

```
In [6]: #Import numpy
import numpy as np

#Seasons
Seasons = ["2015", "2016", "2017", "2018", "2019", "2020", "2021", "2022", "2023", "2024"]
Sdict = {"2015":0, "2016":1, "2017":2, "2018":3, "2019":4, "2020":5, "2021":6, "2022":7, "2023":8, "2024":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, 27810000, 30612500, 34215000, 380160, 4171200, 4484040, 4796880, 6053663, 15506632, 16669630, 17832627, 18990000, 201920, 23841443, 26041250, 294410581, 315779912, 34149243, 37518574, 39450000, 4293160, 45828090, 48061274, 513758000, 54202590, 56647180, 58091770, 59536000, 6144240, 6380160, 6615960, 6874189, 703520500, 72940153, 75359805, 77794500, 7914420, 81380160, 83615960, 8574189, 873520500, 89940153, 92359805, 94779450, 9614420, 98380160, 100615960, 10274189, 1053520500, 107940153, 110359805, 112779450, 11414420, 116380160, 118615960, 12074189, 1233520500, 125940153, 128359805, 130779450, 13214420, 134380160, 136615960, 13874189, 1403520500, 142940153, 145359805, 147779450, 14914420, 151380160, 153615960, 15574189, 1573520500, 159940153, 162359805, 164779450, 16614420, 168380160, 170615960, 17274189, 1743520500, 176940153, 179359805, 181779450, 18314420, 185380160, 187615960, 18974189, 1913520500, 193940153, 196359805, 198779450, 19914420, 201380160, 203615960, 20574189, 2073520500, 209940153, 212359805, 214779450, 21614420, 218380160, 220615960, 22274189, 2243520500, 226940153, 229359805, 231779450, 23314420, 235380160, 237615960, 23974189, 2413520500, 243940153, 246359805, 248779450, 24914420, 251380160, 253615960, 25574189, 2573520500, 259940153, 262359805, 264779450, 26614420, 268380160, 270615960, 27274189, 2743520500, 276940153, 279359805, 281779450, 28314420, 285380160, 287615960, 28974189, 2913520500, 293940153, 296359805, 298779450, 29914420, 301380160, 303615960, 30574189, 3073520500, 309940153, 312359805, 314779450, 31614420, 318380160, 320615960, 32274189, 3243520500, 326940153, 329359805, 331779450, 33314420, 335380160, 337615960, 33974189, 3413520500, 343940153, 346359805, 348779450, 34914420, 351380160, 353615960, 35574189, 3573520500, 359940153, 362359805, 364779450, 36614420, 368380160, 370615960, 37274189, 3743520500, 376940153, 379359805, 381779450, 38314420, 385380160, 387615960, 38974189, 3913520500, 393940153, 396359805, 398779450, 39914420, 401380160, 403615960, 40574189, 4073520500, 409940153, 412359805, 414779450, 41614420, 418380160, 420615960, 42274189, 4243520500, 426940153, 429359805, 431779450, 43314420, 435380160, 437615960, 43974189, 4413520500, 443940153, 446359805, 448779450, 44914420, 451380160, 453615960, 45574189, 4573520500, 459940153, 462359805, 464779450, 46614420, 468380160, 470615960, 47274189, 4743520500, 476940153, 479359805, 481779450, 48314420, 485380160, 487615960, 48974189, 4913520500, 493940153, 496359805, 498779450, 49914420, 501380160, 503615960, 50574189, 5073520500, 509940153, 512359805, 514779450, 51614420, 518380160, 520615960, 52274189, 5243520500, 526940153, 529359805, 531779450, 53314420, 535380160, 537615960, 53974189, 5413520500, 543940153, 546359805, 548779450, 54914420, 551380160, 553615960, 55574189, 5573520500, 559940153, 562359805, 564779450, 56614420, 568380160, 570615960, 57274189, 5743520500, 576940153, 579359805, 581779450, 58314420, 585380160, 587615960, 58974189, 5913520500, 593940153, 596359805, 598779450, 59914420, 601380160, 603615960, 60574189, 6073520500, 609940153, 612359805, 614779450, 61614420, 618380160, 620615960, 62274189, 6243520500, 626940153, 629359805, 631779450, 63314420, 635380160, 637615960, 63974189, 6413520500, 643940153, 646359805, 648779450, 64914420, 651380160, 653615960, 65574189, 6573520500, 659940153, 662359805, 664779450, 66614420, 668380160, 670615960, 67274189, 6743520500, 676940153, 679359805, 681779450, 68314420, 685380160, 687615960, 68974189, 6913520500, 693940153, 696359805, 698779450, 69914420, 701380160, 703615960, 70574189, 7073520500, 709940153, 712359805, 714779450, 71614420, 718380160, 720615960, 72274189, 7243520500, 726940153, 729359805, 731779450, 73314420, 735380160, 737615960, 73974189, 7413520500, 743940153, 746359805, 748779450, 74914420, 751380160, 753615960, 75574189, 7573520500, 759940153, 762359805, 764779450, 76614420, 768380160, 770615960, 77274189, 7743520500, 776940153, 779359805, 781779450, 78314420, 785380160, 787615960, 78974189, 7913520500, 793940153, 796359805, 798779450, 79914420, 801380160, 803615960, 80574189, 8073520500, 809940153, 812359805, 814779450, 81614420, 818380160, 820615960, 82274189, 8243520500, 826940153, 829359805, 831779450, 83314420, 835380160, 837615960, 83974189, 8413520500, 843940153, 846359805, 848779450, 84914420, 851380160, 853615960, 85574189, 8573520500, 859940153, 862359805, 864779450, 86614420, 868380160, 870615960, 87274189, 8743520500, 876940153, 879359805, 881779450, 88314420, 885380160, 887615960, 88974189, 8913520500, 893940153, 896359805, 898779450, 89914420, 901380160, 903615960, 90574189, 9073520500, 909940153, 912359805, 914779450, 91614420, 918380160, 920615960, 92274189, 9243520500, 926940153, 929359805, 931779450, 93314420, 935380160, 937615960, 93974189, 9413520500, 943940153, 946359805, 948779450, 94914420, 951380160, 953615960, 95574189, 9573520500, 959940153, 962359805, 964779450, 96614420, 968380160, 970615960, 97274189, 9743520500, 976940153, 979359805, 981779450, 98314420, 985380160, 987615960, 98974189, 9913520500, 993940153, 996359805, 998779450, 99914420, 1001380160, 1003615960, 100574189, 10073520500, 1009940153, 1012359805, 1014779450, 101614420, 1018380160, 1020615960, 102274189, 10243520500, 1026940153, 1029359805, 1031779450, 103314420, 1035380160, 1037615960, 103974189, 10413520500, 1043940153, 1046359805, 1048779450, 104914420, 1051380160, 1053615960, 105574189, 10573520500, 1059940153, 1062359805, 1064779450, 106614420, 1068380160, 1070615960, 107274189, 10743520500, 1076940153, 1079359805, 1081779450, 108314420, 1085380160, 1087615960, 108974189, 10913520500, 1093940153, 1096359805, 1098779450, 109914420, 1101380160, 1103615960, 110574189, 11073520500, 1109940153, 1112359805, 1114779450, 111614420, 1118380160, 1120615960, 112274189, 11243520500, 1126940153, 1129359805, 1131779450, 113314420, 1135380160, 1137615960, 113974189, 11413520500, 1143940153, 1146359805, 1148779450, 114914420, 1151380160, 1153615960, 115574189, 11573520500, 1159940153, 1162359805, 1164779450, 116614420, 1168380160, 1170615960, 117274189, 11743520500, 1176940153, 1179359805, 1181779450, 118314420, 1185380160, 1187615960, 118974189, 11913520500, 1193940153, 1196359805, 1198779450, 119914420, 1201380160, 1203615960, 120574189, 12073520500, 1209940153, 1212359805, 1214779450, 121614420, 1218380160, 1220615960, 122274189, 12243520500, 1226940153, 1229359805, 1231779450, 123314420, 1235380160, 1237615960, 123974189, 12413520500, 1243940153, 1246359805, 1248779450, 124914420, 1251380160, 1253615960, 125574189, 12573520500, 1259940153, 1262359805, 1264779450, 126614420, 1268380160, 1270615960, 127274189, 12743520500, 1276940153, 1279359805, 1281779450, 128314420, 1285380160, 1287615960, 128974189, 12913520500, 1293940153, 1296359805, 1298779450, 129914420, 1301380160, 1303615960, 130574189, 13073520500, 1309940153, 1312359805, 1314779450, 131614420, 1318380160, 1320615960, 132274189, 13243520500, 1326940153, 1329359805, 1331779450, 133314420, 1335380160, 1337615960, 133974189, 13413520500, 1343940153, 1346359805, 1348779450, 134914420, 1351380160, 1353615960, 135574189, 13573520500, 1359940153, 1362359805, 1364779450, 136614420, 1368380160, 1370615960, 137274189, 13743520500, 1376940153, 1379359805, 1381779450, 138314420, 1385380160, 1387615960, 138974189, 13913520500, 1393940153, 1396359805, 1398779450, 139914420, 1401380160, 1403615960, 140574189, 14073520500, 1409940153, 1412359805, 1414779450, 141614420, 1418380160, 1420615960, 142274189, 14243520500, 1426940153, 1429359805, 1431779450, 143314420, 1435380160, 1437615960, 143974189, 14413520500, 1443940153, 1446359805, 1448779450, 144914420, 1451380160, 1453615960, 145574189, 14573520500, 1459940153, 1462359805, 1464779450, 146614420, 1468380160, 1470615960, 147274189, 14743520500, 1476940153, 1479359805, 1481779450, 148314420, 1485380160, 1487615960, 148974189, 14913520500, 1493940153, 1496359805, 1498779450, 149914420, 1501380160, 1503615960, 150574189, 15073520500, 1509940153, 1512359805, 1514779450, 151614420, 1518380160, 1520615960, 152274189, 15243520500, 1526940153, 1529359805, 1531779450, 153314420, 1535380160, 1537615960, 153974189, 15413520500, 1543940153, 1546359805, 1548779450, 154914420, 1551380160, 1553615960, 155574189, 15573520500, 1559940153, 1562359805, 1564779450, 156614420, 1568380160, 1570615960, 157274189, 15743520500, 1576940153, 1579359805, 1581779450, 158314420, 1585380160, 1587615960, 158974189, 15913520500, 1593940153, 1596359805, 1598779450, 159914420, 1601380160, 1603615960, 160574189, 16073520500, 1609940153, 1612359805, 1614779450, 161614420, 1618380160, 1620615960, 162274189, 16243520500, 1626940153, 1629359805, 1631779450, 163314420, 1635380160, 1637615960, 163974189, 16413520500, 1643940153, 1646359805, 1648779450, 164914420, 1651380160, 1653615960, 165574189, 16573520500, 1659940153, 1662359805, 1664779450, 166614420, 1668380160, 1670615960, 167274189, 16743520500, 1676940153, 1679359805, 1681779450, 168314420, 1685380160, 1687615960, 168974189, 16913520500, 1693940153, 1696359805, 1698779450, 169914420, 1701380160, 1703615960, 170574189, 17073520500, 1709940153, 1712359805, 1714779450, 171614420, 1718380160, 1720615960, 172274189, 17243520500, 1726940153, 1729359805, 1731779450, 173314420, 1735380160, 1737615960, 173974189, 17413520500, 1743940153, 1746359805, 1748779450, 174914420, 1751380160, 1753615960, 175574189, 17573520500, 1759940153, 1762359805, 1764779450, 176614420, 1768380160, 1770615960, 177274189, 17743520500, 1776940153, 1779359805, 1781779450, 178314420, 1785380160, 1787615960, 178974189, 17913520500, 1793940153, 1796359805, 1798779450, 179914420, 1801380160, 1803615960, 180574189, 18073520500, 1809940153, 1812359805, 1814779450, 181614420, 1818380160, 1820615960, 182274189, 18243520500, 1826940153, 1829359805, 1831779450, 183314420, 1835380160, 1837615960, 183974189, 18413520500, 1843940153, 1846359805, 1848779450, 184914420, 1851380160, 1853615960, 185574189, 18573520500, 1859940153, 1862359805, 1864779450, 186614420, 1868380160, 1870615960, 187274189, 18743520500, 1876940153, 1879359805, 1881779450, 188314420, 1885380160, 1887615960, 188974189, 18913520500, 1893940153, 1896359805, 1898779450, 189914420, 1901380160, 1903615960, 190574189, 19073520500, 1909940153, 1912359805, 1914779450, 191614420, 1918380160, 1920615960, 192274189, 19243520500, 1926940153, 1929359805, 1931779450, 193314420, 1935380160, 1937615960, 193974189, 19413520500, 1943940153, 1946359805, 1948779450, 194914420, 1951380160, 1953615960, 195574189, 19573520500, 1959940153, 1962359805, 1964779450, 196614420, 1968380160, 1970615960, 197274189, 19743520500, 1976940153, 1979359805, 1981779450, 198314420, 1985380160, 1987615960, 198974189, 19913520500, 1993940153, 1996359805, 1998779450, 199914420, 2001380160, 2003615960, 200574189, 20073520500, 2009940153, 2012359805, 2014779450, 201614420, 20183
```

```
Out[7]: 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 [8]: Games

```
Out[8]: 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 [9]: Points

```
Out[9]: 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 [10]: Sdict

```
Out[10]: {'2015': 0,
          '2016': 1,
          '2017': 2,
          '2018': 3,
          '2019': 4,
          '2020': 5,
          '2021': 6,
          '2022': 7,
          '2023': 8,
          '2024': 9}
```

In [11]: Pdict

```
Out[11]: {'Sachin': 0,
          'Rahul': 1,
          'Smith': 2,
          'Sami': 3,
          'Pollard': 4,
          'Morris': 5,
          'Samson': 6,
          'Dhoni': 7,
          'Kohli': 8,
          'Sky': 9}
```

In [12]: 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 [13]: np.reshape(mydata,(4,5)) # 5 rows & 4columns

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

In [14]: mydata

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

In [15]: #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[15]: 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]: np.int64(19)
```

```
In [14]: matr1[3,3]
```

```
Out[14]: np.int64(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]: np.int64(11)
```

```
In [17]: mydata
```

```
Out[17]: 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]: np.int64(19)
```

```
In [21]: matr2[0,2]
```

```
Out[21]: np.int64(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 [24]: matr2[-2,-1]
```

```
Out[24]: np.int64(18)
```

```
In [26]: matr2[-3,-3]
```

```
Out[26]: np.int64(7)
```

```
In [27]: mydata
```

```
Out[27]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
   17, 18, 19])
```

```
In [28]: matr3 = np.reshape(mydata, (5,4), order = 'A')
matr3
```

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

```
In [29]: matr2 # Fshaped
```

```
Out[29]: array([[ 0,  5, 10, 15],
   [ 1,  6, 11, 16],
   [ 2,  7, 12, 17],
   [ 3,  8, 13, 18],
   [ 4,  9, 14, 19]])
```

```
In [30]: matr1 # C shaped
```

```
Out[30]: array([[ 0,  1,  2,  3],
   [ 4,  5,  6,  7],
   [ 8,  9, 10, 11],
   [12, 13, 14, 15],
   [16, 17, 18, 19]])
```

```
In [31]: a1 = ['welcome', 'to', 'datascience']
a2 = ['required', 'hard', 'work' ]
a3 = [1,2,3]
```

```
In [32]: [a1,a2,a3] # List same datatype
```

```
Out[32]: [['welcome', 'to', 'datascience'], ['required', 'hard', 'work'], [1, 2, 3]]
```

```
In [33]: np.array([a1,a2,a3]) # u11 - unicode 11 character : 3*3 matrix
```

```
Out[33]: array([['welcome', 'to', 'datascience'],
   ['required', 'hard', 'work'],
   ['1', '2', '3']], dtype='<U21')
```

```
In [34]: Games
```

```
Out[34]: 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 [6]: Games[5]
```

```
Out[6]: array([70, 69, 67, 77, 70, 77, 57, 74, 79, 44])
```

```
In [9]: Games[5,3]
```

```
Out[9]: np.int64(77)
```

```
In [10]: Salary[0]
```

```
Out[10]: array([15946875, 17718750, 19490625, 21262500, 23034375, 24806250,
   25244493, 27849149, 30453805, 23500000])
```

```
In [11]: Games[0]
```

```
Out[11]: array([80, 77, 82, 82, 73, 82, 58, 78, 6, 35])
```

```
In [36]: Games
```

```
Out[36]: 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 [35]: Games[0,5]
```

```
Out[35]: np.int64(82)
```

```
In [37]: Games
```

```
Out[37]: 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 [38]: Games[0:2]
```

```
Out[38]: array([[80, 77, 82, 82, 73, 82, 58, 78, 6, 35],  
                 [82, 57, 82, 79, 76, 72, 60, 72, 79, 80]])
```

```
In [39]: Games
```

```
Out[39]: 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 [40]: Games[1:2]
```

```
Out[40]: array([[82, 57, 82, 79, 76, 72, 60, 72, 79, 80]])
```

```
In [41]: Games[2]
```

```
Out[41]: array([79, 78, 75, 81, 76, 79, 62, 76, 77, 69])
```

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

```
Out[42]: 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 [43]: Games[2,8]
```

```
Out[43]: np.int64(77)
```

```
In [44]: Games
```

```
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],
   [40, 40, 40, 81, 78, 81, 39, 0, 10, 51],
   [75, 51, 51, 79, 77, 76, 49, 69, 54, 62]])
```

In [45]: Games[-3:-1]

```
Out[45]: array([[35, 35, 80, 74, 82, 78, 66, 81, 81, 27],
   [40, 40, 40, 81, 78, 81, 39, 0, 10, 51]])
```

In [46]: Games[-3,-1]

```
Out[46]: np.int64(27)
```

In [47]: Points

```
Out[47]: 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 [48]: Points[0]

```
Out[48]: array([2832, 2430, 2323, 2201, 1970, 2078, 1616, 2133, 83, 782])
```

In [49]: Points

```
Out[49]: 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 [51]: Points[6,1]

```
Out[51]: np.int64(1104)
```

In [52]: Points

```
Out[52]: 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 [53]: Points[3:6]

```
Out[53]: array([[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]])
```

In [54]: Points

```
Out[54]: 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 [55]: Points[-6:-1]

```
Out[55]: np.int64(646)
```

#==== Dictionary =====#

dict does not maintain the order

```
In [56]: dict1 = {'key1': 'val1', 'key2': 'val2', 'key3': 'val3'}
dict1
```

```
Out[56]: {'key1': 'val1', 'key2': 'val2', 'key3': 'val3'}
```

In [58]: dict1['key2']

```
Out[58]: 'val2'
```

```
In [61]: dict2 = {'bang': 2, 'hyd': 'we are hear', 'pune': True}
dict2
```

```
Out[61]: {'bang': 2, 'hyd': 'we are hear', 'pune': True}
```

```
In [63]: dict3 = {'Germany': 'I have been here', 'France': 2, 'Spain': True}
dict3
```

```
Out[63]: {'Germany': 'I have been here', 'France': 2, 'Spain': True}
```

```
In [64]: dict3['Germany']
```

```
Out[64]: 'I have been here'
```

- if you check the dataset seasons & players are dictionary type of data
- if you look at the pdict players names are key part: nos are the values
- dictionary can guide us which player at which level and which row
- main advantage of the dictionary is we don't required to count which no row which players are sitting

```
In [65]: Games
```

```
Out[65]: 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 [66]: Pdict
```

```
Out[66]: {'Sachin': 0,
 'Rahul': 1,
 'Smith': 2,
 'Sami': 3,
 'Pollard': 4,
 'Morris': 5,
 'Samson': 6,
 'Dhoni': 7,
 'Kohli': 8,
 'Sky': 9}
```

```
In [67]: # how do i know player kobe Bryant is at
```

```
Pdict['Sachin']
```

```
Out[67]: 0
```

```
In [68]: Games[0]
```

```
Out[68]: array([80, 77, 82, 82, 73, 82, 58, 78, 6, 35])
```

In [69]: Games

```
Out[69]: 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 [70]: Pdict['Rahul']

Out[70]: 1

In [72]: Games[1]

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

Games

In [74]: Games[Pdict['Rahul']]

Out[74]: array([82, 57, 82, 79, 76, 72, 60, 72, 79, 80])

In [75]: Points

```
Out[75]: 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 [76]: Salary

```
Out[76]: 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 [77]: `Salary[2,4]`

```
Out[77]: np.int64(15779912)
```

In [78]: `Salary`

```
Out[78]: 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 [79]: `Salary[Pdict['Sky']][Sdict['2019']]`

```
Out[79]: np.int64(15779912)
```

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]: Games

```
Out[81]: 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 [82]: Salary / Games

```
C:\Users\ASUS\AppData\Local\Temp\ipykernel_17252\1572766764.py:1: RuntimeWarning: divide by zero encountered in divide
    Salary / Games
```

```
Out[82]: 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 [84]: np.round(Salary/Games)
```

```
C:\Users\ASUS\AppData\Local\Temp\ipykernel_17252\3232172828.py:1: RuntimeWarning: divide by zero encountered in divide
np.round(Salary/Games)
```

```
Out[84]: 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 [12]: `Salary[0] / Games[0]`

```
Out[12]: array([ 199335.9375 ,  230113.63636364,  237690.54878049,
   259298.7804878 ,  315539.38356164,  302515.24390244,
   435249.87931034,  357040.37179487,  5075634.16666667,
   671428.57142857])
```

In [13]: `np.round(Salary[0] / Games[0])`

```
Out[13]: array([ 199336.,  230114.,  237691.,  259299.,  315539.,  302515.,
   435250.,  357040.,  5075634.,  671429.])
```

```
In [4]: import warnings
warnings.filterwarnings('ignore')
# to ignores unwanted error write the code as ignore all
#np.round(FieldGoals/Games)
#FieldGoals/Games # this matrix is lot of decimal points yo can not round
#round()
```

lets visualize the data

In [5]: `import numpy as np`
`import matplotlib.pyplot as plt`

In [90]: `#%matplotlib inline # keep the plot inside jupyter notes insted of getting in other`

In [91]: `Salary`

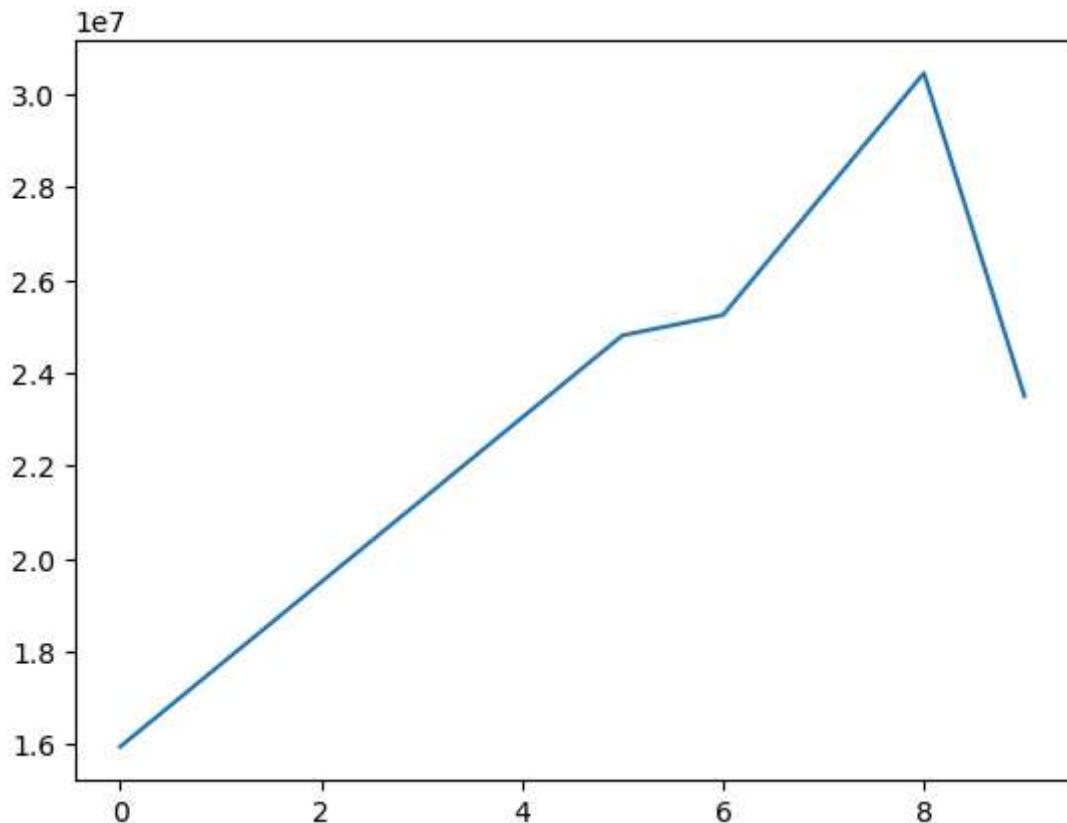
```
Out[91]: array([[15946875, 17718750, 19490625, 21262500, 23034375, 24806250,
   25244493, 27849149, 30453805, 23500000],
   [12000000, 12744189, 13488377, 14232567, 14976754, 16324500,
   18038573, 19752645, 21466718, 23180790],
   [ 4621800, 5828090, 13041250, 14410581, 15779912, 14500000,
   16022500, 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 [92]: Salary[0]
```

```
Out[92]: array([15946875, 17718750, 19490625, 21262500, 23034375, 24806250,
   25244493, 27849149, 30453805, 23500000])
```

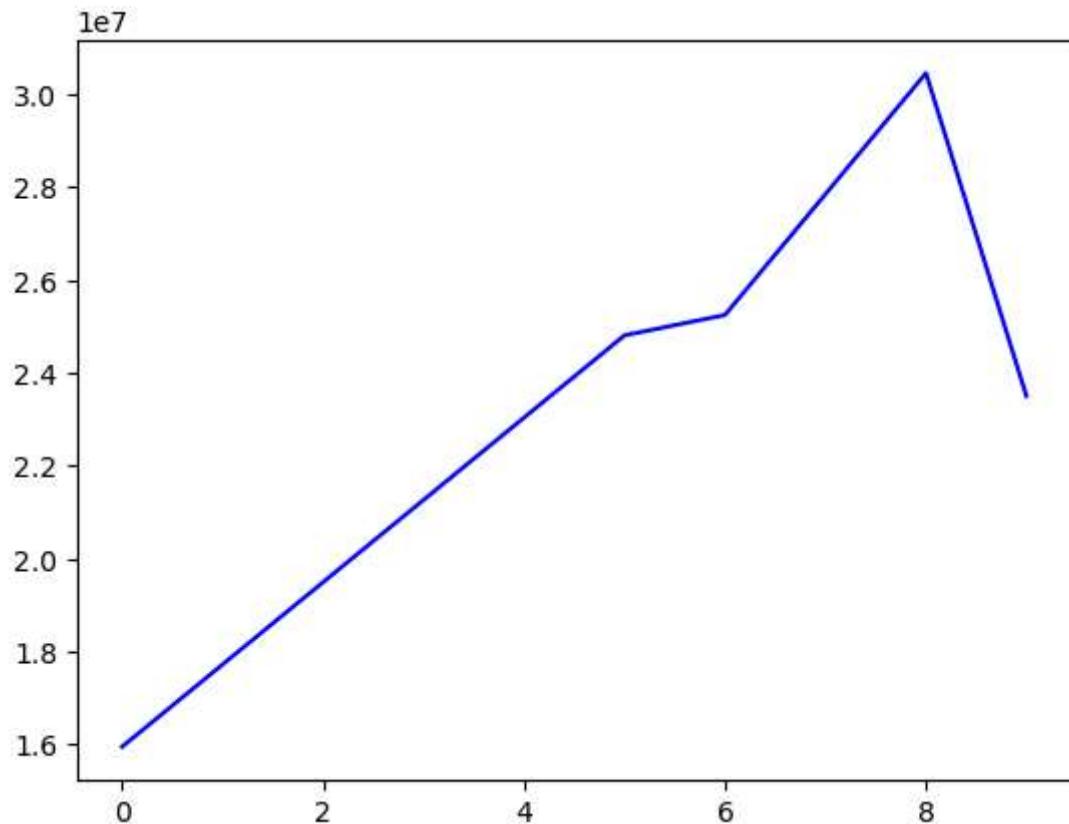
```
In [93]: plt.plot(Salary[0])
```

```
Out[93]: [
```



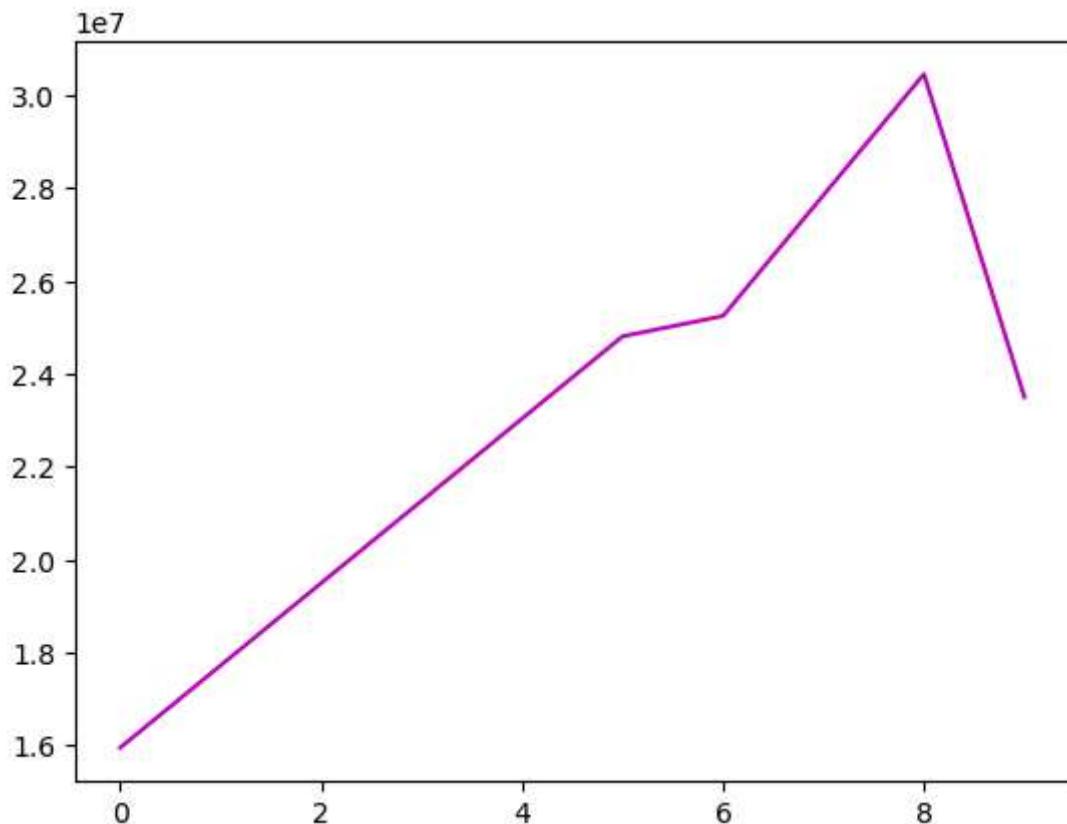
```
In [94]: plt.plot(Salary[0], c = 'blue')
```

```
Out[94]: [
```



```
In [21]: plt.plot(Salary[0], c = 'm')
```

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



```
In [16]: %matplotlib inline  
plt.rcParams['figure.figsize'] = 10,6
```

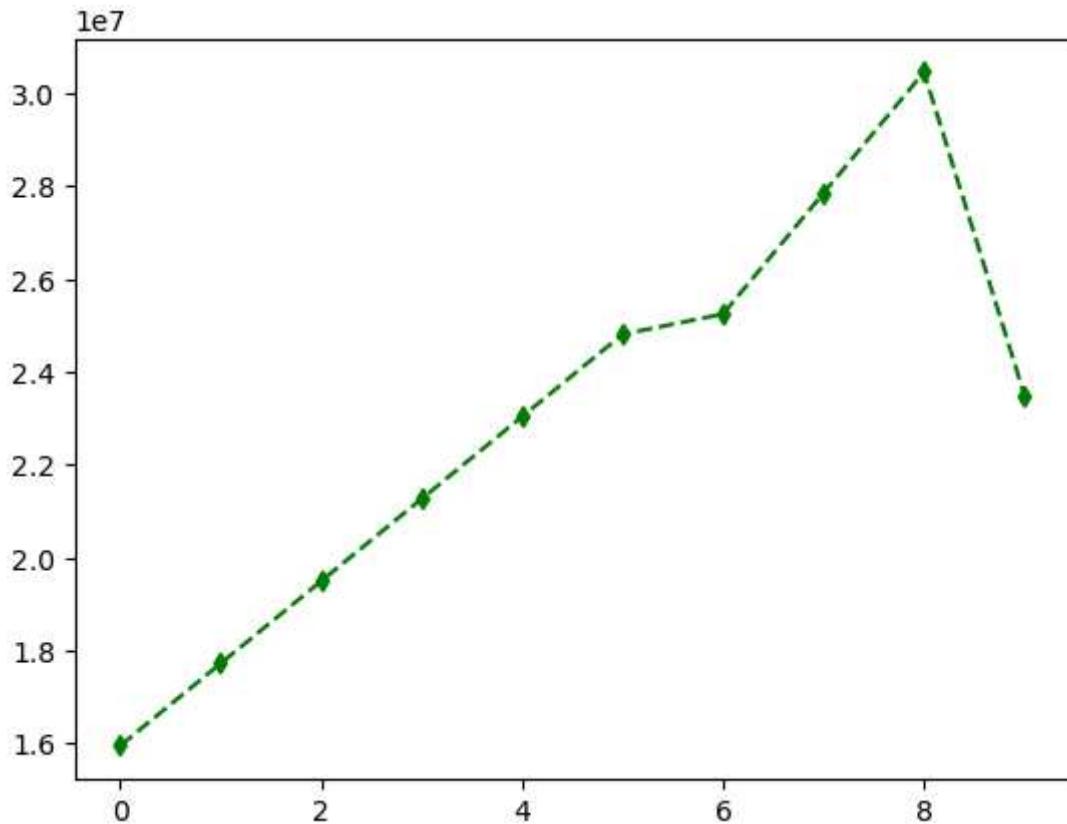
```
In [25]: plt.plot(Salary[0], c = 'g' , ls = '--')
```

```
Out[25]: [<matplotlib.lines.Line2D at 0x24c51925810>]
```

```
In [ ]:
```

```
In [10]: plt.plot(Salary[0], c = 'g' , ls = '--', marker = 'd', ms=5)
```

```
Out[10]: [<matplotlib.lines.Line2D at 0x230c3143610>]
```

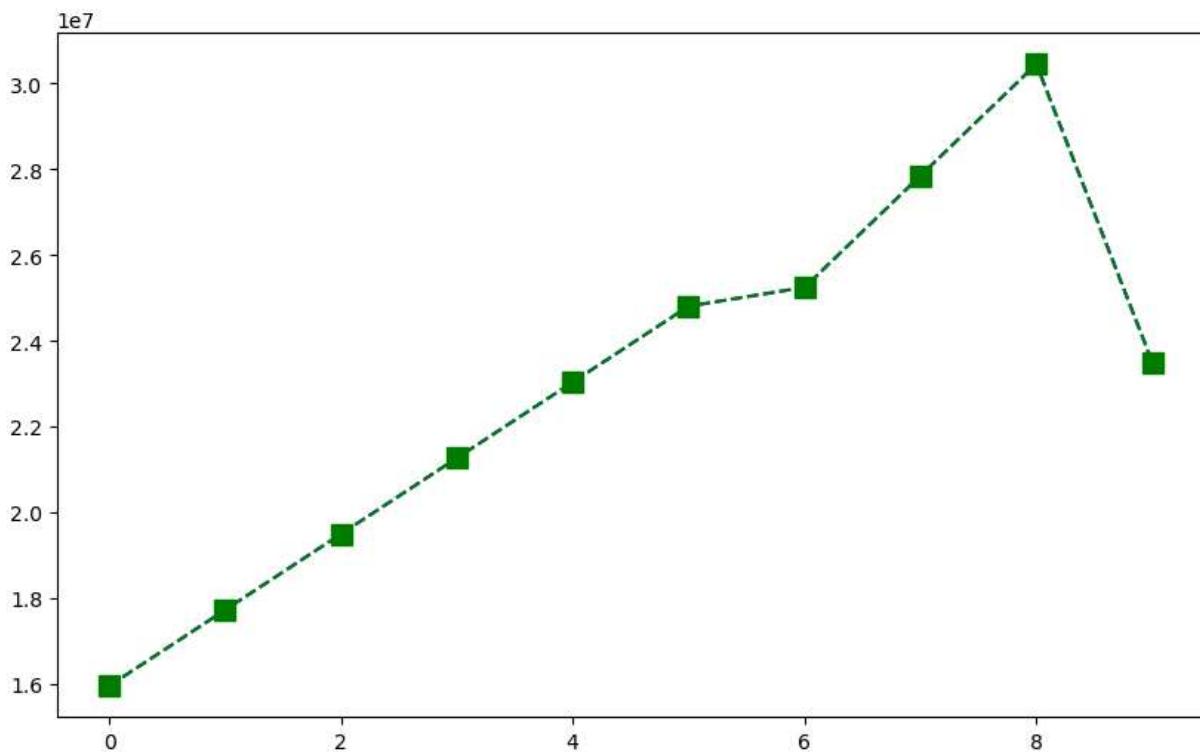


```
In [9]: Games
```

```
Out[9]: 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 [18]: %matplotlib inline  
plt.rcParams['figure.figsize'] = 10,8 #runtime configuration parameter
```

```
In [19]: plt.plot(Salary[0], c='Green', ls = '--', marker = 's', ms = 10)  
plt.show()
```



```
In [26]: list(range(0,10))
```

```
Out[26]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
In [27]: Sdict
```

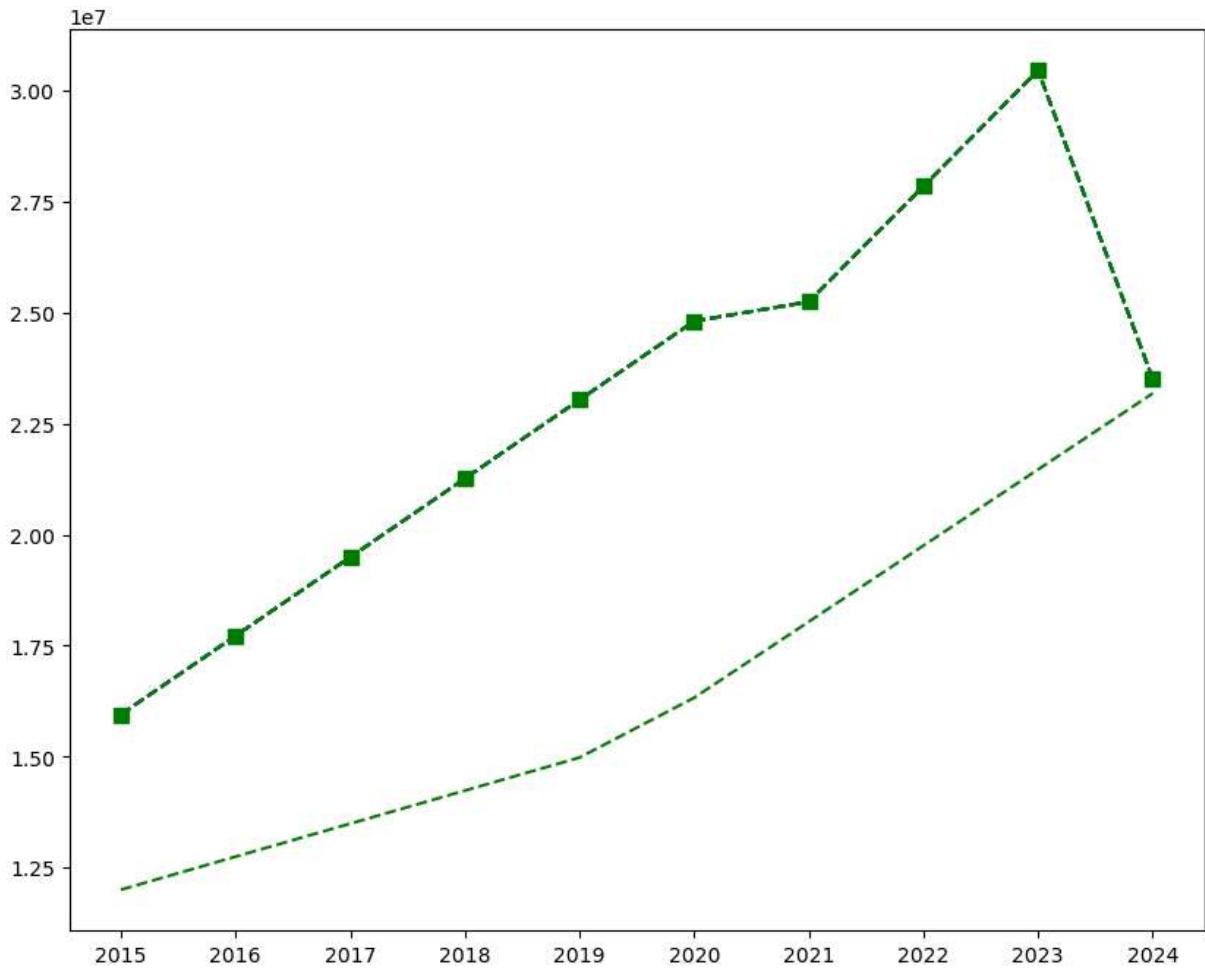
```
Out[27]: {'2015': 0,
          '2016': 1,
          '2017': 2,
          '2018': 3,
          '2019': 4,
          '2020': 5,
          '2021': 6,
          '2022': 7,
          '2023': 8,
          '2024': 9}
```

```
In [28]: Pdict
```

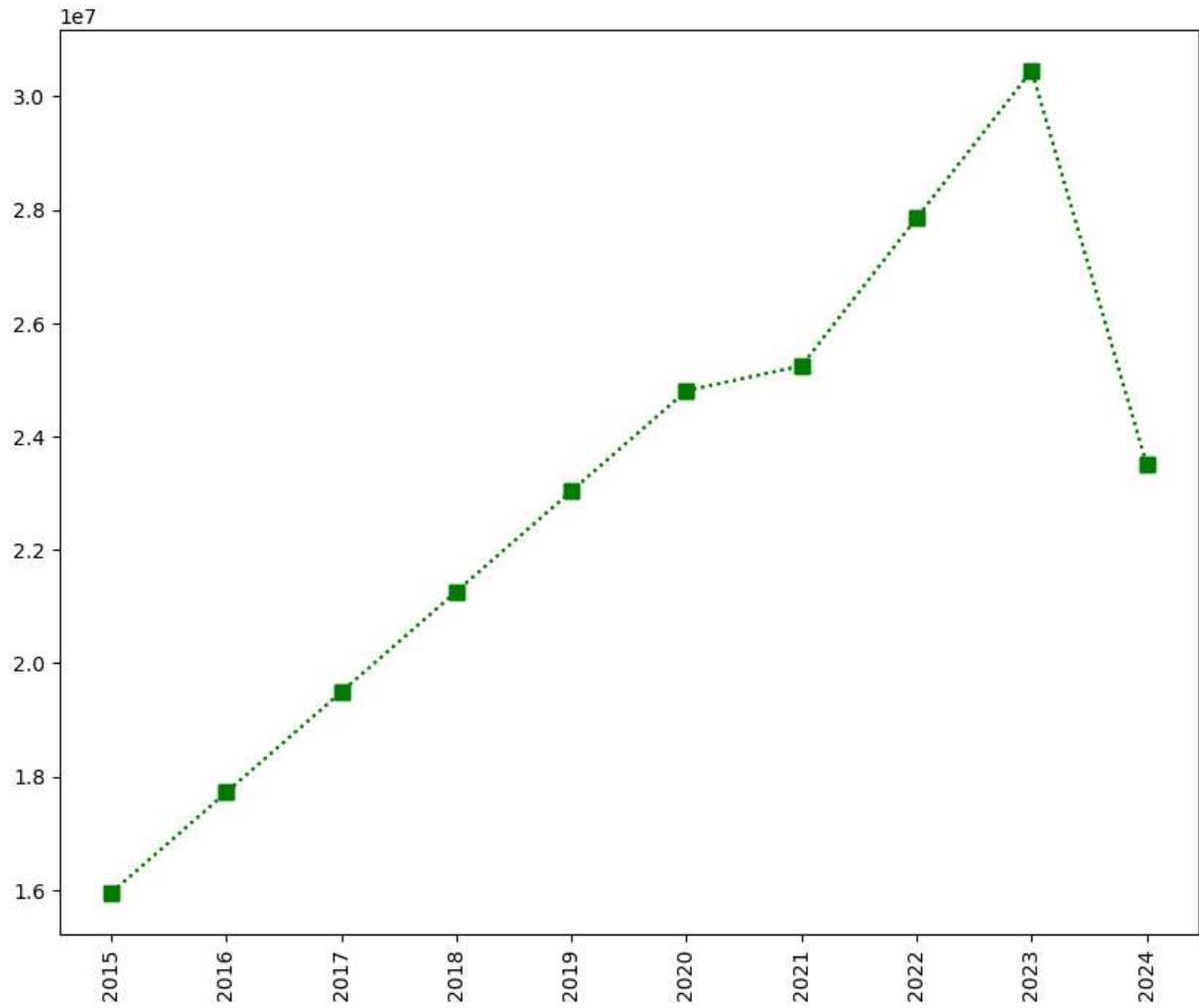
```
Out[28]: {'Sachin': 0,
          'Rahul': 1,
          'Smith': 2,
          'Sami': 3,
          'Pollard': 4,
          'Morris': 5,
          'Samson': 6,
          'Dhoni': 7,
          'Kohli': 8,
          'Sky': 9}
```

```
In [29]: plt.plot(Salary[0], c='Green', ls = '--', marker = 's', ms = 7)
plt.xticks(list(range(0,10)), Seasons)
```

```
plt.show()
```



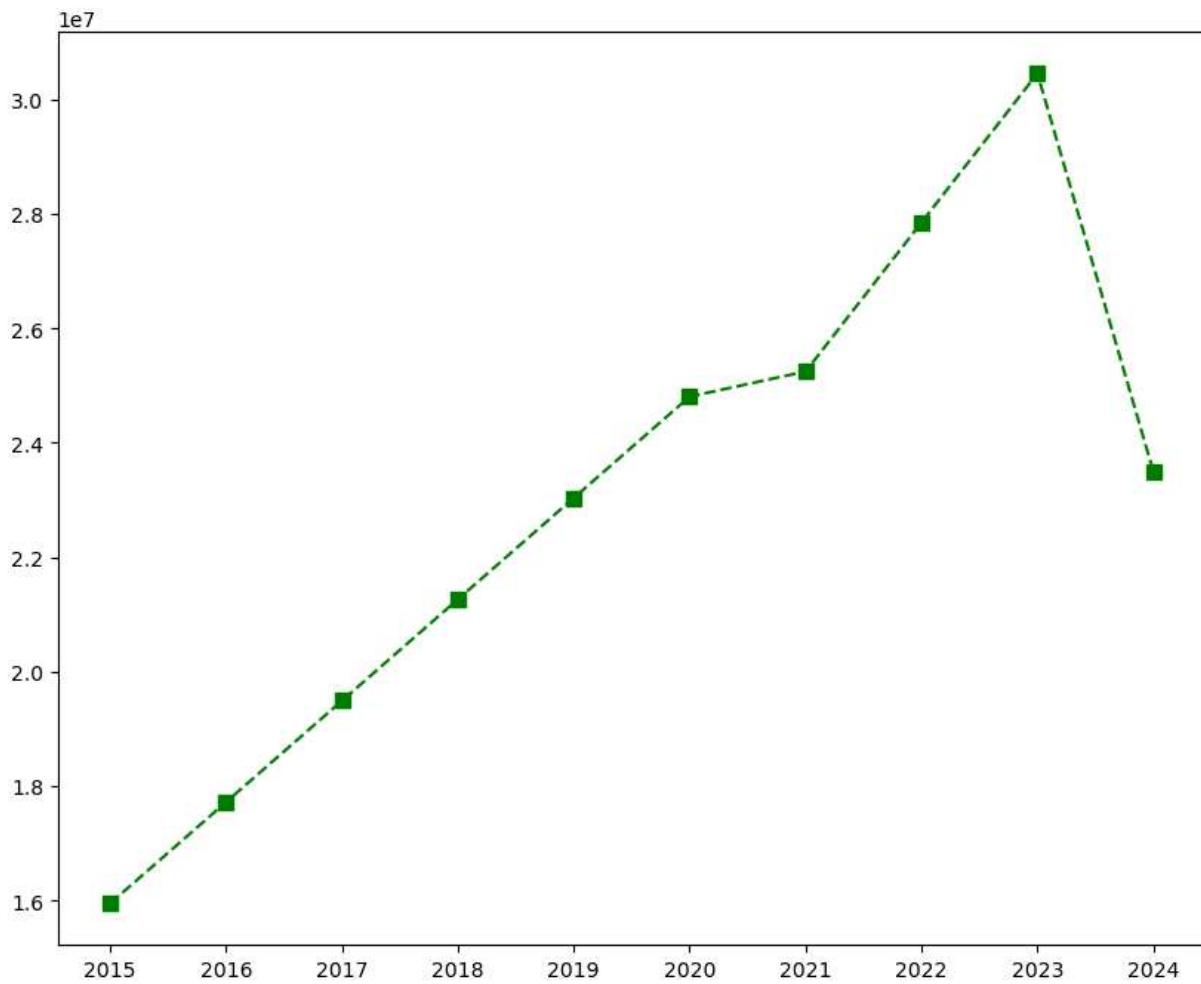
```
In [31]: plt.plot(Salary[0], c='Green', ls = ':', marker = 's', ms = 7, label = Players[0])
plt.xticks(list(range(0,10)), Seasons, rotation='vertical')
plt.show()
```



```
In [32]: Games
```

```
Out[32]: 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 [33]: 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 [34]: `Salary[0]`

Out[34]: `array([15946875, 17718750, 19490625, 21262500, 23034375, 24806250, 25244493, 27849149, 30453805, 23500000])`

In [35]: `Salary[1]`

Out[35]: `array([12000000, 12744189, 13488377, 14232567, 14976754, 16324500, 18038573, 19752645, 21466718, 23180790])`

In [41]: `plt.plot(Salary[1], c='Blue', ls=':', marker='o', ms=10, label=Players[1])`

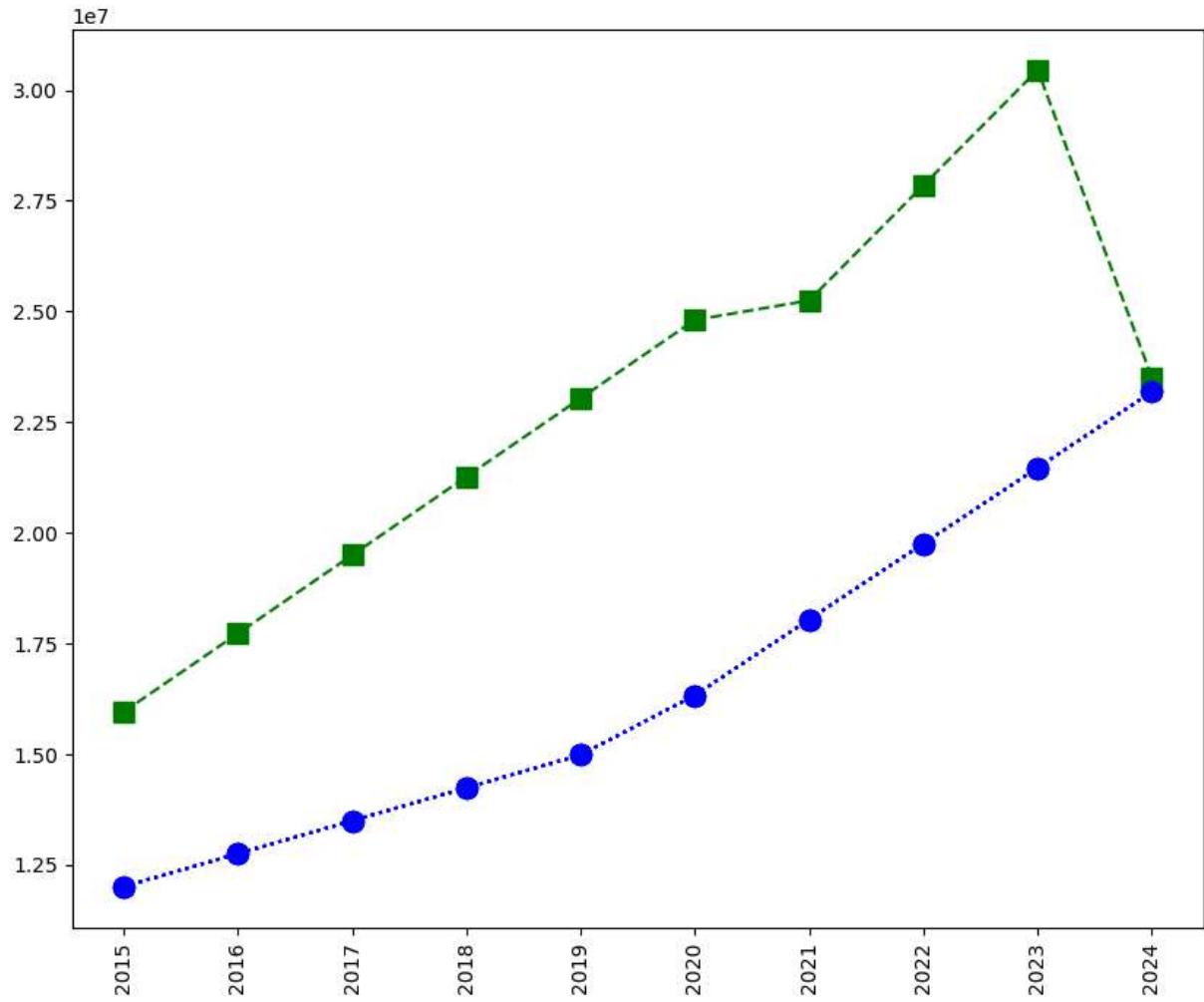
Out[41]: [`<matplotlib.lines.Line2D at 0x24c51ab7890>`]

More Visualization

```
In [42]: 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')

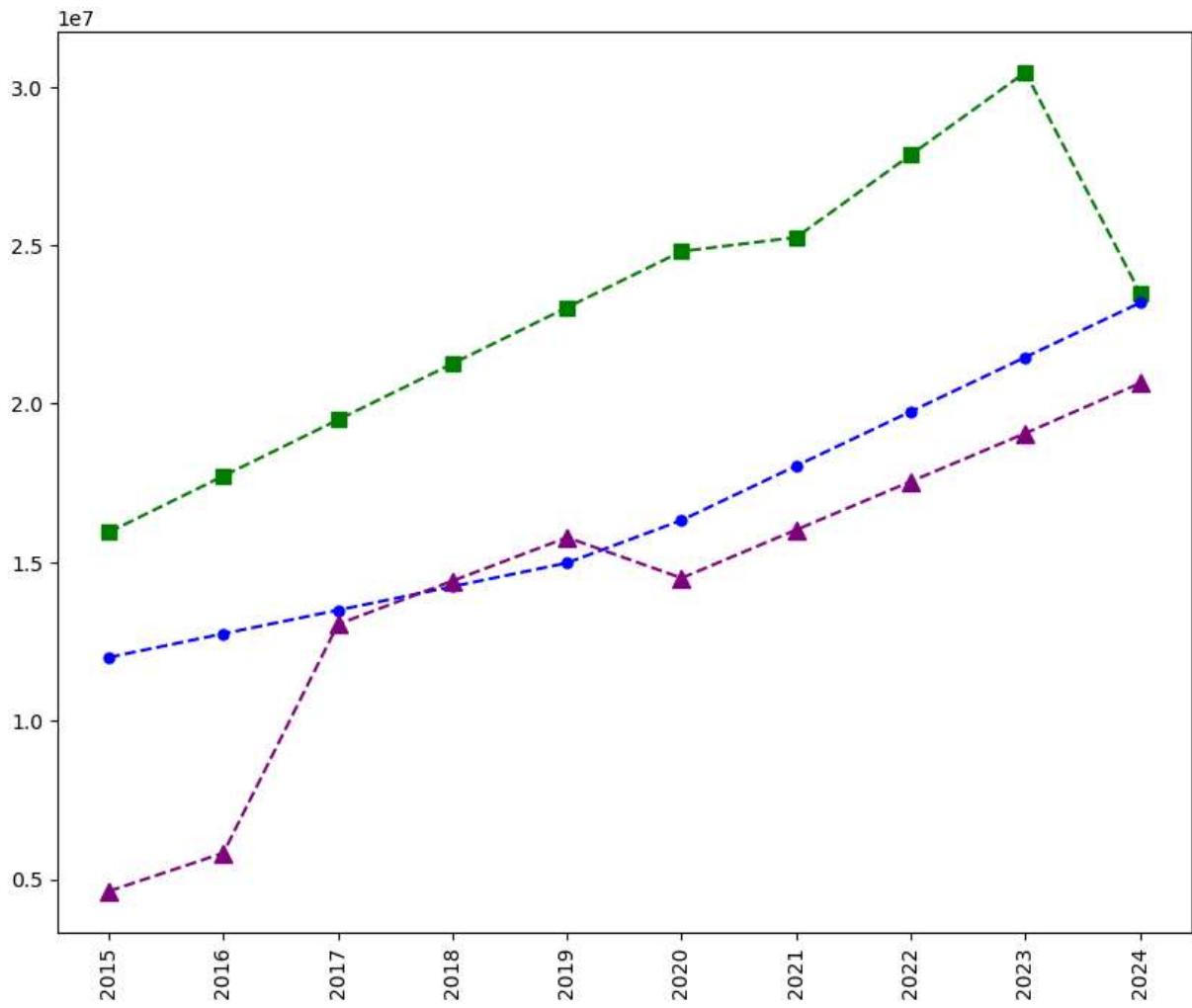
plt.show()
```



```
In [43]: 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')

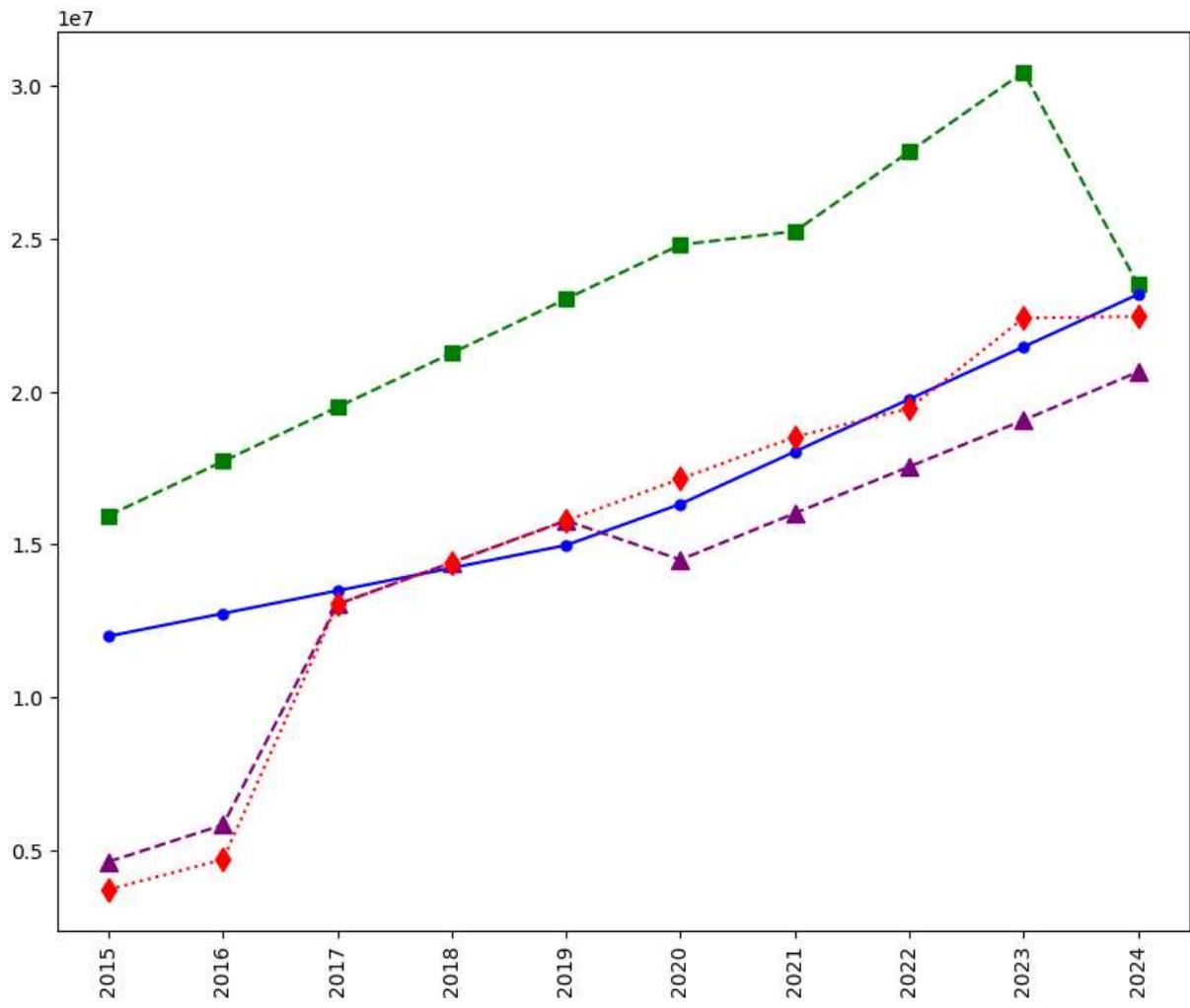
plt.show()
```



```
In [44]: 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')

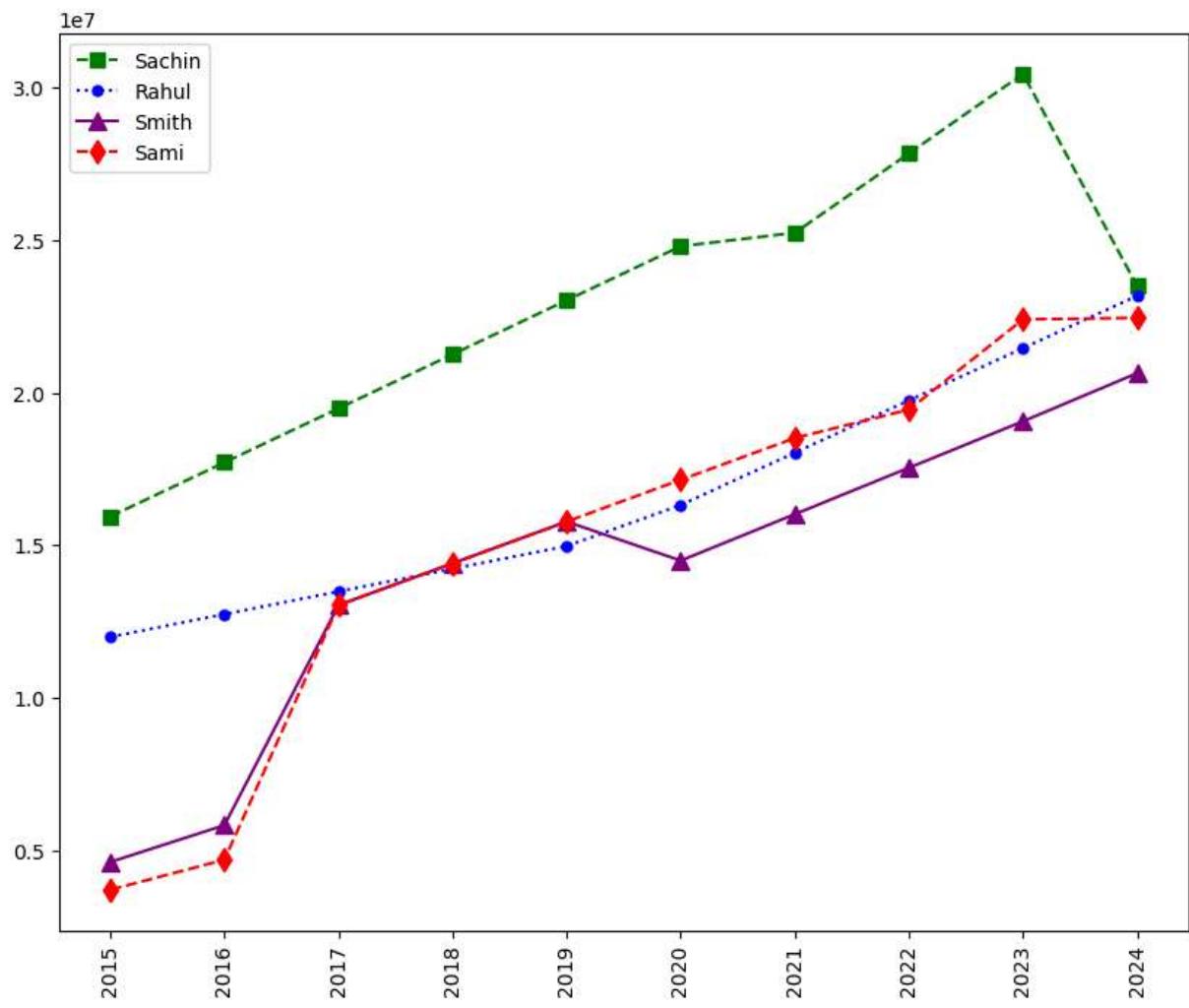
plt.show()
```



```
In [45]: # how to add Legned in visualisation
```

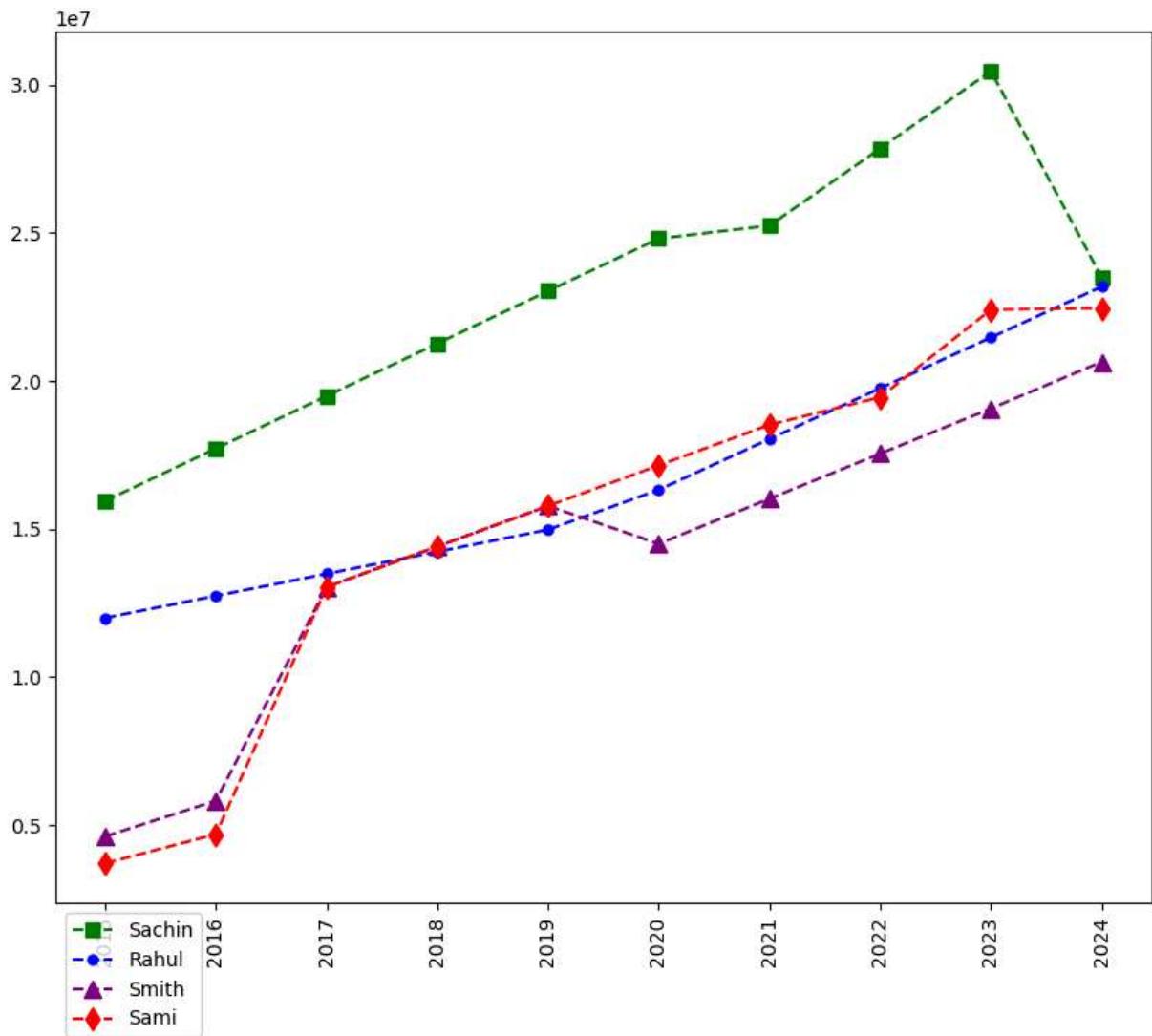
```
plt.plot(Salary[0], c='Green', ls = '--', marker = 's', ms = 7, label = Players[0])
plt.plot(Salary[1], c='Blue', ls = ':', marker = 'o', ms = 5, label = Players[1])
plt.plot(Salary[2], c='purple', ls = '-.', marker = '^', ms = 8, label = Players[2])
plt.plot(Salary[3], c='Red', ls = '--', marker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

plt.show()
```



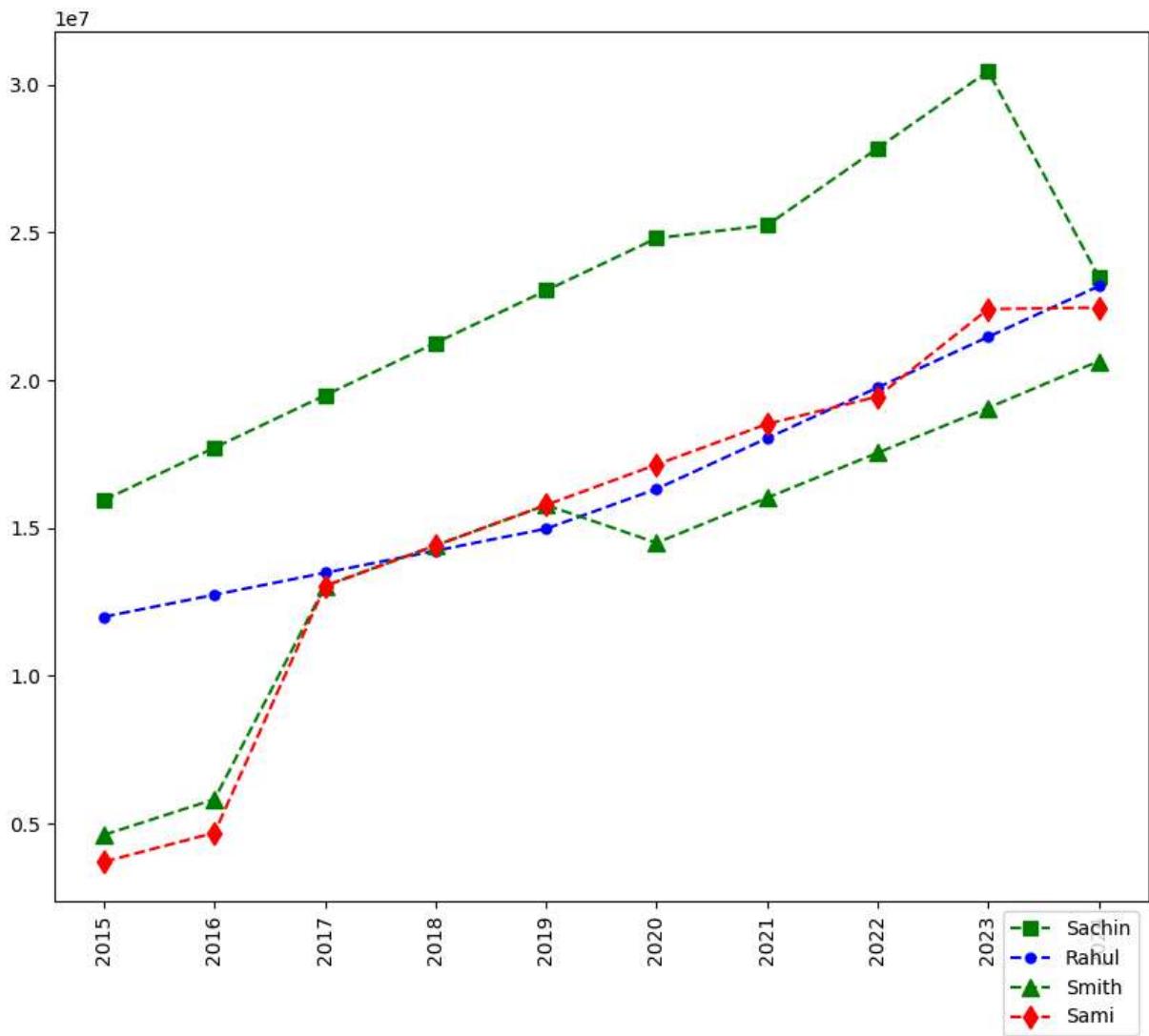
```
In [46]: plt.plot(Salary[0], c='Green', ls = '--', marker = 's', ms = 7, label = Players[0])
plt.plot(Salary[1], c='Blue', ls = '--', marker = 'o', ms = 5, label = Players[1])
plt.plot(Salary[2], c='purple', ls = '--', marker = '^', ms = 8, label = Players[2])
plt.plot(Salary[3], c='Red', ls = '--', 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 [47]: 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 [48]: # we can visualize the how many games played by a player

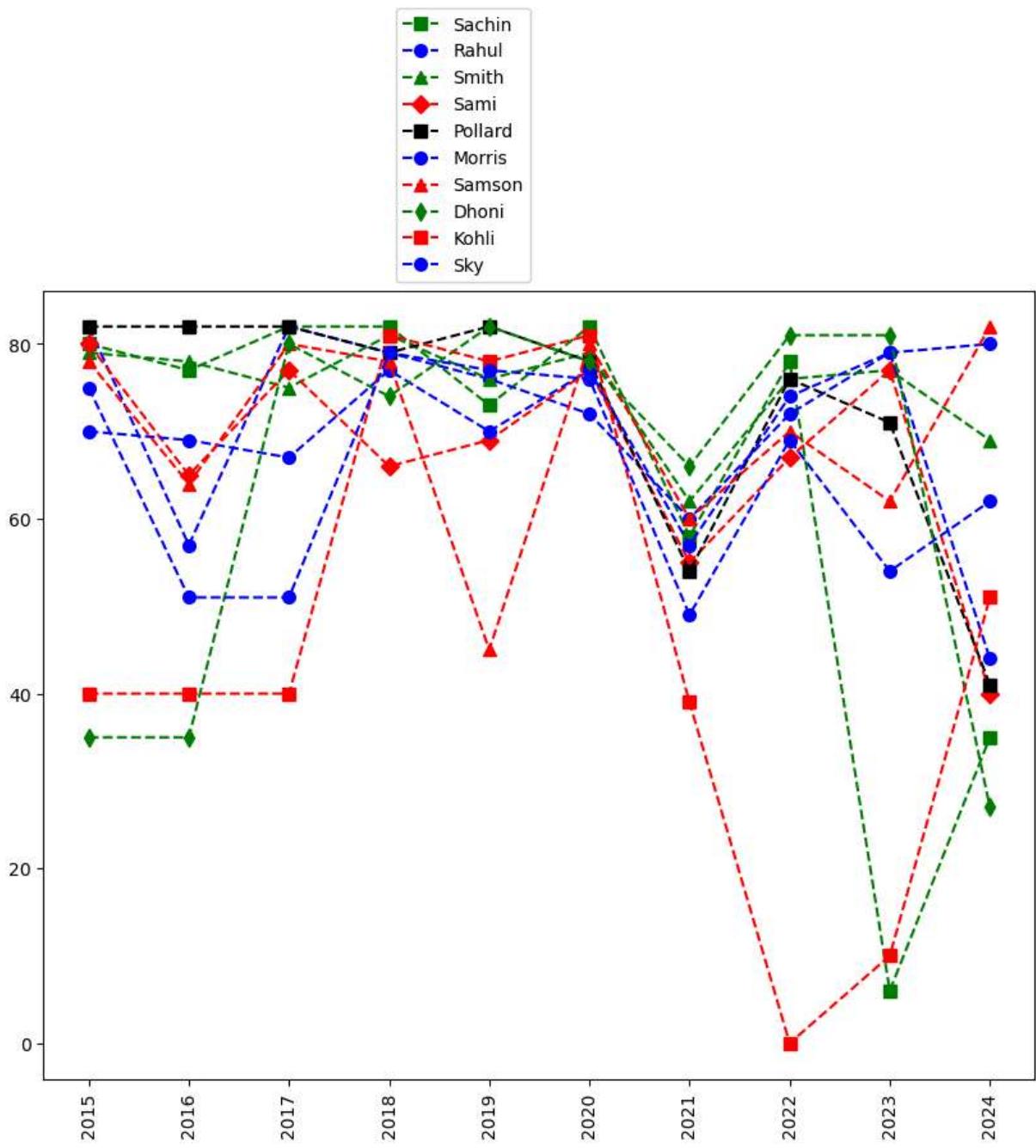
```

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 this section we learned -

- Matrices
- Building matrices - np.reshape
- Dictionary in python (order does not matter) (keys & values)
- Visualizing using pyplot
- Basketball analysis

In []: