salary
```

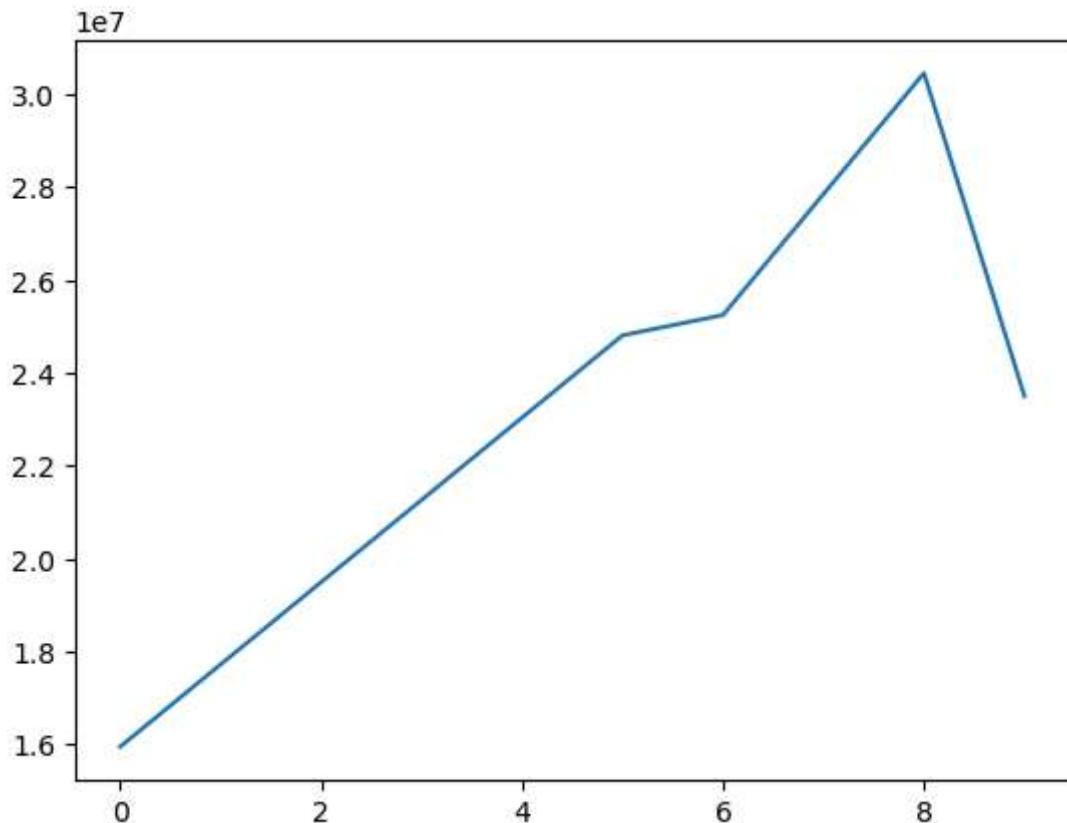
```
Out[94]: array([[15946875, 17718750, 19490625, 21262500, 23034375, 24806250,
   25244493, 27849149, 30453805, 23500000],
   [12000000, 12744189, 13488377, 14232567, 14976754, 16324500,
   18038573, 19752645, 21466718, 23180790],
   [ 4621800, 5828090, 13041250, 14410581, 15779912, 14500000,
   16022500, 17545000, 19067500, 20644400],
   [ 3713640, 4694041, 13041250, 14410581, 15779912, 17149243,
   18518574, 19450000, 22407474, 22458000],
   [ 4493160, 4806720, 6061274, 13758000, 15202590, 16647180,
   18091770, 19536360, 20513178, 21436271],
   [ 3348000, 4235220, 12455000, 14410581, 15779912, 14500000,
   16022500, 17545000, 19067500, 20644400],
   [ 3144240, 3380160, 3615960, 4574189, 13520500, 14940153,
   16359805, 17779458, 18668431, 20068563],
   [ 0, 0, 4171200, 4484040, 4796880, 6053663,
   15506632, 16669630, 17832627, 18995624],
   [ 0, 0, 0, 4822800, 5184480, 5546160,
   6993708, 16402500, 17632688, 18862875],
   [ 3031920, 3841443, 13041250, 14410581, 15779912, 14200000,
   15691000, 17182000, 18673000, 15000000]])
```

```
In [96]: Salary[0]
```

```
Out[96]: array([15946875, 17718750, 19490625, 21262500, 23034375, 24806250,
   25244493, 27849149, 30453805, 23500000])
```

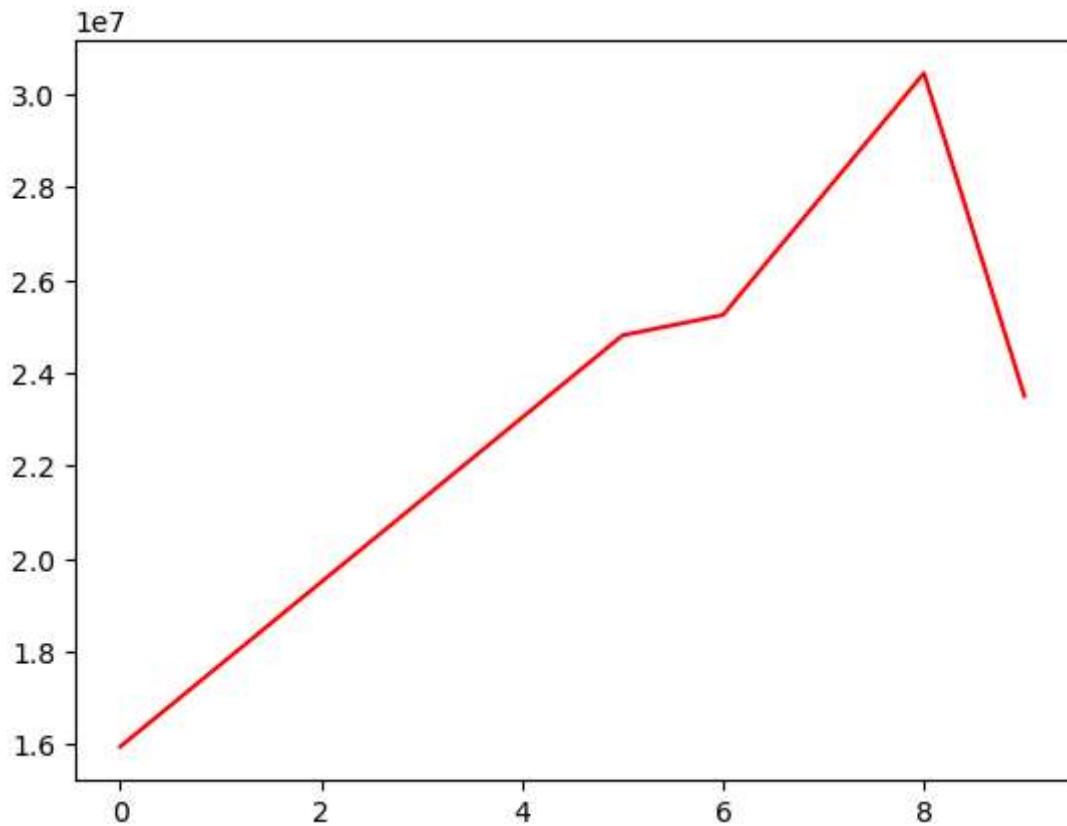
```
In [97]: plt.plot(Salary[0])
```

```
Out[97]: [<matplotlib.lines.Line2D at 0x23c718d5c10>]
```



```
In [98]: plt.plot(Salary[0], c='red')
```

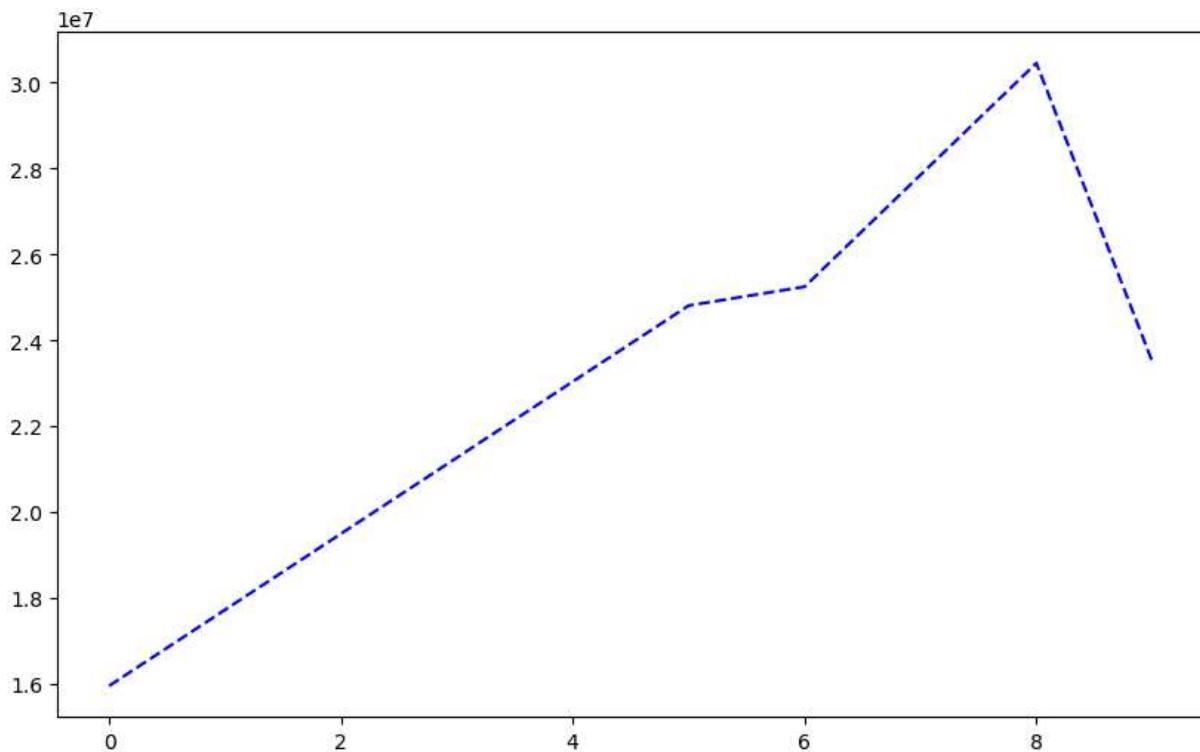
```
Out[98]: [<matplotlib.lines.Line2D at 0x23c71a0a750>]
```



```
In [99]: %matplotlib inline  
plt.rcParams['figure.figsize'] = 10,6
```

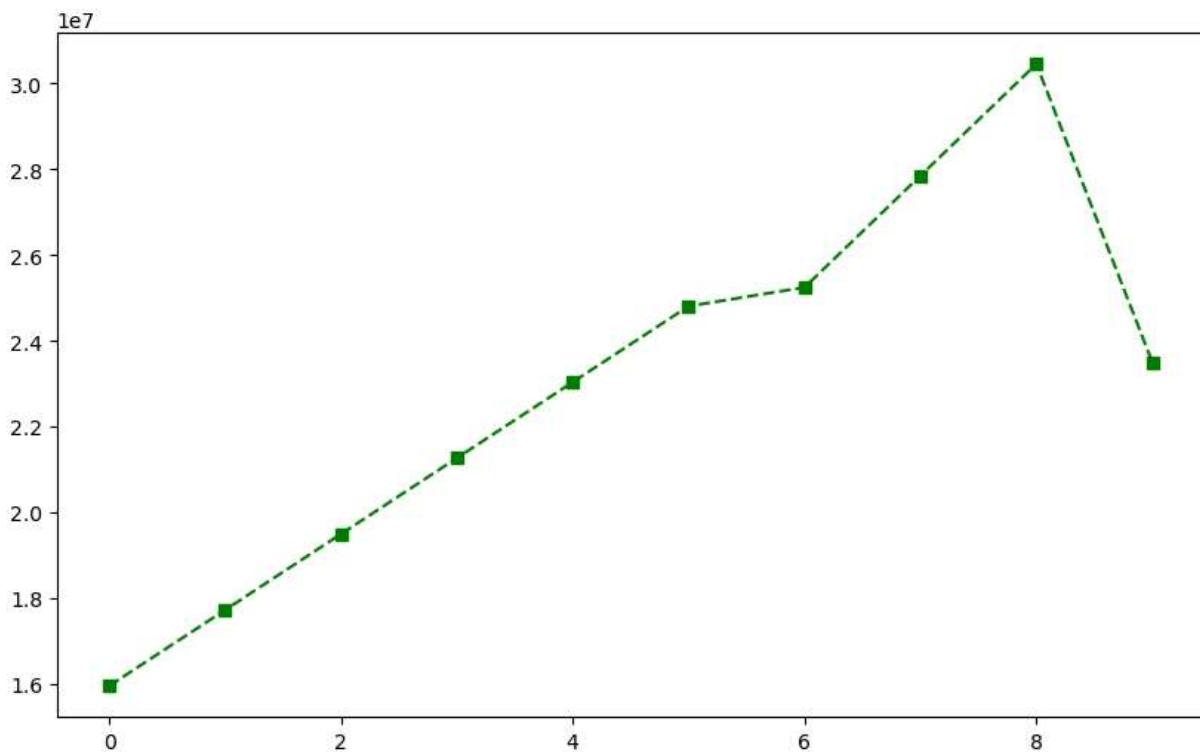
```
In [100... plt.plot(Salary[0], c='Blue', ls = 'dashed')
```

```
Out[100... [<matplotlib.lines.Line2D at 0x23c71a90920>]
```



```
In [101... plt.plot(Salary[0], c='Green', ls = '--', marker = 's') # s - squares
```

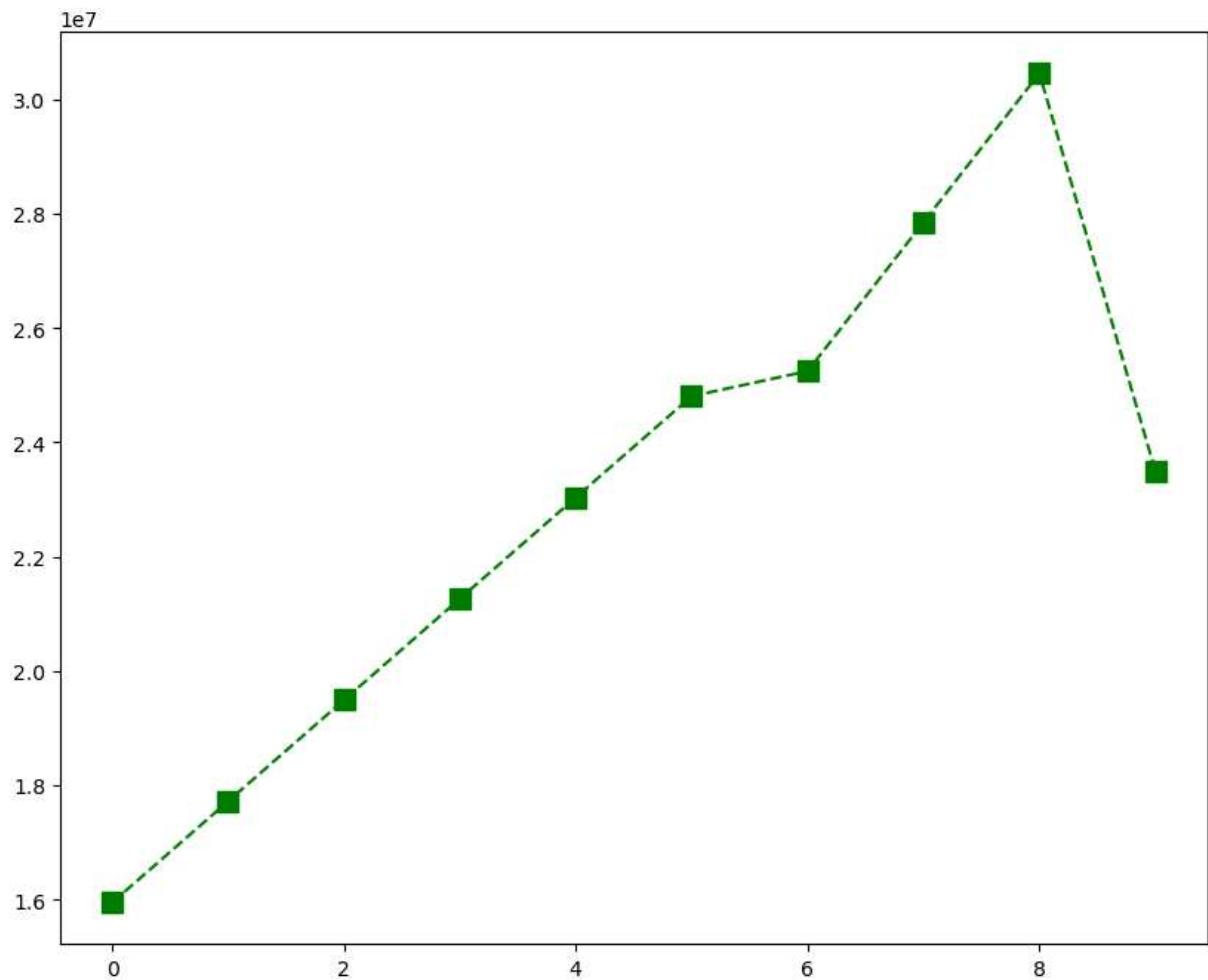
```
Out[101... <matplotlib.lines.Line2D at 0x23c71af2240>]
```



```
In [102... %matplotlib inline  
plt.rcParams['figure.figsize'] = 10,8 #runtime configuration parameter
```

```
In [103... plt.plot(Salary[0], c='Green', ls = '--', marker = 's', ms = 10)
```

```
plt.show()
```



```
In [104...]: list(range(0,10))
```

```
Out[104...]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

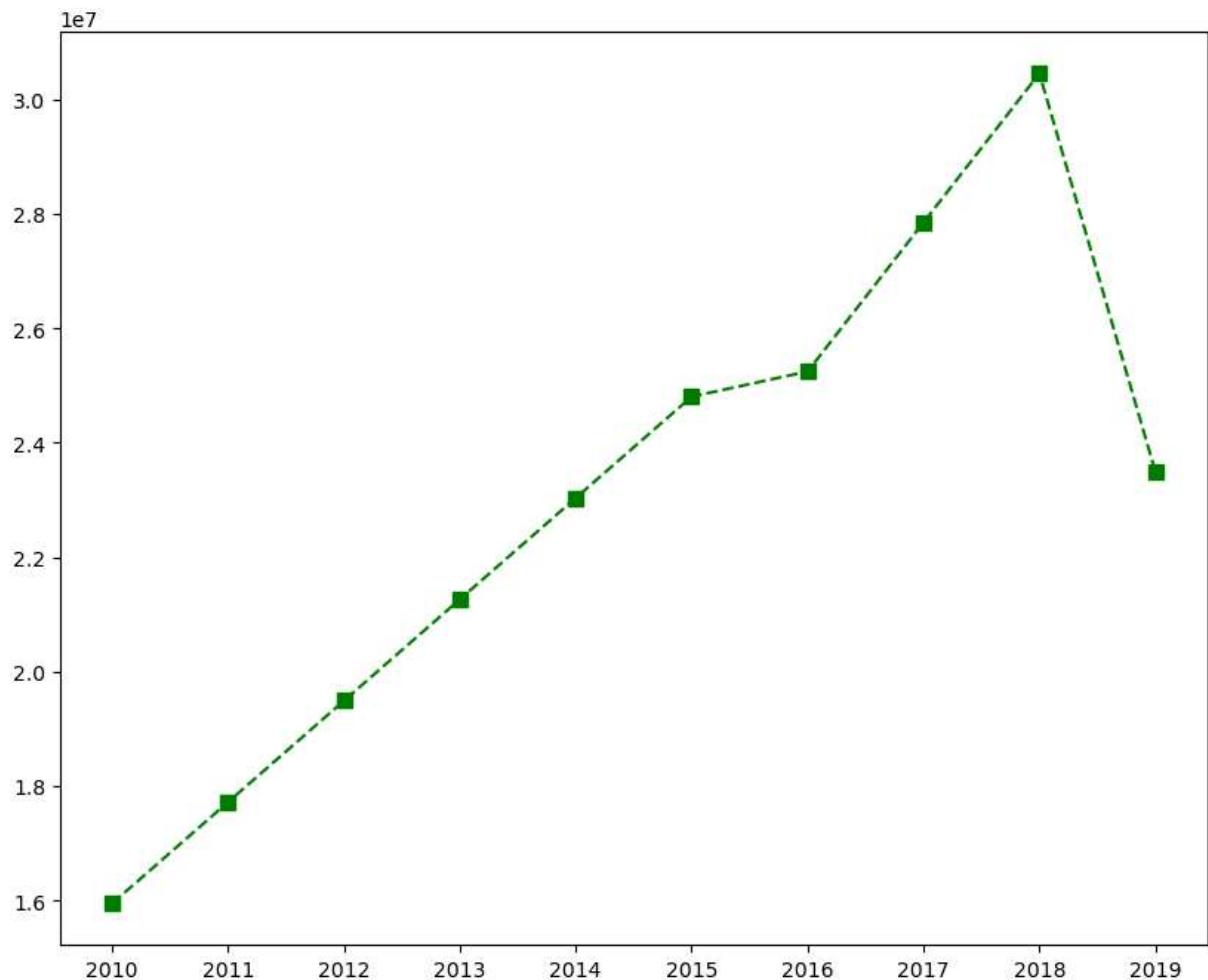
```
In [105...]: Sdict
```

```
Out[105...]: {'2010': 0,
 '2011': 1,
 '2012': 2,
 '2013': 3,
 '2014': 4,
 '2015': 5,
 '2016': 6,
 '2017': 7,
 '2018': 8,
 '2019': 9}
```

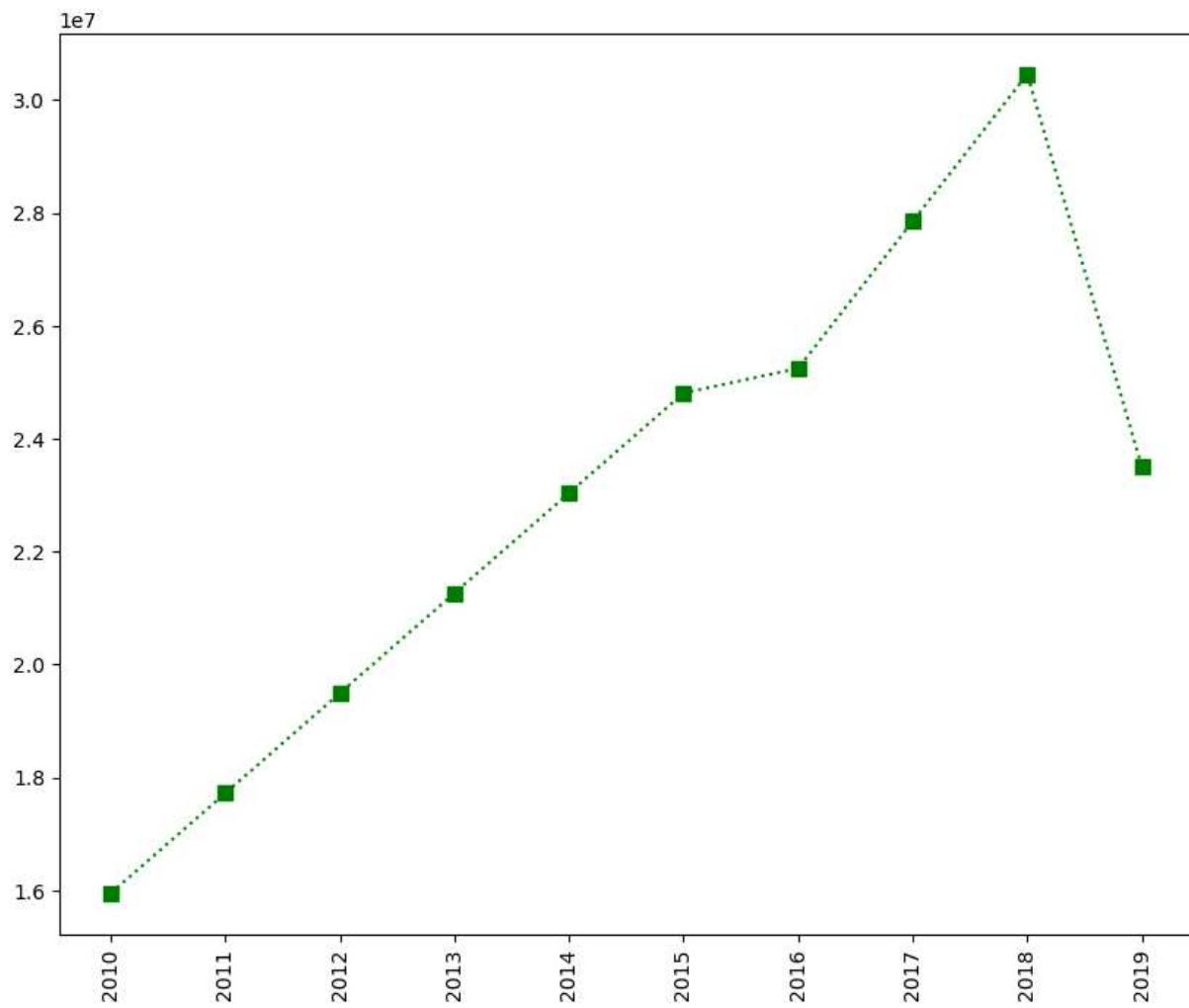
```
In [106...]: Pdict
```

```
Out[106... {'Sachin': 0,
 'Rahul': 1,
 'Smith': 2,
 'Sami': 3,
 'Pollard': 4,
 'Morris': 5,
 'Samson': 6,
 'Dhoni': 7,
 'Kohli': 8,
 'Sky': 9}
```

```
In [107... plt.plot(Salary[0], c='Green', ls = '--', marker = 's', ms = 7)
plt.xticks(list(range(0,10)), Seasons)
plt.show()
```



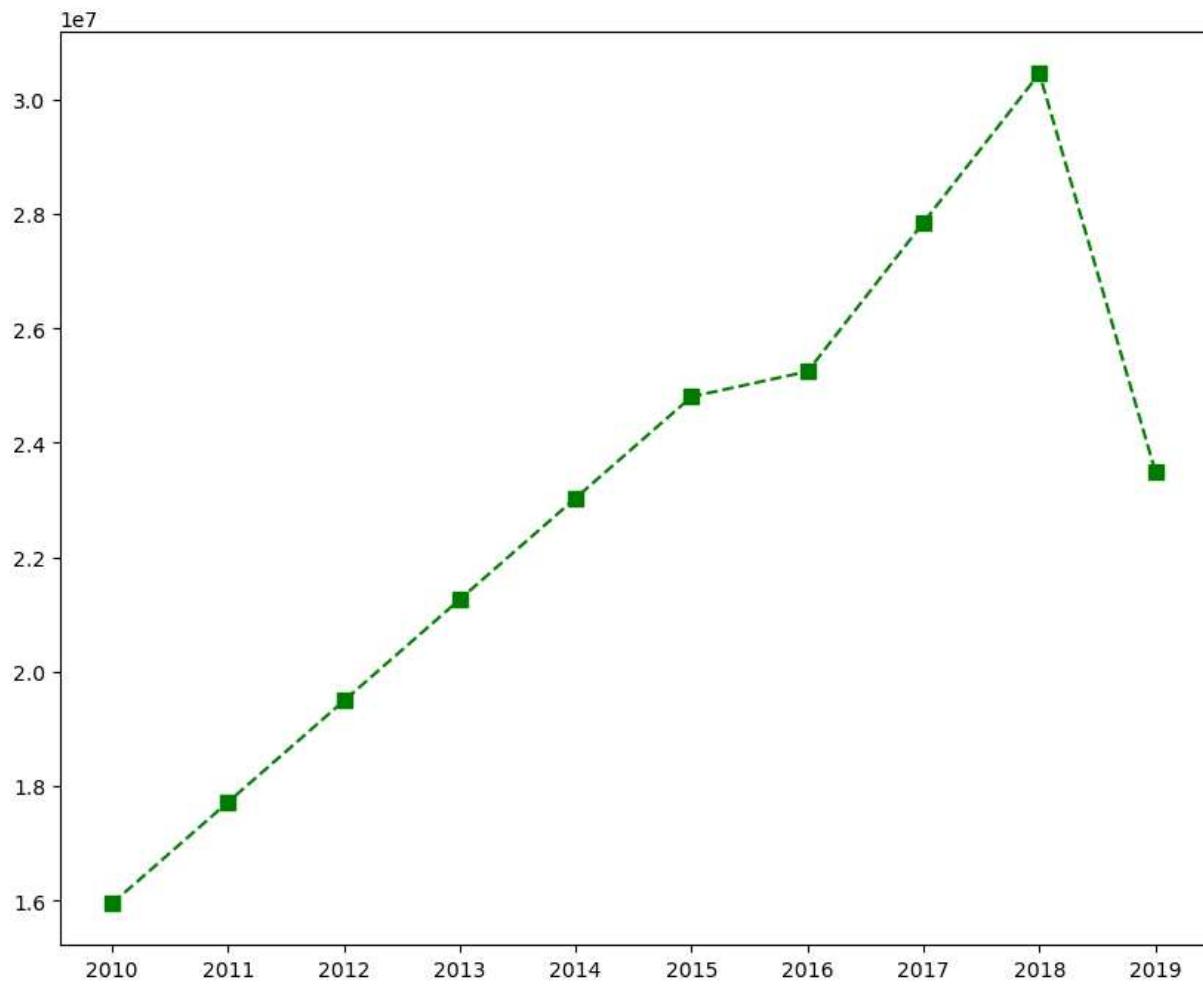
```
In [108... plt.plot(Salary[0], c='Green', ls = ':', marker = 's', ms = 7, label = Players[0])
plt.xticks(list(range(0,10)), Seasons, rotation='vertical')
plt.show()
```



```
In [109...]: Games
```

```
Out[109...]: array([[80, 77, 82, 82, 73, 82, 58, 78, 6, 35],
       [82, 57, 82, 79, 76, 72, 60, 72, 79, 80],
       [79, 78, 75, 81, 76, 79, 62, 76, 77, 69],
       [80, 65, 77, 66, 69, 77, 55, 67, 77, 40],
       [82, 82, 82, 79, 82, 78, 54, 76, 71, 41],
       [70, 69, 67, 77, 70, 77, 57, 74, 79, 44],
       [78, 64, 80, 78, 45, 80, 60, 70, 62, 82],
       [35, 35, 80, 74, 82, 78, 66, 81, 81, 27],
       [40, 40, 40, 81, 78, 81, 39, 0, 10, 51],
       [75, 51, 51, 79, 77, 76, 49, 69, 54, 62]])
```

```
In [110...]: plt.plot(Salary[0], c='Green', ls = '--', marker = 's', ms = 7, label = Players[0])
plt.xticks(list(range(0,10)), Seasons, rotation='horizontal')
plt.show()
```



```
In [111... Salary[0]
```

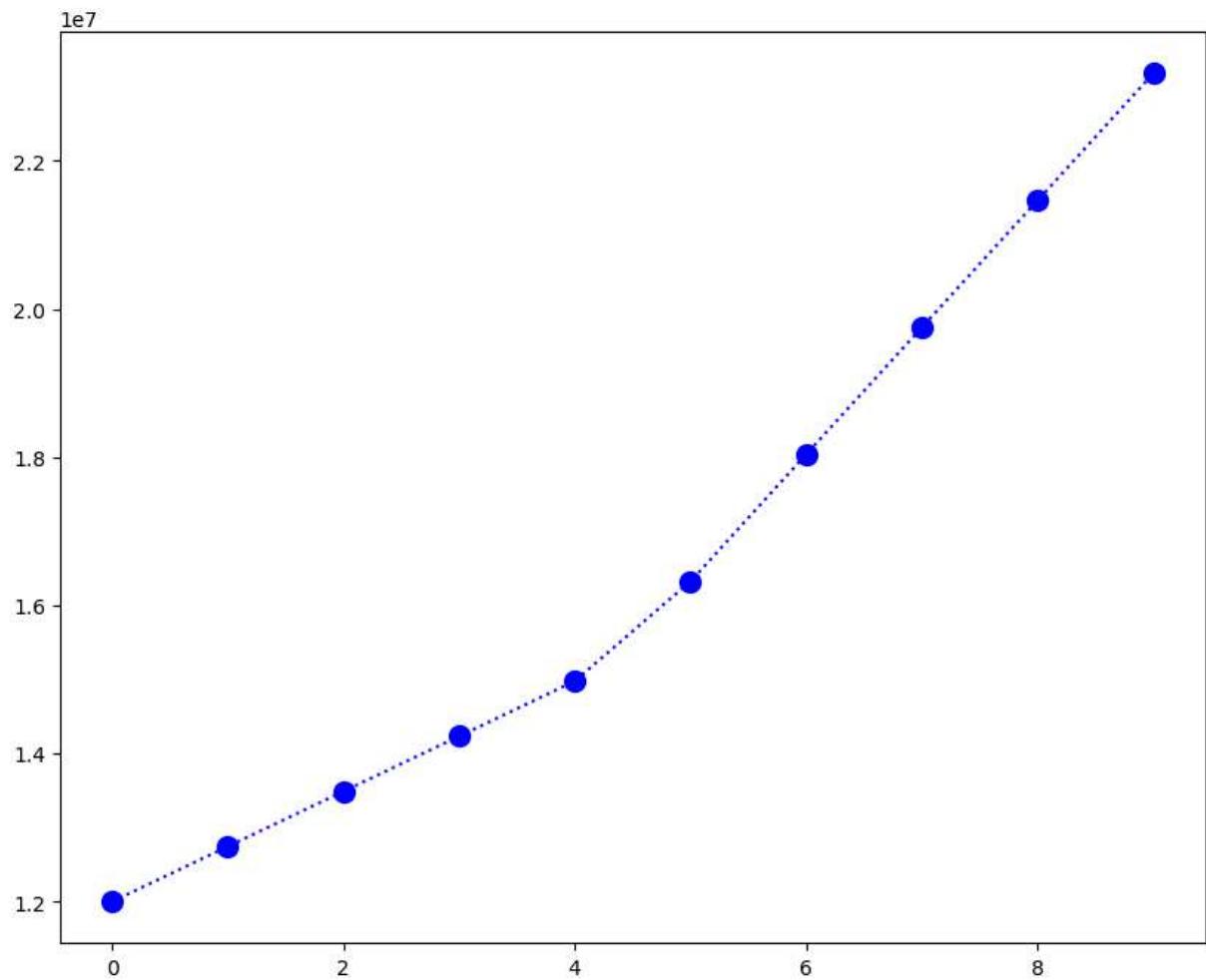
```
Out[111... array([15946875, 17718750, 19490625, 21262500, 23034375, 24806250,
   25244493, 27849149, 30453805, 23500000])
```

```
In [112... Salary[1]
```

```
Out[112... array([12000000, 12744189, 13488377, 14232567, 14976754, 16324500,
   18038573, 19752645, 21466718, 23180790])
```

```
In [113... plt.plot(Salary[1], c='Blue', ls = ':', marker = 'o', ms = 10, label = Players[1])
```

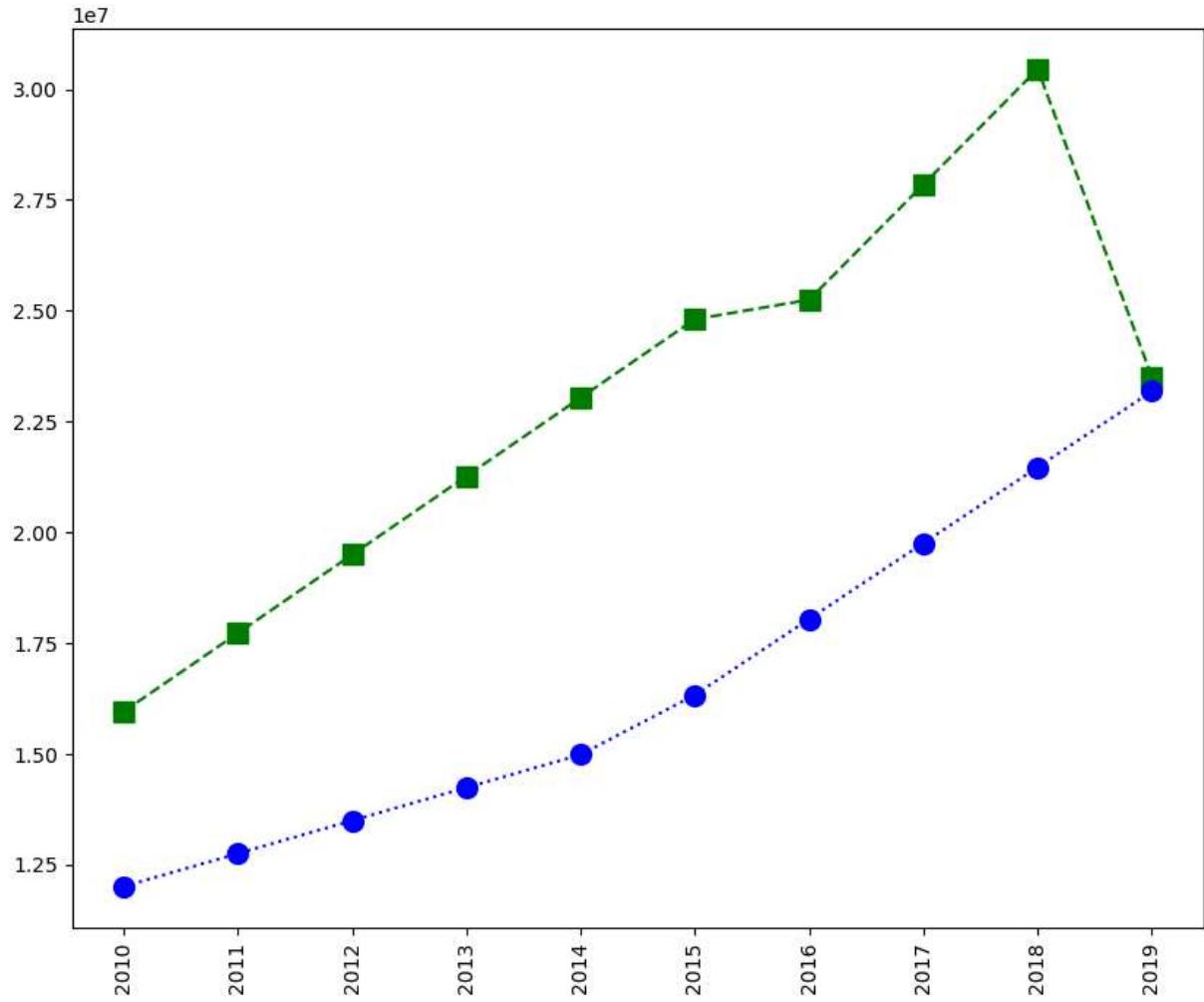
```
Out[113... [<matplotlib.lines.Line2D at 0x23c73c18da0>]
```



```
In [114]: plt.plot(Salary[0], c='Green', ls = '--', marker = 's', ms = 10, label = Players[0])
plt.plot(Salary[1], c='Blue', ls = ':', marker = 'o', ms = 10, label = Players[1])

plt.xticks(list(range(0,10)), Seasons, rotation='vertical')

plt.show()
```

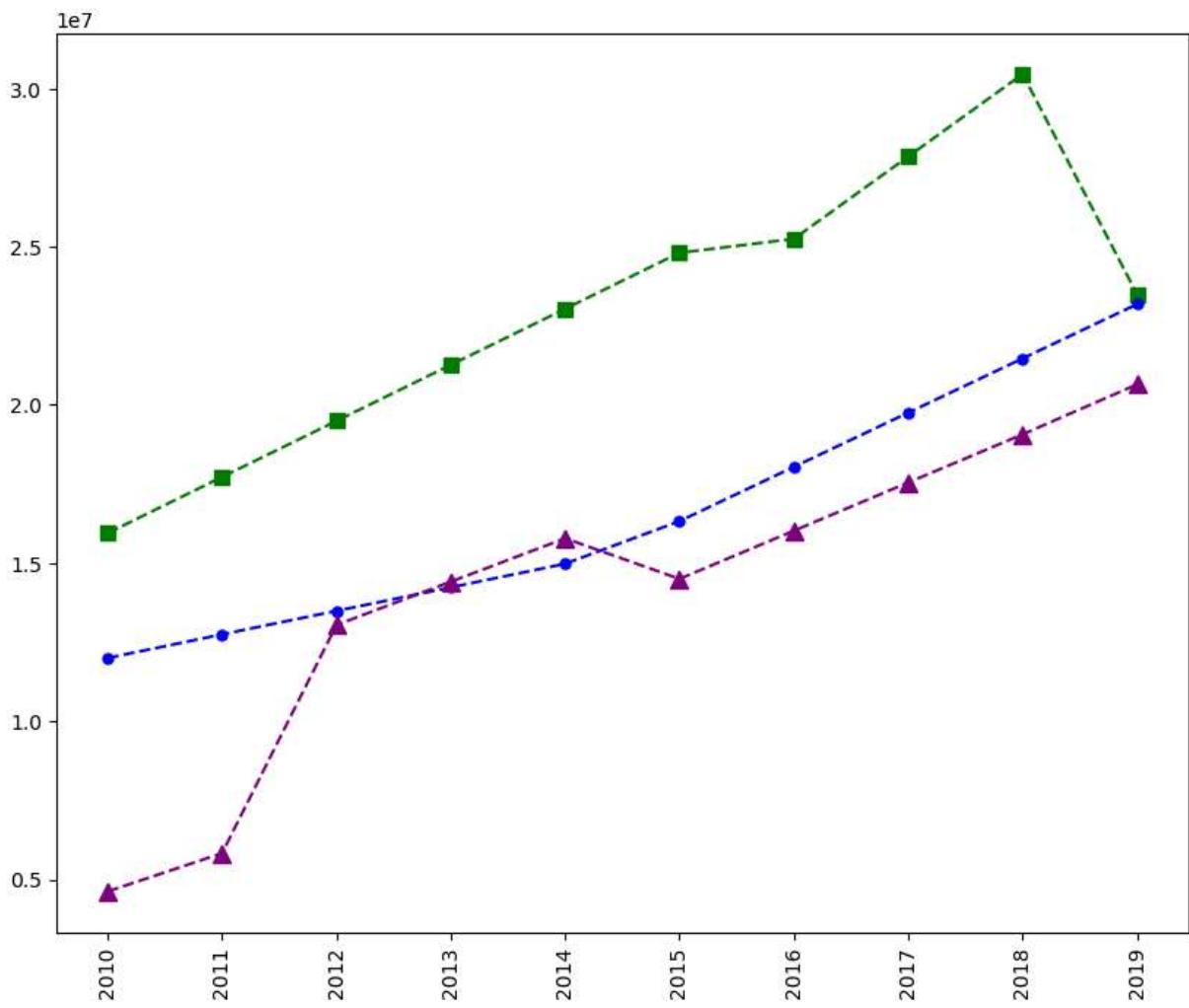


In [115]:

```
plt.plot(Salary[0], c='Green', ls = '--', marker = 's', ms = 7, label = Players[0])
plt.plot(Salary[1], c='Blue', ls = '--', marker = 'o', ms = 5, label = Players[1])
plt.plot(Salary[2], c='purple', ls = '--', marker = '^', ms = 8, label = Players[2])

plt.xticks(list(range(0,10)), Seasons, rotation='vertical')

plt.show()
```

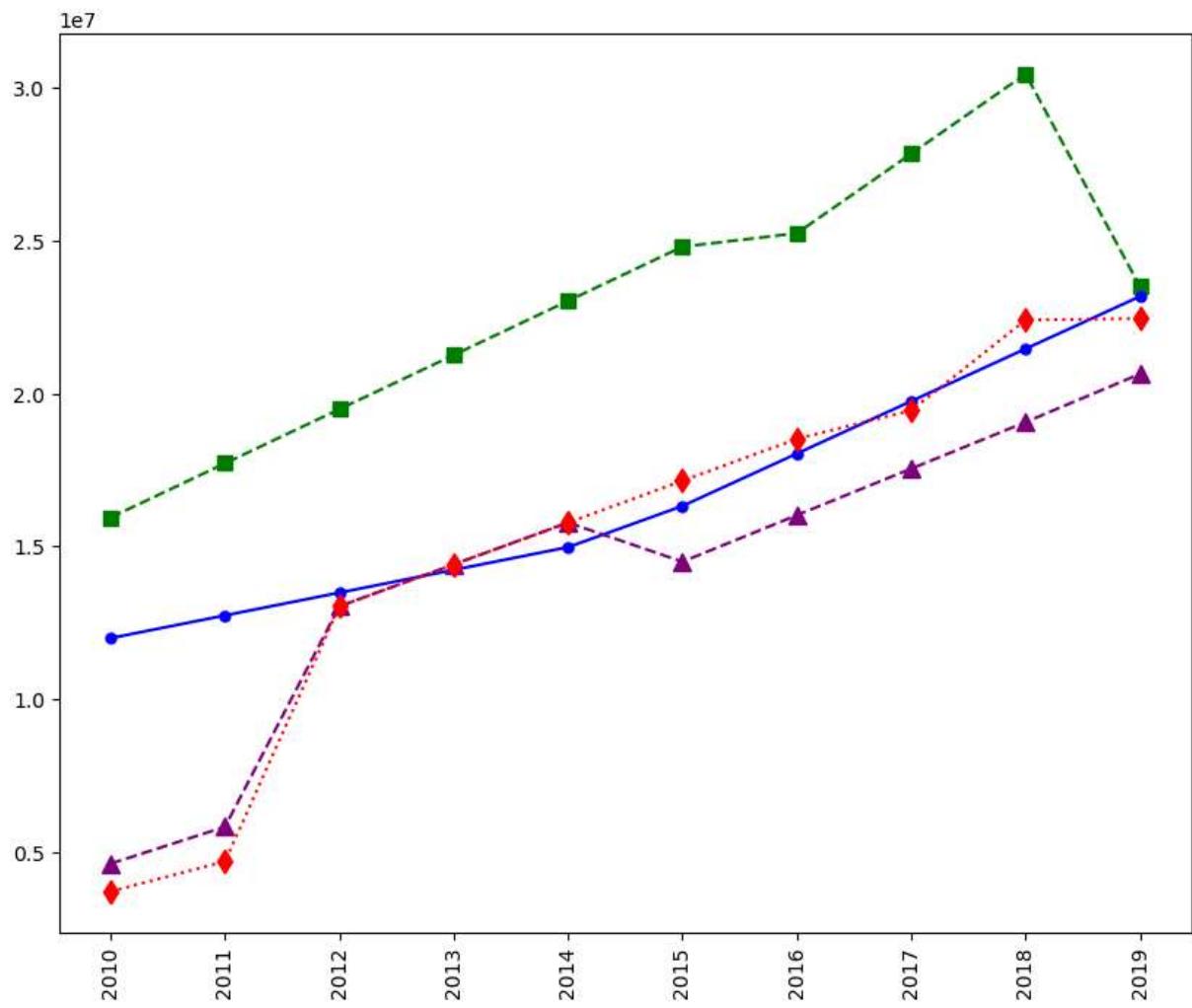


In [116]:

```
plt.plot(Salary[0], c='Green', ls = '--', marker = 's', ms = 7, label = Players[0])
plt.plot(Salary[1], c='Blue', ls = '-.', marker = 'o', ms = 5, label = Players[1])
plt.plot(Salary[2], c='purple', ls = '--', marker = '^', ms = 8, label = Players[2])
plt.plot(Salary[3], c='Red', ls = ':', marker = 'd', ms = 8, label = Players[3])

plt.xticks(list(range(0,10)), Seasons, rotation='vertical')

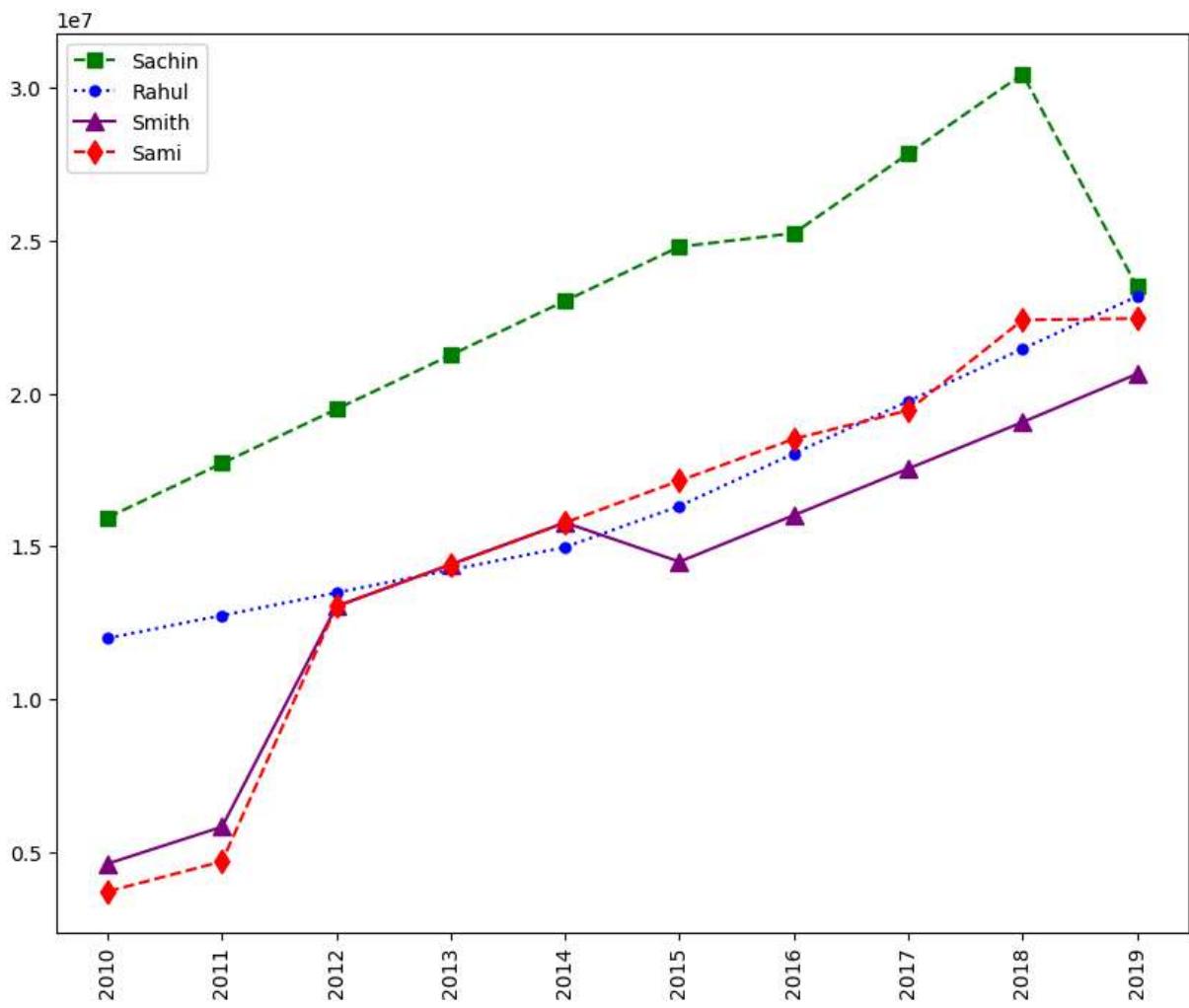
plt.show()
```



In [117]: # how to add Legned in visualisation

```
plt.plot(Salary[0], c='Green', ls = '--', marker = 's', ms = 7, label = Players[0])
plt.plot(Salary[1], c='Blue', ls = ':', marker = 'o', ms = 5, label = Players[1])
plt.plot(Salary[2], c='purple', ls = '-.', marker = '^', ms = 8, label = Players[2])
plt.plot(Salary[3], c='Red', ls = '--', marker = 'd', ms = 8, label = Players[3])
plt.legend()
plt.xticks(list(range(0,10)), Seasons, rotation='vertical')

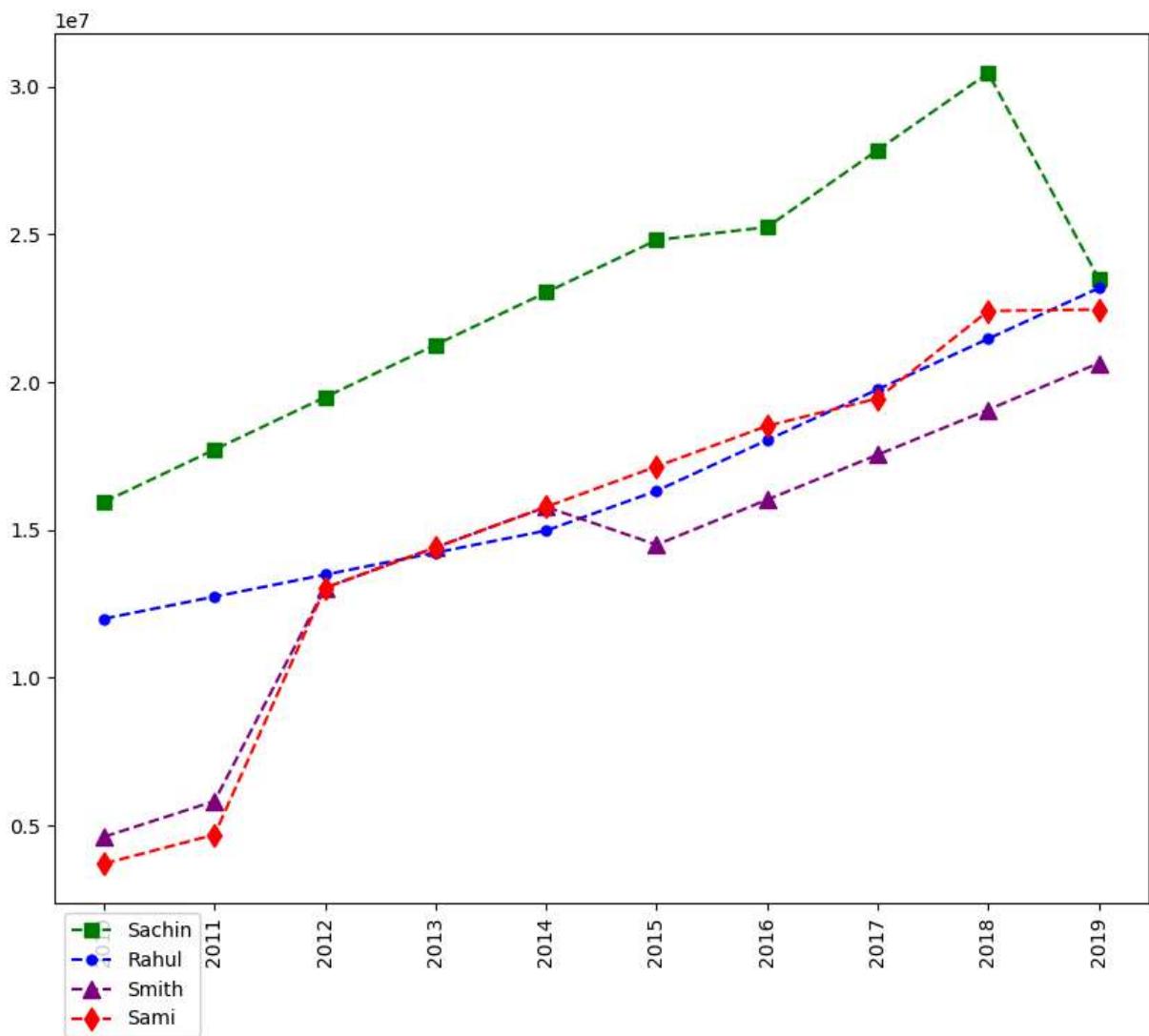
plt.show()
```



In [118...]

```
plt.plot(Salary[0], c='Green', ls = '--', marker = 's', ms = 7, label = Players[0])
plt.plot(Salary[1], c='Blue', ls = '--', marker = 'o', ms = 5, label = Players[1])
plt.plot(Salary[2], c='purple', ls = '--', marker = '^', ms = 8, label = Players[2])
plt.plot(Salary[3], c='Red', ls = '--', marker = 'd', ms = 8, label = Players[3])
plt.legend(loc = 'upper left',bbox_to_anchor=(0,0) )
plt.xticks(list(range(0,10)), Seasons, rotation='vertical')

plt.show()
```



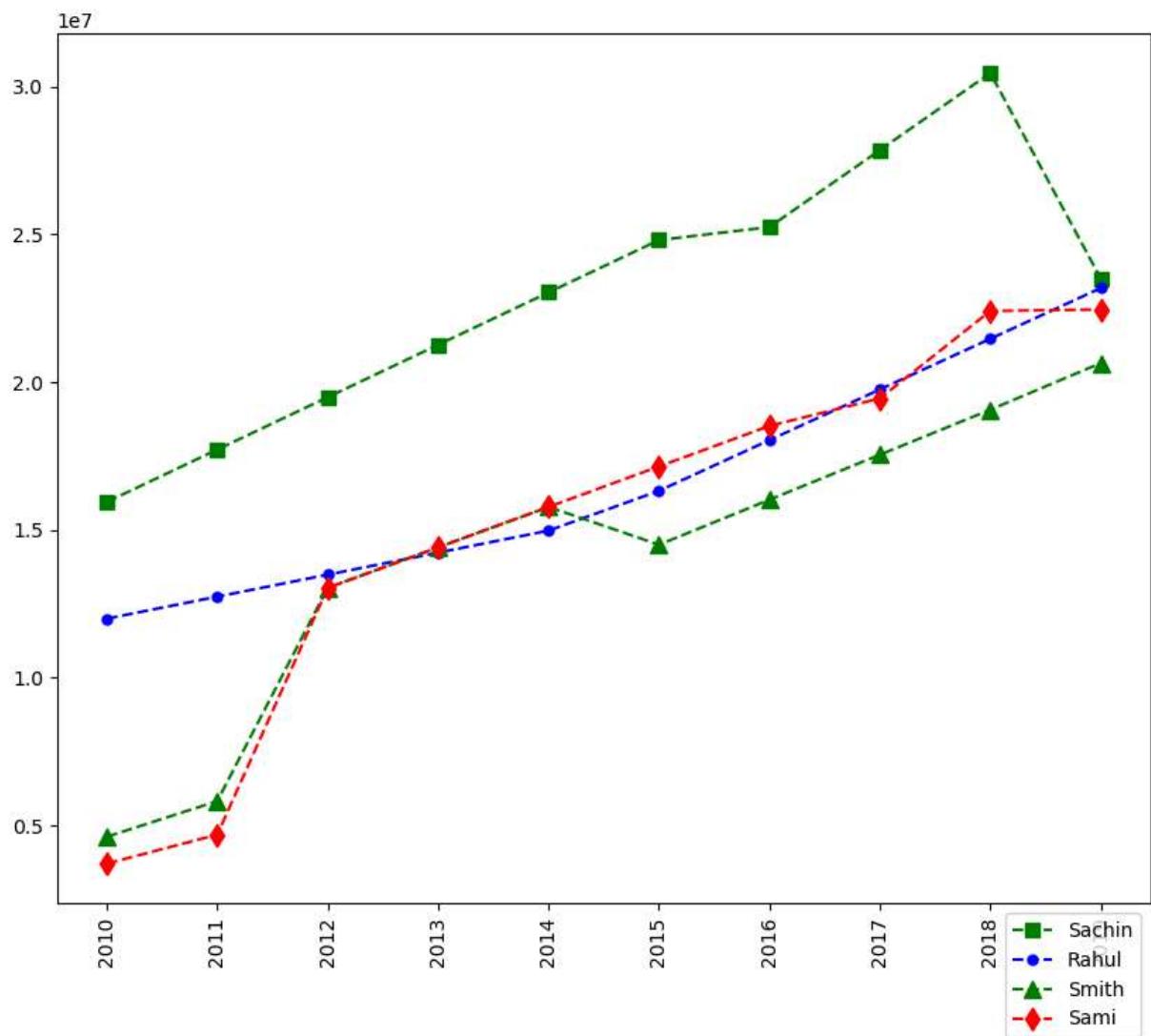
In [119]:

```

plt.plot(Salary[0], c='Green', ls = '--', marker = 's', ms = 7, label = Players[0])
plt.plot(Salary[1], c='Blue', ls = '--', marker = 'o', ms = 5, label = Players[1])
plt.plot(Salary[2], c='Green', ls = '--', marker = '^', ms = 8, label = Players[2])
plt.plot(Salary[3], c='Red', ls = '--', marker = 'd', ms = 8, label = Players[3])
plt.legend(loc = 'upper right',bbox_to_anchor=(1,0) )
plt.xticks(list(range(0,10)), Seasons, rotation='vertical')

plt.show()

```



In [120]:

```

plt.plot(Salary[0], c='Green', ls = '--', marker = 's', ms = 7, label = Players[0])
plt.plot(Salary[1], c='Blue', ls = '--', marker = 'o', ms = 5, label = Players[1])
plt.plot(Salary[2], c='Green', ls = '--', marker = '^', ms = 8, label = Players[2])
plt.plot(Salary[3], c='Red', ls = '--', marker = 'd', ms = 8, label = Players[3])
plt.legend(loc = 'lower right',bbox_to_anchor=(0.5,1))
plt.xticks(list(range(0,10)), Seasons, rotation='vertical')

plt.show()

```

```

-----
ValueError                                Traceback (most recent call last)
Cell In[120], line 4
      2 plt.plot(Salary[1], c='Blue', ls = '--', marker = 'o', ms = 5, label = Playe
rs[1])
      3 plt.plot(Salary[2], c='Green', ls = '--', marker = '^', ms = 8, label = Play
ers[2])
--> 4 plt.plot(Salary[3], c='Read', ls = '--', marker = 'd', ms = 8, label = Playe
rs[3])
      5 plt.legend(loc = 'lower right',bbox_to_anchor=(0.5,1) )
      6 plt.xticks(list(range(0,10)), Seasons,rotation='vertical')

File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\pyplot.py:3590, in plot(s
calex, scaley, data, *args, **kwargs)
    3582 @_copy_docstring_and_deprecators(Axes.plot)
    3583 def plot(
    3584     *args: float | ArrayLike | str,
    (...),
    3588     **kwargs,
    3589 ) -> list[Line2D]:
-> 3590     return gca().plot(
    3591         *args,
    3592         scalex=scalex,
    3593         scaley=scaley,
    3594         **({ "data": data} if data is not None else {}),
    3595         **kwargs,
    3596     )

File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\axes\_axes.py:1724, in Ax
es.plot(self, scalex, scaley, data, *args, **kwargs)
    1481 """
    1482 Plot y versus x as lines and/or markers.
    1483
    (...),
    1721 ('`'green'``) or hex strings (``#008000``).
    1722 """
    1723 kwargs = cbook.normalize_kwargs(kwargs, mlines.Line2D)
-> 1724 lines = [*self._get_lines(self, *args, data=data, **kwargs)]
    1725 for line in lines:
    1726     self.add_line(line)

File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\axes\_base.py:303, in _pr
ocess_plot_var_args.__call__(self, axes, data, *args, **kwargs)
    301     this += args[0],
    302     args = args[1:]
--> 303 yield from self._plot_args(
    304     axes, this, kwargs, ambiguous_fmt_datakey=ambiguous_fmt_datakey)

File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\axes\_base.py:539, in _pr
ocess_plot_var_args._plot_args(self, axes, tup, kwargs, return_kwargs, ambiguous_fmt
_datakey)
    537     return list(result)
    538 else:
--> 539     return [l[0] for l in result]

File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\axes\_base.py:532, in <ge

```

```

nexpr>(.0)
 529 else:
 530     labels = [label] * n_datasets
--> 532 result = (make_artist(axes, x[:, j % ncx], y[:, j % ncy], kw,
 533                               {**kwargs, 'label': label})
 534             for j, label in enumerate(labels))
 536 if return_kwargs:
 537     return list(result)

File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\axes\_base.py:346, in _process_plot_var_args._makeline(self, axes, x, y, kw, kwargs)
 344 default_dict = self._getdefaults(set(), kw)
 345 self._setdefaults(default_dict, kw)
--> 346 seg = mlines.Line2D(x, y, **kw)
 347 return seg, kw

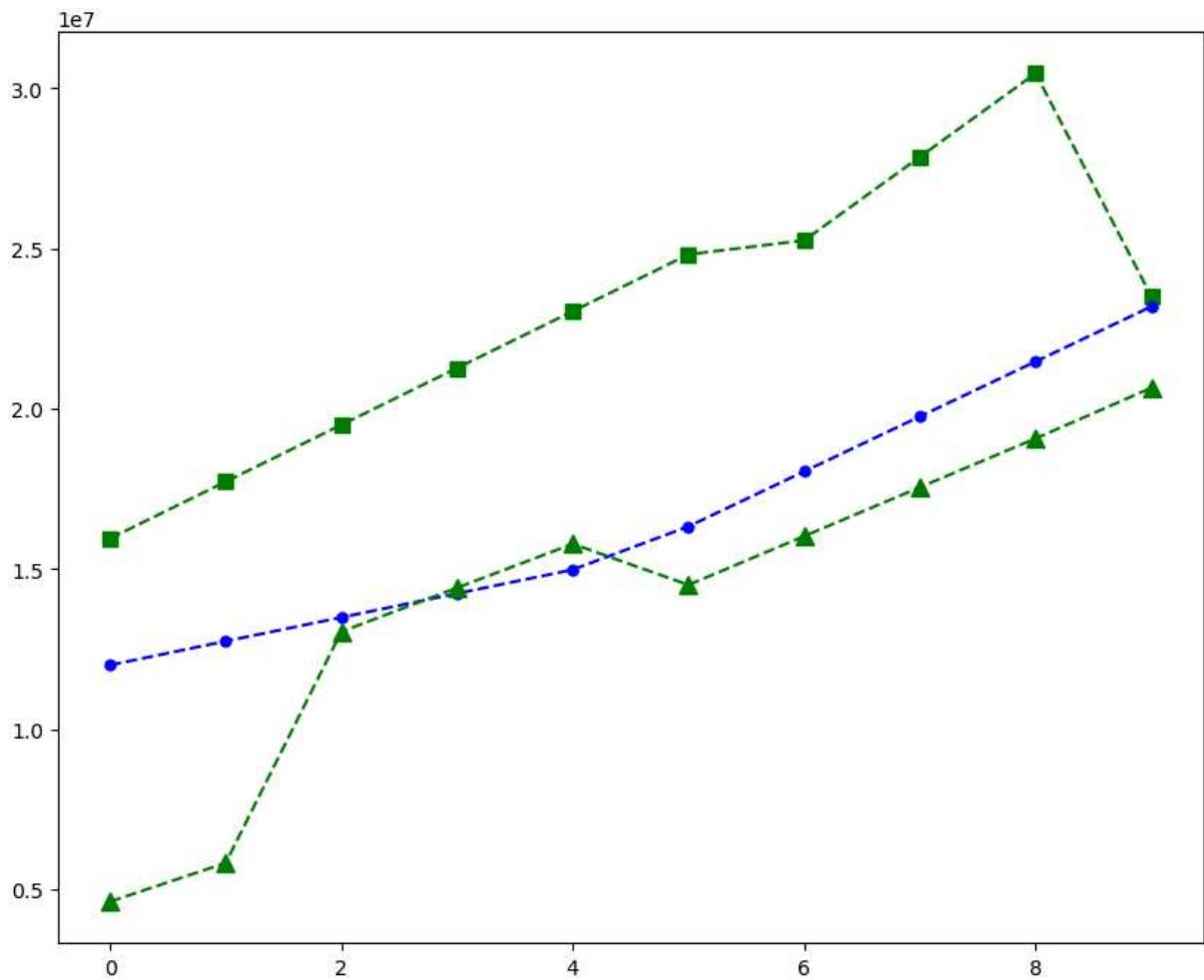
File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\lines.py:376, in Line2D.__init__(self, xdata, ydata, linewidth, linestyle, color, gapcolor, marker, markersize, markeredgewidth, markeredgecolor, markerfacecolor, markerfacecoloralt, fillstyle, antialiased, dash_capstyle, solid_capstyle, dash_joinstyle, solid_joinstyle, pickradius, drawstyle, markevery, **kwargs)
 373 self.set_drawstyle(drawstyle)
 375 self._color = None
--> 376 self.set_color(color)
 377 if marker is None:
 378     marker = 'none' # Default.

File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\lines.py:1061, in Line2D.set_color(self, color)
 1053 def set_color(self, color):
 1054     """
 1055     Set the color of the line.
 1056
 1057     ...
 1058     color : color
 1059     """
 1060
-> 1061     mcolors._check_color_like(color=color)
 1062     self._color = color
 1063     self.stale = True

File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\colors.py:246, in _check_color_like(**kwargs)
 244 for k, v in kwargs.items():
 245     if not is_color_like(v):
--> 246         raise ValueError(f"{v!r} is not a valid value for {k}")

ValueError: 'Read' is not a valid value for color

```



In [121...]

```
plt.plot(Salary[0], c='Green', ls = '--', marker = 's', ms = 7, label = Players[0])
plt.plot(Salary[1], c='Blue', ls = '--', marker = 'o', ms = 7, label = Players[1])
plt.plot(Salary[2], c='Green', ls = '--', marker = '^', ms = 7, label = Players[2])
plt.plot(Salary[3], c='Purple', ls = '--', marker = 'D', ms = 7, label = Players[3])
plt.plot(Salary[4], c='Black', ls = '--', marker = 's', ms = 7, label = Players[4])
plt.plot(Salary[5], c='Red', ls = '--', marker = 'o', ms = 7, label = Players[5])
plt.plot(Salary[6], c='Red', ls = '--', marker = '^', ms = 7, label = Players[6])
plt.plot(Salary[7], c='Red', ls = '--', marker = 'd', ms = 7, label = Players[7])
plt.plot(Salary[8], c='Red', ls = '--', marker = 's', ms = 7, label = Players[8])
plt.plot(Salary[9], c='Red', ls = '--', marker = 'o', ms = 7, label = Players[9])

plt.legend(loc = 'lower right',bbox_to_anchor=(0.5,1) )
plt.xticks(list(range(0,10)), Seasons, rotation='vertical')

plt.show()
```

```

-----
ValueError                                Traceback (most recent call last)
Cell In[121], line 12
    9 plt.plot(Salary[8], c='Red', ls = '--', marker = 's', ms = 7, label = Player
   10 s[8])
   11 plt.plot(Salary[9], c='Red', ls = '--', marker = 'o', ms = 7, label = Player
   12 s[9])
--> 12 plt.legend(loc = 'lower right',bbox_to_anchor=(0.5,1) )
   13 plt.xticks(list(range(0,10)), Seasons, rotation='vertical')
   14 plt.show()

File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\pyplot.py:3384, in legend(*args, **kwargs)
    3382 @_copy_docstring_and_deprecators(Axes.legend)
    3383 def legend(*args, **kwargs) -> Legend:
-> 3384     return gca().legend(*args, **kwargs)

File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\axes\_axes.py:323, in Axes.legend(self, *args, **kwargs)
    206 """
    207 Place a legend on the Axes.
    208
    (...)

 320 .. plot:: gallery/text_labels_and_annotations/legend.py
 321 """
 322 handles, labels, kwargs = mlegend._parse_legend_args([self], *args, **kwargs)
--> 323 self.legend_ = mlegend.Legend(self, handles, labels, **kwargs)
 324 self.legend_.remove_method = self._remove_legend
 325 return self.legend_

File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\legend.py:566, in Legend.__init__(self, parent, handles, labels, loc, numpoints, markerscale, markerfirst, reverse, scatterpoints, scatteryoffsets, prop, fontsize, labelcolor, borderpad, labels_pacing, handlelength, handleheight, handletextpad, borderaxespad, columnspacing, nocols, mode, fancybox, shadow, title, title_fontsize, framealpha, edgecolor, facecolor, bbox_to_anchor, bbox_transform, frameon, handler_map, title_fontproperties, alignment, ncol, draggable)
    563 self._init_legend_box(handles, labels, markerfirst)
    565 # Set legend location
--> 566 self.set_loc(loc)
    568 # figure out title font properties:
    569 if title_fontsize is not None and title_fontproperties is not None:

File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\legend.py:687, in Legend.set_loc(self, loc)
    685         loc = locs[0] + ' ' + locs[1]
    686     # check that loc is in acceptable strings
--> 687     loc = _api.check_getitem(self.codes, loc=loc)
    688 elif np.iterable(loc):
    689     # coerce iterable into tuple
    690     loc = tuple(loc)

File C:\ProgramData\anaconda3\Lib\site-packages\matplotlib\_api\__init__.py:183, in check_getitem(mapping, **kwargs)
    181     return mapping[v]

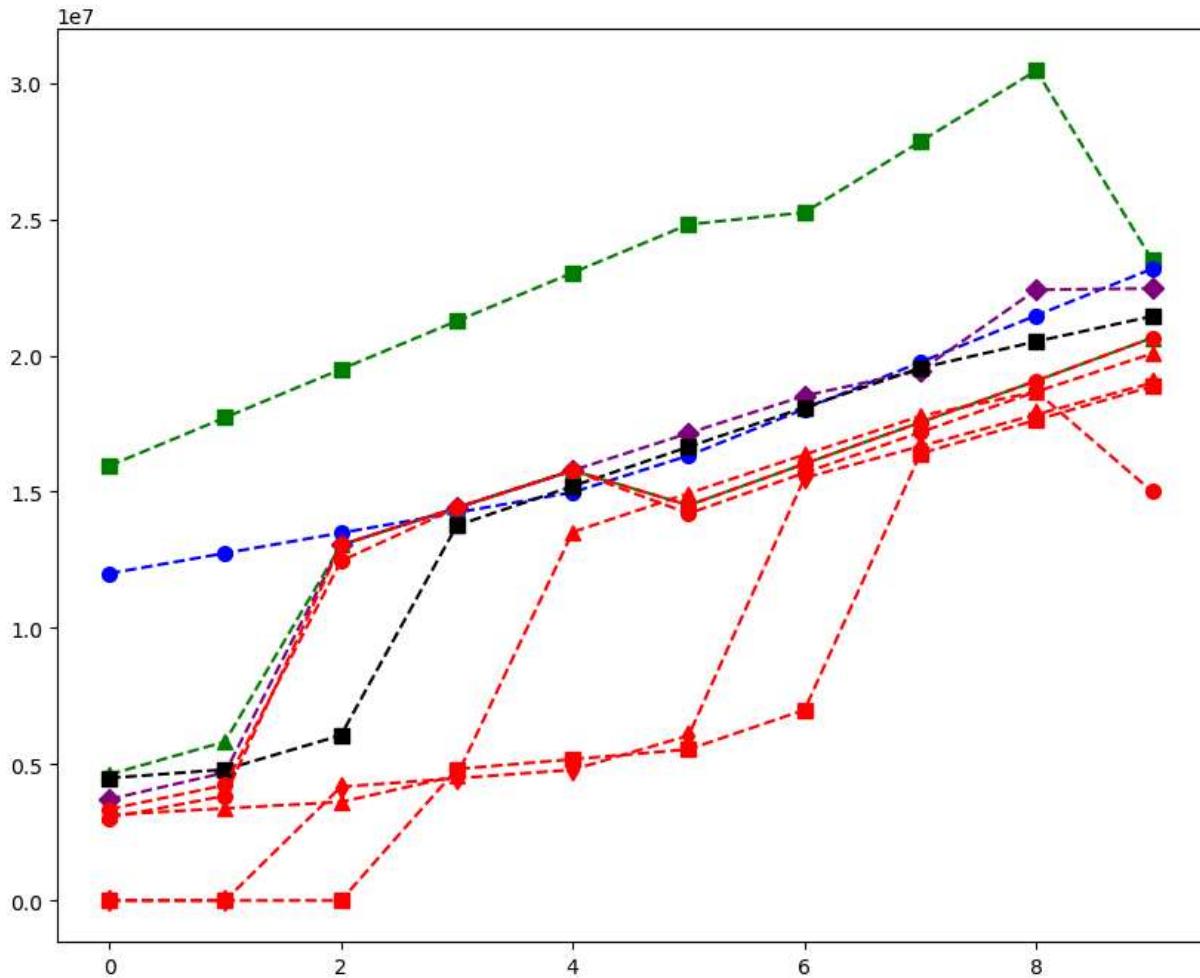
```

```

182 except KeyError:
--> 183     raise ValueError(
184         f"{v!r} is not a valid value for {k}; supported values are "
185         f"{'', '.join(map(repr, mapping))}'") from None

```

ValueError: 'lower right' is not a valid value for loc; supported values are 'best', 'upper right', 'upper left', 'lower left', 'lower right', 'right', 'center left', 'center right', 'lower center', 'upper center', 'center'



In [122...]

```
# we can visualize the how many games played by a player
```

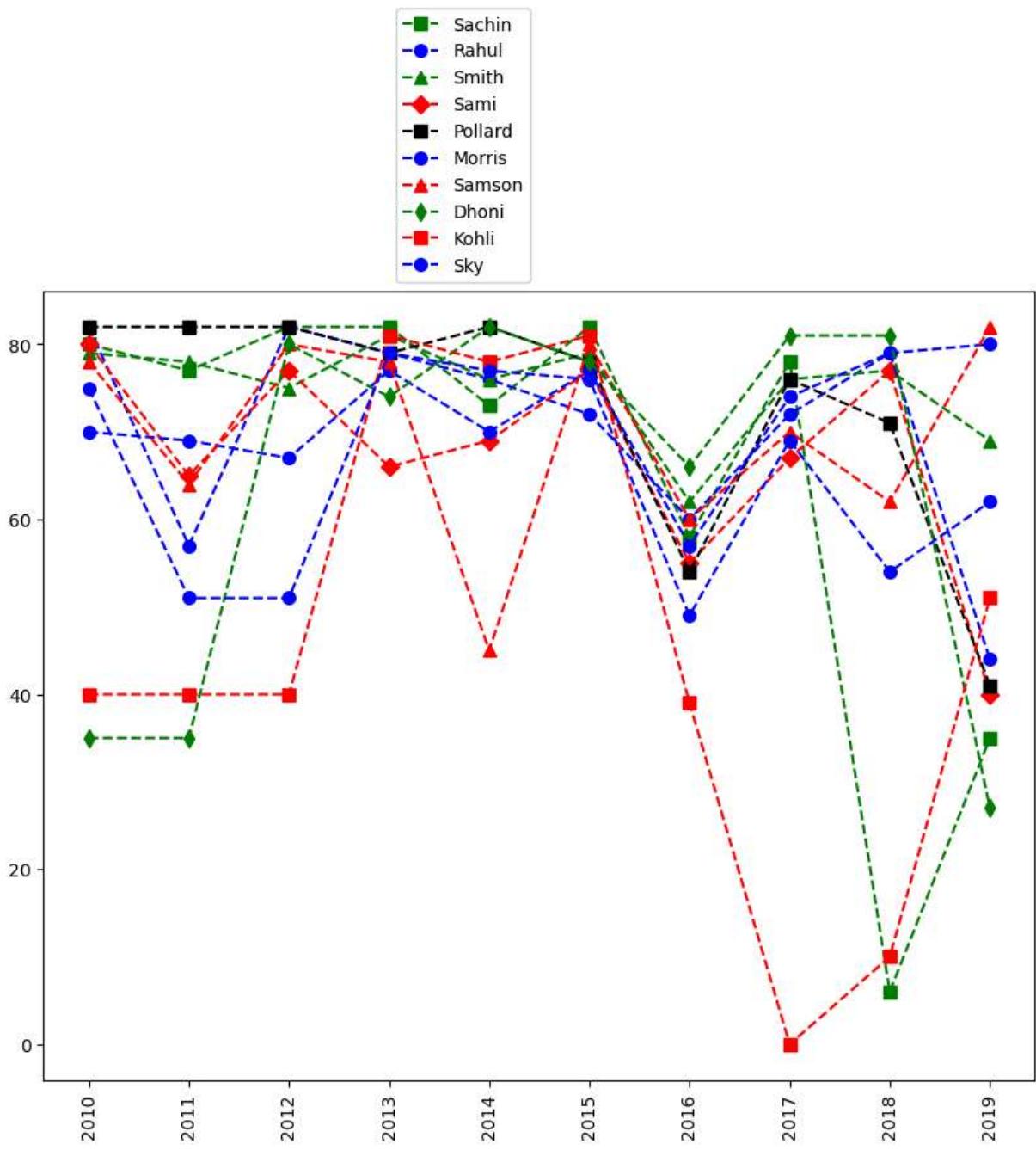
```

plt.plot(Games[0], c='Green', ls = '--', marker = 's', ms = 7, label = Players[0])
plt.plot(Games[1], c='Blue', ls = '--', marker = 'o', ms = 7, label = Players[1])
plt.plot(Games[2], c='Green', ls = '--', marker = '^', ms = 7, label = Players[2])
plt.plot(Games[3], c='Red', ls = '--', marker = 'D', ms = 7, label = Players[3])
plt.plot(Games[4], c='Black', ls = '--', marker = 's', ms = 7, label = Players[4])
plt.plot(Games[5], c='Blue', ls = '--', marker = 'o', ms = 7, label = Players[5])
plt.plot(Games[6], c='red', ls = '--', marker = '^', ms = 7, label = Players[6])
plt.plot(Games[7], c='Green', ls = '--', marker = 'd', ms = 7, label = Players[7])
plt.plot(Games[8], c='Red', ls = '--', marker = 's', ms = 7, label = Players[8])
plt.plot(Games[9], c='Blue', ls = '--', marker = 'o', ms = 7, label = Players[9])

plt.legend(loc = 'lower right',bbox_to_anchor=(0.5,1))
plt.xticks(list(range(0,10)), Seasons, rotation='vertical')

plt.show()

```



In []:

In []:

In []:

In []:

In []: