

```
In [1]: #Import numpy
```

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

## Seasons

```
In [3]: Seasons = ["2015","2016","2017","2018","2019","2020","2021","2022","2023","2024"]
```

```
In [4]: Sdict = {"2015":0,"2016":1,"2017":2,"2018":3,"2019":4,"2020":5,"2021":6,"2022":7,"2
```

## Players

```
In [5]: Players = ["Sachin","Rahul","Smith","Sami","Pollard","Morris","Samson","Dhoni","Koh
```

```
In [6]: Pdict = {"Sachin":0,"Rahul":1,"Smith":2,"Sami":3,"Pollard":4,"Morris":5,"Samson":6,
```

## Salaries

```
In [7]: Sachin_Salary = [15946875,17718750,19490625,21262500,23034375,24806250,25244493,278  
Rahul_Salary = [12000000,12744189,13488377,14232567,14976754,16324500,18038573,1975  
Smith_Salary = [4621800,5828090,13041250,14410581,15779912,14500000,16022500,175450  
Sami_Salary = [3713640,4694041,13041250,14410581,15779912,17149243,18518574,1945000  
Pollard_Salary = [4493160,4806720,6061274,13758000,15202590,16647180,18091770,19536  
Morris_Salary = [3348000,4235220,12455000,14410581,15779912,14500000,16022500,17545  
Samson_Salary = [3144240,3380160,3615960,4574189,13520500,14940153,16359805,1777945  
Dhoni_Salary = [0,0,4171200,4484040,4796880,6053663,15506632,16669630,17832627,1899  
Kohli_Salary = [0,0,0,4822800,5184480,5546160,6993708,16402500,17632688,18862875]  
Sky_Salary = [3031920,3841443,13041250,14410581,15779912,14200000,15691000,17182000]
```

```
In [8]: # Matrix
```

```
In [9]: Salary = np.array([Sachin_Salary, Rahul_Salary, Smith_Salary, Sami_Salary, Pollard_
```

## Games

```
In [10]: Sachin_G = [80,77,82,82,73,82,58,78,6,35]  
Rahul_G = [82,57,82,79,76,72,60,72,79,80]  
Smith_G = [79,78,75,81,76,79,62,76,77,69]  
Sami_G = [80,65,77,66,69,77,55,67,77,40]  
Pollard_G = [82,82,82,79,82,78,54,76,71,41]  
Morris_G = [70,69,67,77,70,77,57,74,79,44]  
Samson_G = [78,64,80,78,45,80,60,70,62,82]  
Dhoni_G = [35,35,80,74,82,78,66,81,81,27]  
Kohli_G = [40,40,40,81,78,81,39,0,10,51]  
Sky_G = [75,51,51,79,77,76,49,69,54,62]
```

```
In [11]: Games = np.array([Sachin_G, Rahul_G, Smith_G, Sami_G, Pollard_G, Morris_G, Samson_G])
```

```
In [ ]:
```

## Points

```
In [12]: Sachin PTS = [2832, 2430, 2323, 2201, 1970, 2078, 1616, 2133, 83, 782]
Rahul PTS = [1653, 1426, 1779, 1688, 1619, 1312, 1129, 1170, 1245, 1154]
Smith PTS = [2478, 2132, 2250, 2304, 2258, 2111, 1683, 2036, 2089, 1743]
Sami PTS = [2122, 1881, 1978, 1504, 1943, 1970, 1245, 1920, 2112, 966]
Pollard PTS = [1292, 1443, 1695, 1624, 1503, 1784, 1113, 1296, 1297, 646]
Morris PTS = [1572, 1561, 1496, 1746, 1678, 1438, 1025, 1232, 1281, 928]
Samson PTS = [1258, 1104, 1684, 1781, 841, 1268, 1189, 1186, 1185, 1564]
Dhoni PTS = [903, 903, 1624, 1871, 2472, 2161, 1850, 2280, 2593, 686]
Kohli PTS = [597, 597, 597, 1361, 1619, 2026, 852, 0, 159, 904]
Sky PTS = [2040, 1397, 1254, 2386, 2045, 1941, 1082, 1463, 1028, 1331]
```

```
In [13]: # Matrix
```

```
In [14]: Points = np.array([Sachin PTS, Rahul PTS, Smith PTS, Sami PTS, Pollard PTS, Morris_
```

```
In [15]: Salary #matrix format
```

```
Out[15]: 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 [16]: # buliding your first matrix
Games
```

```
Out[16]: 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 [17]: Points
```

```
Out[17]: 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 [18]: 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 [19]: np.reshape(mydata,(4,5)) # 5 rows & 4 columns
```

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

```
In [20]: #np.reshape(mydata,(5,4), order = 'c') #'C' means to read / write the elements using column-major order  
MATR1 = np.reshape(mydata, (5,4), order = 'c')  
MATR1
```

```
Out[20]: array([[ 0,  1,  2,  3],  
   [ 4,  5,  6,  7],  
   [ 8,  9, 10, 11],  
   [12, 13, 14, 15],  
   [16, 17, 18, 19]])
```

```
In [21]: MATR1
```

```
Out[21]: array([[ 0,  1,  2,  3],  
   [ 4,  5,  6,  7],  
   [ 8,  9, 10, 11],  
   [12, 13, 14, 15],  
   [16, 17, 18, 19]])
```

```
In [22]: # If i want to get only no.3  
MATR1[4,3]
```

```
Out[22]: 19
```

```
In [23]: MATR1[3,3]
```

```
Out[23]: 15
```

```
In [24]: MATR1
```

```
Out[24]: array([[ 0,  1,  2,  3],
                [ 4,  5,  6,  7],
                [ 8,  9, 10, 11],
                [12, 13, 14, 15],
                [16, 17, 18, 19]])
```

```
In [25]: MATR1[-3,-1]
```

```
Out[25]: 11
```

```
In [26]: MATR1
```

```
Out[26]: array([[ 0,  1,  2,  3],
                [ 4,  5,  6,  7],
                [ 8,  9, 10, 11],
                [12, 13, 14, 15],
                [16, 17, 18, 19]])
```

```
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]: MATR2 = np.reshape(mydata, (5,4), order = 'F') # reshape behaviour are - 'C','F',''
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[4,3]
```

```
Out[29]: 19
```

```
In [30]: MATR2[0,2]
```

```
Out[30]: 10
```

```
In [31]: MATR2[0:2]
```

```
Out[31]: array([[ 0,  5, 10, 15],
                [ 1,  6, 11, 16]])
```

```
In [32]: MATR2
```

```
Out[32]: array([[ 0,  5, 10, 15],  
                 [ 1,  6, 11, 16],  
                 [ 2,  7, 12, 17],  
                 [ 3,  8, 13, 18],  
                 [ 4,  9, 14, 19]])
```

```
In [33]: MATR2[1:2]
```

```
Out[33]: array([[ 1,  6, 11, 16]])
```

```
In [34]: MATR2[1,2]
```

```
Out[34]: 11
```

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

```
Out[36]: 18
```

```
In [37]: MATR2[-1,1]
```

```
Out[37]: 9
```

```
In [38]: MATR2[0:2]
```

```
Out[38]: array([[ 0,  5, 10, 15],  
                 [ 1,  6, 11, 16]])
```

```
In [39]: MATR3 = np.reshape(mydata, (5,4), order = 'A')  
MATR3
```

```
Out[39]: array([[ 0,  1,  2,  3],  
                 [ 4,  5,  6,  7],  
                 [ 8,  9, 10, 11],  
                 [12, 13, 14, 15],  
                 [16, 17, 18, 19]])
```

```
In [40]: MATR2 ## F shaped
```

```
Out[40]: array([[ 0,  5, 10, 15],  
                 [ 1,  6, 11, 16],  
                 [ 2,  7, 12, 17],  
                 [ 3,  8, 13, 18],  
                 [ 4,  9, 14, 19]])
```

```
In [41]: MATR1 # C shaped
```

```
Out[41]: array([[ 0,  1,  2,  3],
   [ 4,  5,  6,  7],
   [ 8,  9, 10, 11],
   [12, 13, 14, 15],
   [16, 17, 18, 19]])
```

```
In [42]: a1 = ['welcome', 'to', 'datascience']
a2 = ['required', 'hard', 'work' ]
a3 = [1,2,3]
```

```
In [43]: [a1,a2,a3] # List same datatype
```

```
Out[43]: [['welcome', 'to', 'datascience'], ['required', 'hard', 'work'], [1, 2, 3]]
```

```
In [44]: np.array([a1,a2,a3]) # u11 - unicode 11 characer : 3*3 matrix
```

```
Out[44]: array([['welcome', 'to', 'datascience'],
   ['required', 'hard', 'work'],
   ['1', '2', '3']], dtype='<U11')
```

```
In [45]: Games
```

```
Out[45]: 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 [46]: Games[1]
```

```
Out[46]: array([82, 57, 82, 79, 76, 72, 60, 72, 79, 80])
```

```
In [47]: Games[0:3]
```

```
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]])
```

```
In [48]: Games[-5]
```

```
Out[48]: array([70, 69, 67, 77, 70, 77, 57, 74, 79, 44])
```

```
In [49]: Games[-1:-3]
```

```
Out[49]: array([], shape=(0, 10), dtype=int32)
```

```
In [50]: Games[0,2]
```

```
Out[50]: 82
```

```
In [51]: Games[2,8]
```

```
Out[51]: 77
```

```
In [52]: Games[-2:-5]
```

```
Out[52]: array([], shape=(0, 10), dtype=int32)
```

```
In [53]: Points
```

```
Out[53]: 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 [54]: Points[0]
```

```
Out[54]: array([2832, 2430, 2323, 2201, 1970, 2078, 1616, 2133, 83, 782])
```

```
In [55]: Points[6,1]
```

```
Out[55]: 1104
```

```
In [56]: Points[-3,-2]
```

```
Out[56]: 2593
```

```
In [57]: ===== DICTIONARY =====
```

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

```
In [58]: dict1
```

```
Out[58]: {'key1': 'val1', 'key2': 'val2', 'key3': 'val3'}
```

```
In [59]: dict1['key2']
```

```
Out[59]: 'val2'
```

```
In [60]: dict2 = {'bang':2, 'hyd':'we are hear', 'pune':True}  
dict2
```

```
Out[60]: {'bang': 2, 'hyd': 'we are hear', 'pune': True}
```

```
In [61]: dict3 = {'Germany': 'I have been here', 'France': 2, 'Spain': True}
dict3
```

```
Out[61]: {'Germany': 'I have been here', 'France': 2, 'Spain': True}
```

```
In [62]: dict3['Germany']
```

```
Out[62]: 'I have been here'
```

```
In [63]: # if you check cheat 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 dont required to count which no row which
```

```
In [64]: Games
```

```
Out[64]: 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 [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]: # how do i know player kobe bryant is at
```

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

```
Out[66]: 0
```

```
In [67]: Games[0]
```

```
Out[67]: array([80, 77, 82, 82, 73, 82, 58, 78, 6, 35])
```

```
In [68]: Pdict['Rahul']
```

```
Out[68]: 1
```

```
In [69]: Games[1]
```

```
Out[69]: array([82, 57, 82, 79, 76, 72, 60, 72, 79, 80])
```

```
In [70]: Games[Pdict['Rahul']]
```

```
Out[70]: array([82, 57, 82, 79, 76, 72, 60, 72, 79, 80])
```

```
In [71]: Points
```

```
Out[71]: 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 [72]: Salary
```

```
Out[72]: 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 [73]: Salary[2,4]
```

```
Out[73]: 15779912
```

```
In [74]: Salary[Pdict['Sky']][Sdict['2019']]
```

```
Out[74]: 15779912
```

```
In [75]: Salary/Games
```

```
C:\Users\vishnu\AppData\Local\Temp\ipykernel_12860\3709746658.py:1: RuntimeWarning:  
divide by zero encountered in divide  
    Salary/Games
```

```
Out[75]: 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 [76]: np.round(Salary/Games)
```

```
C:\Users\vishnu\AppData\Local\Temp\ipykernel_12860\3232172828.py:1: RuntimeWarning:  
divide by zero encountered in divide  
np.round(Salary/Games)
```

```
Out[76]: 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 [77]: import warnings
warnings.filterwarnings('ignore')
```

```
In [78]: import matplotlib.pyplot as plt
```

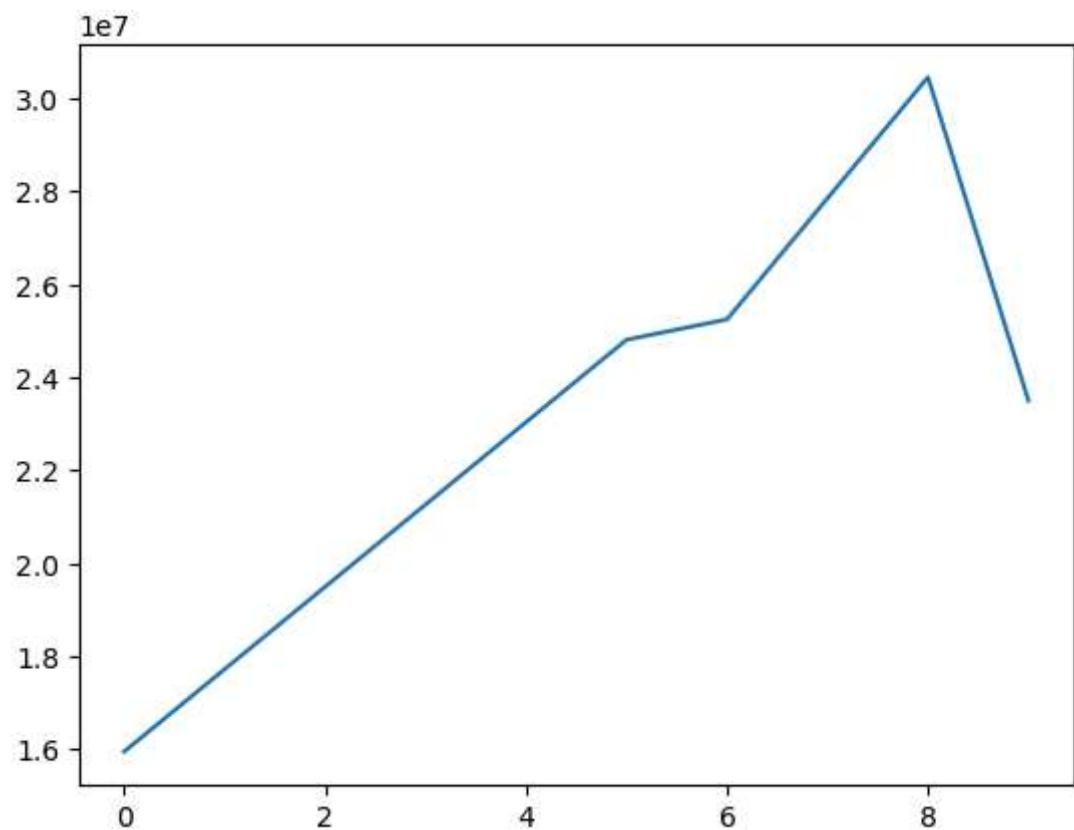
```
In [79]: matplotlib inline
```

```
In [80]: Salary[0]
```

```
Out[80]: array([15946875, 17718750, 19490625, 21262500, 23034375, 24806250,
   25244493, 27849149, 30453805, 23500000])
```

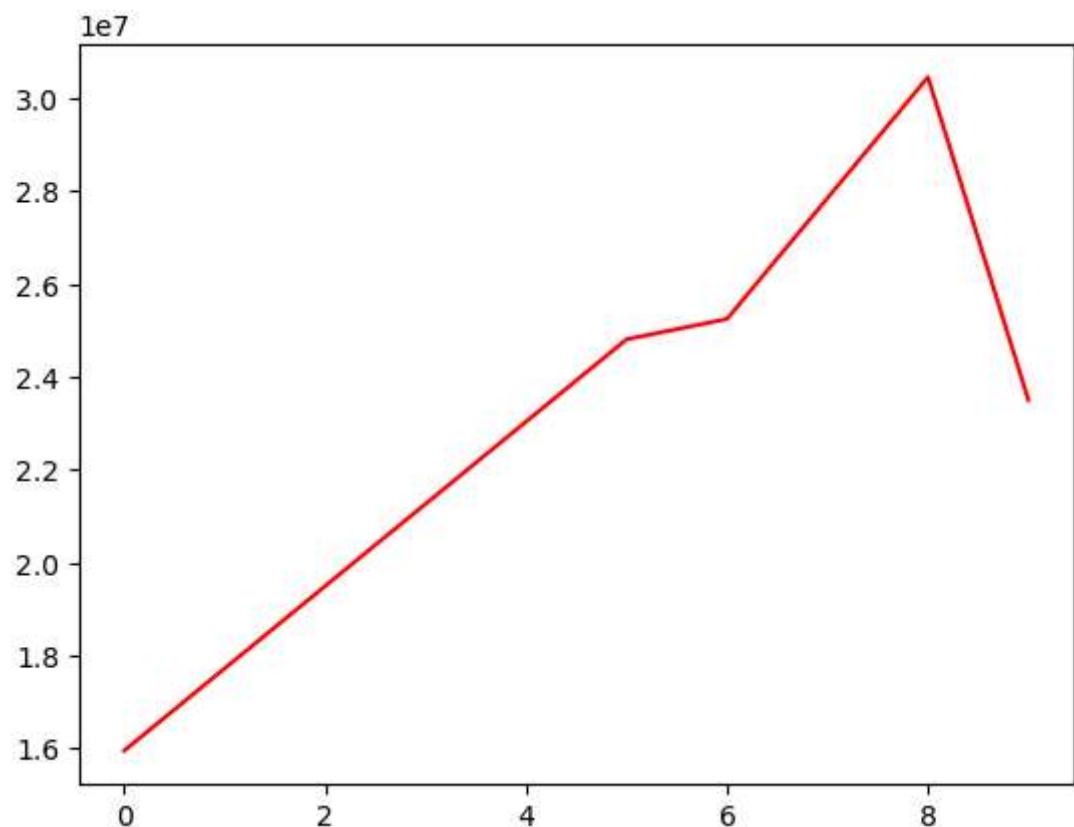
```
In [81]: plt.plot(Salary[0])
```

```
Out[81]: [<matplotlib.lines.Line2D at 0x18d92a13110>]
```



```
In [82]: plt.plot(Salary[0], c='red')
```

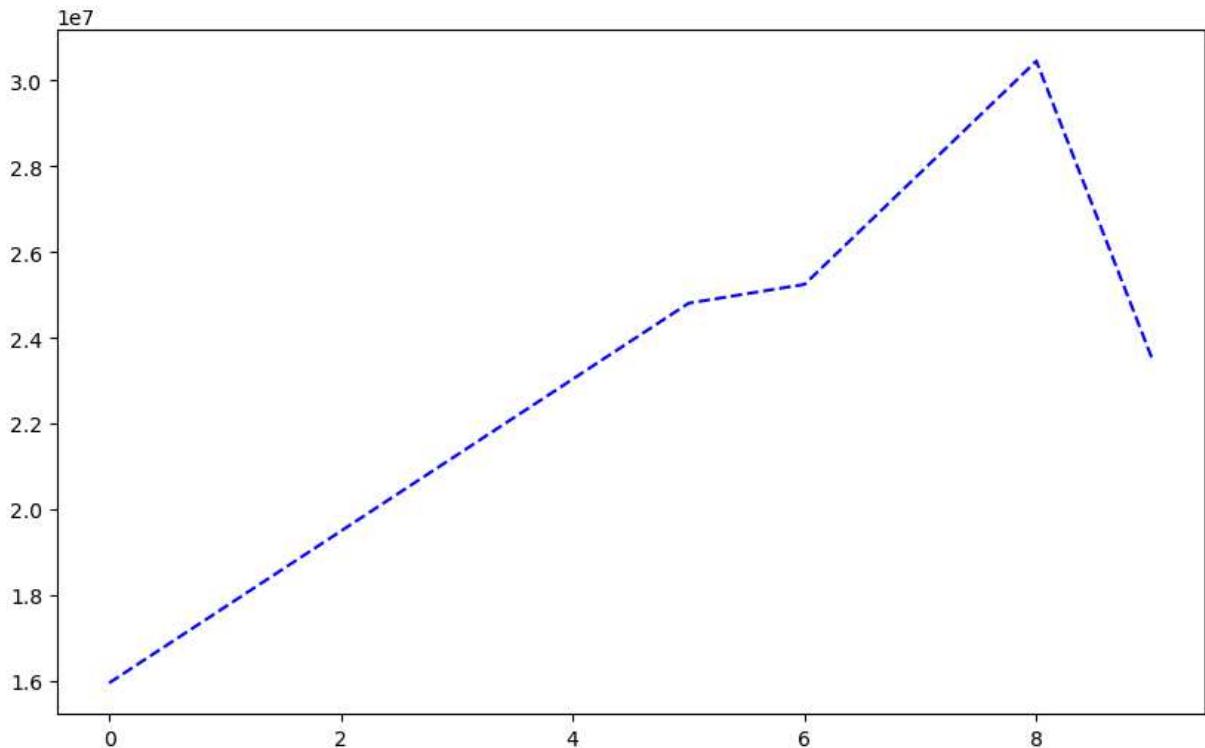
```
Out[82]: [<matplotlib.lines.Line2D at 0x18d9335c4d0>]
```



```
In [83]: %matplotlib inline  
plt.rcParams['figure.figsize'] = 10,6
```

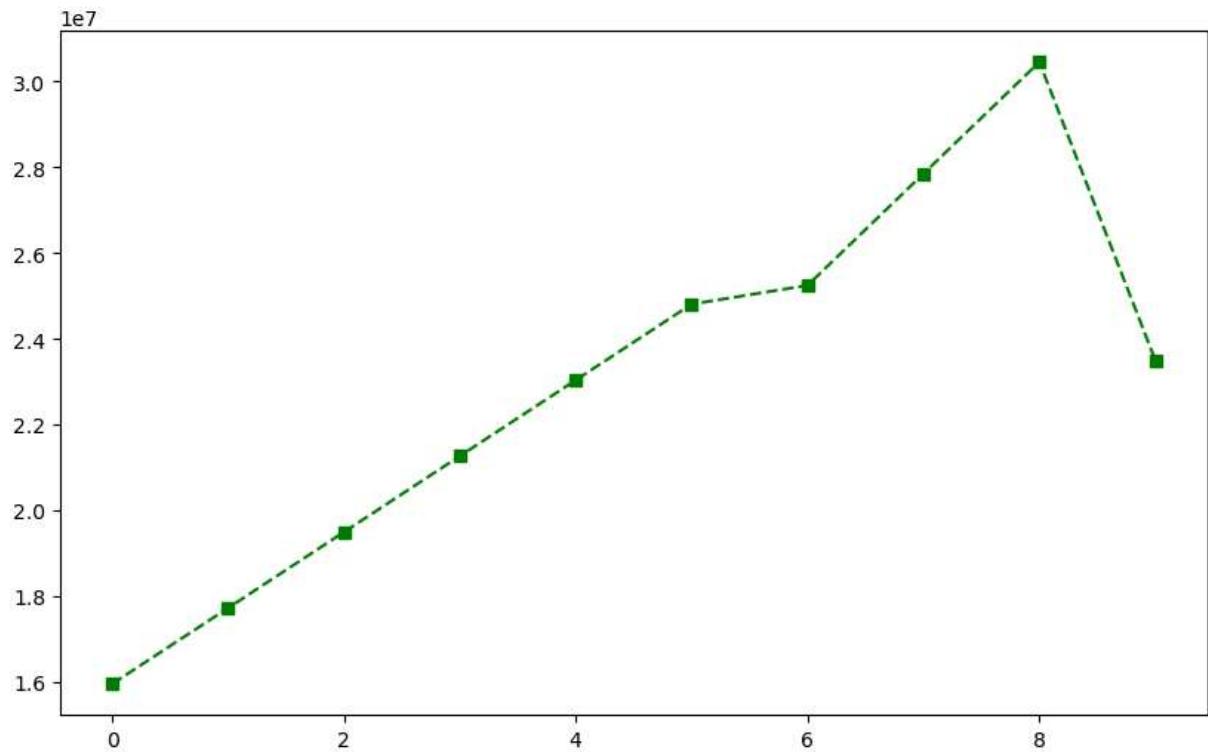
```
In [84]: plt.plot(Salary[0], c='Blue', ls = 'dashed')
```

```
Out[84]: [<matplotlib.lines.Line2D at 0x18d9335da30>]
```



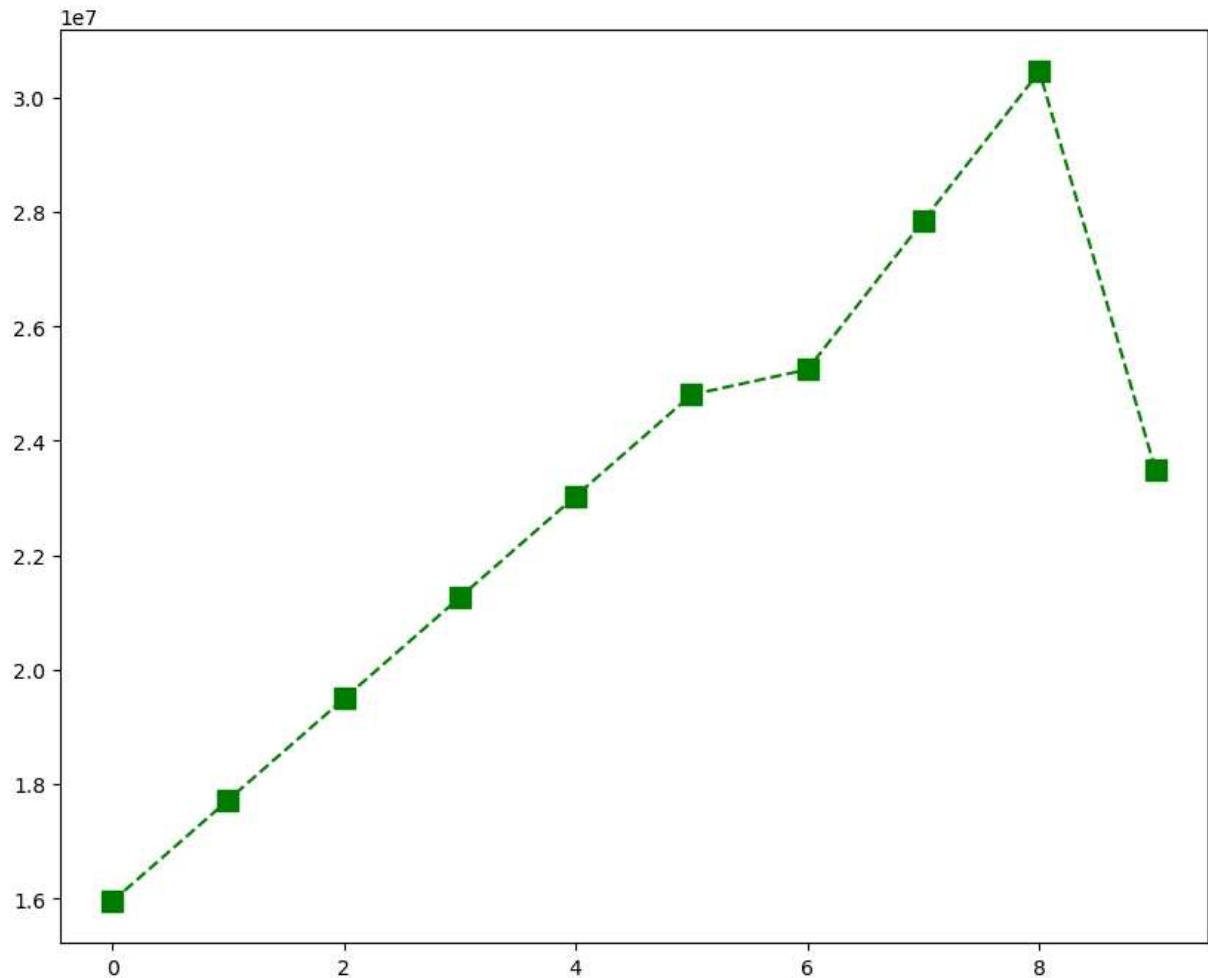
```
In [85]: plt.plot(Salary[0], c='Green', ls = '--', marker = 's') # s - squares
```

```
Out[85]: [<matplotlib.lines.Line2D at 0x18d92b3b8f0>]
```



```
In [86]: %matplotlib inline  
plt.rcParams['figure.figsize'] = 10,8 #runtime configuration parameter
```

```
In [87]: plt.plot(Salary[0],c='Green',ls='--',marker='s',ms=10)  
plt.show()
```



```
In [88]: list(range(0,10))
```

```
Out[88]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

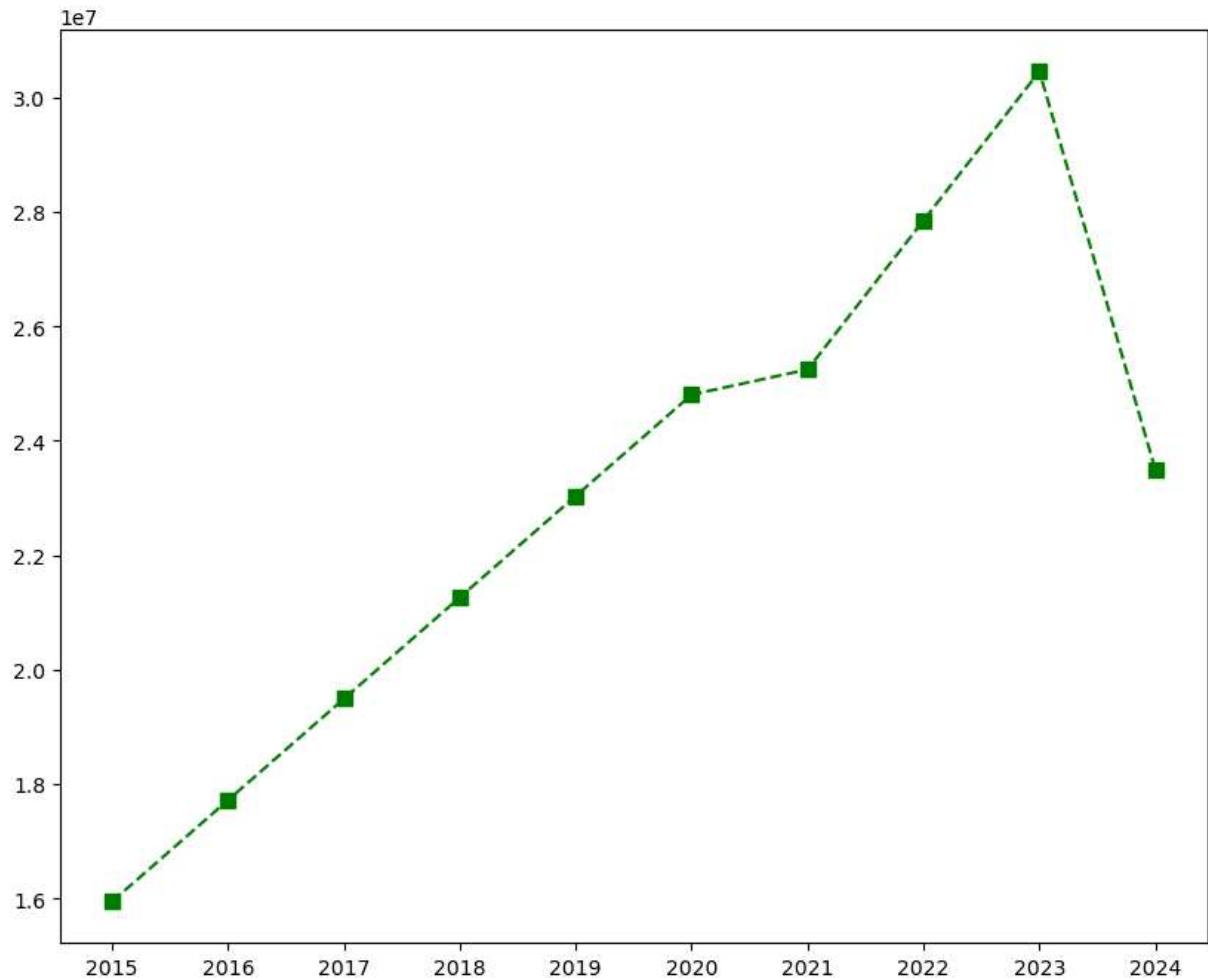
```
In [89]: Sdict
```

```
Out[89]: {'2015': 0,
           '2016': 1,
           '2017': 2,
           '2018': 3,
           '2019': 4,
           '2020': 5,
           '2021': 6,
           '2022': 7,
           '2023': 8,
           '2024': 9}
```

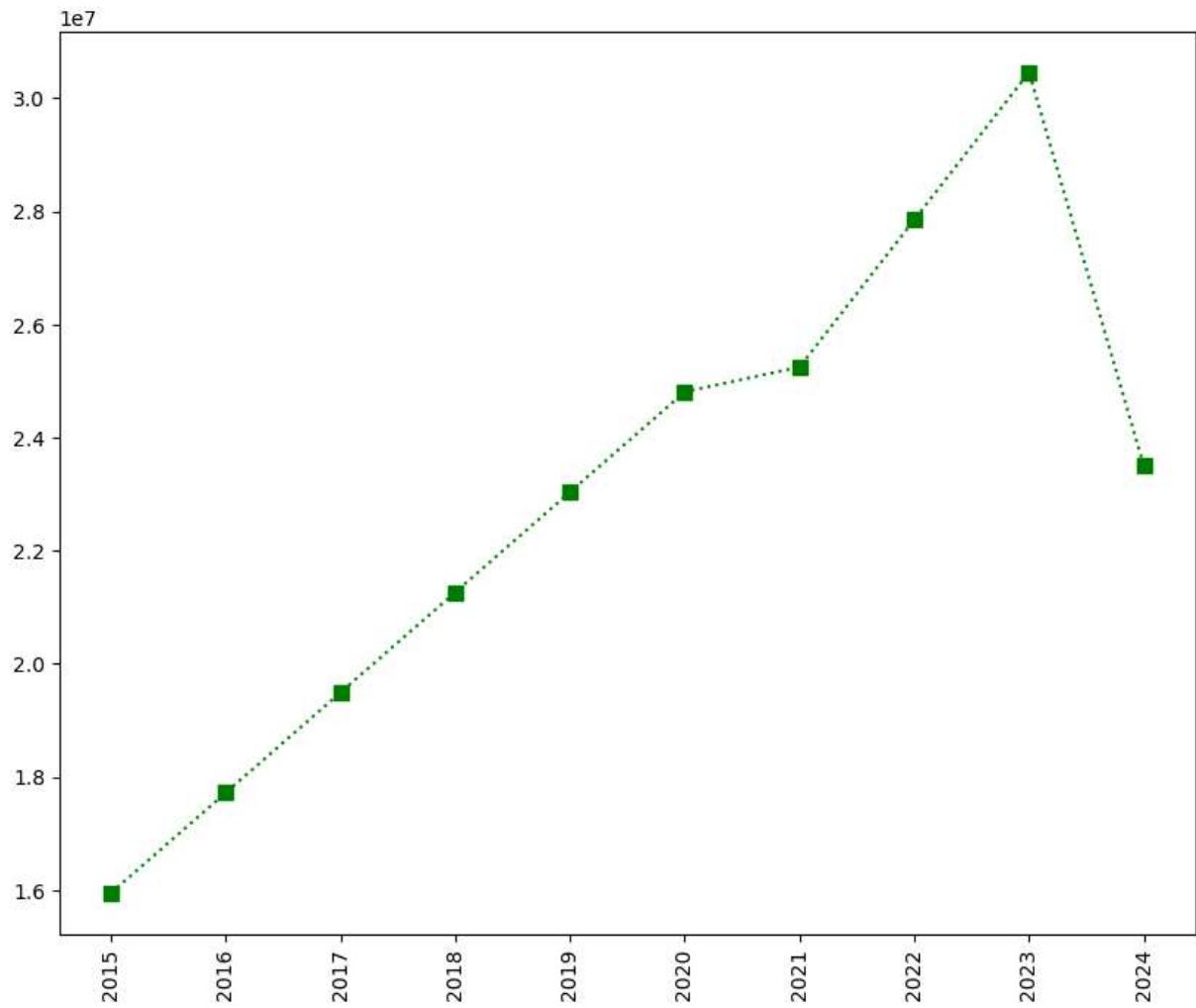
```
In [90]: Pdict
```

```
Out[90]: {'Sachin': 0,  
          'Rahul': 1,  
          'Smith': 2,  
          'Sami': 3,  
          'Pollard': 4,  
          'Morris': 5,  
          'Samson': 6,  
          'Dhoni': 7,  
          'Kohli': 8,  
          'Sky': 9}
```

```
In [91]: plt.plot(Salary[0], c='Green', ls='--', marker='s', ms=7)  
plt.xticks(list(range(0,10)),Seasons)  
plt.show()
```



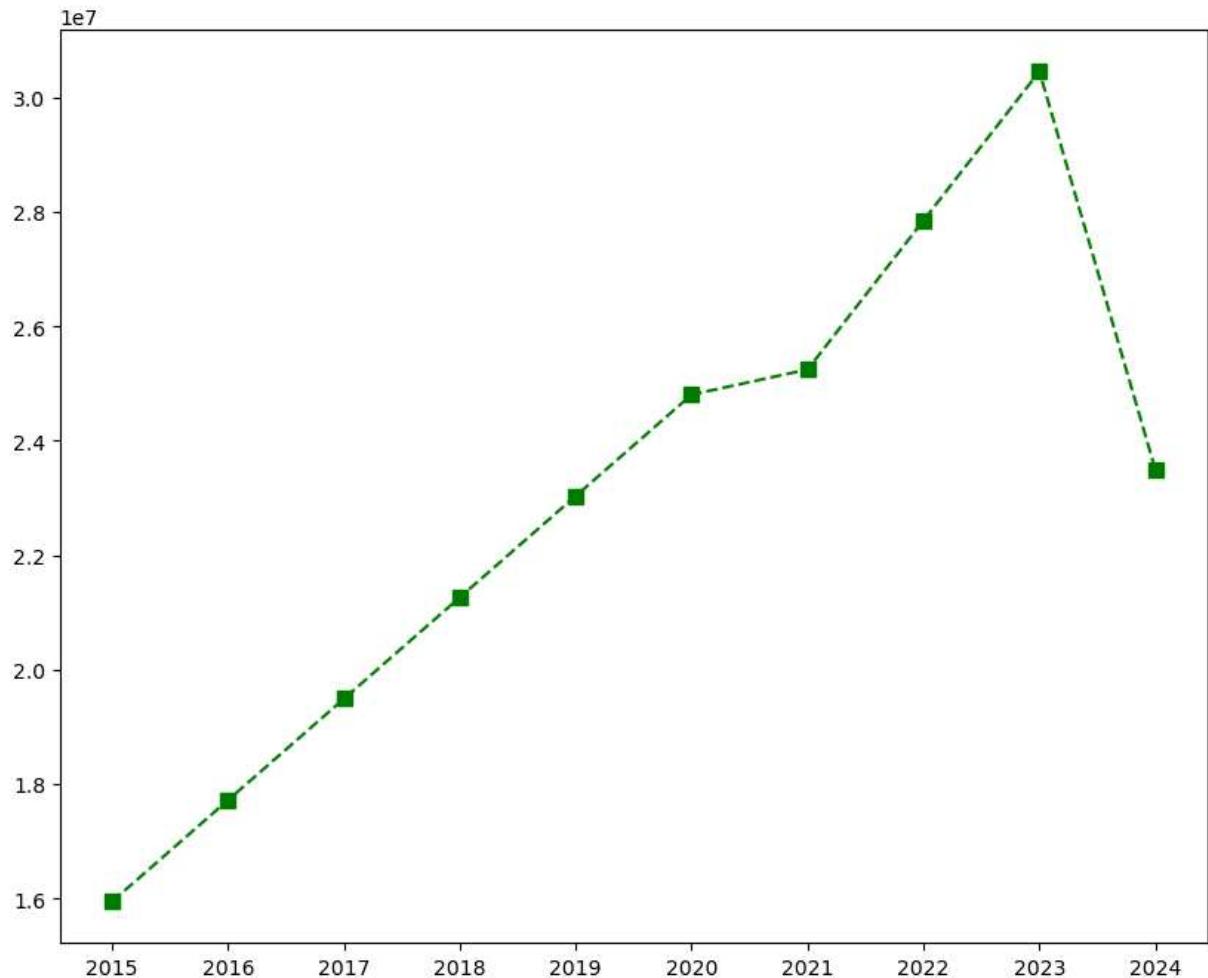
```
In [92]: plt.plot(Salary[0], c='Green', ls = ':', marker = 's', ms = 7, label = Players[0])  
plt.xticks(list(range(0,10)), Seasons, rotation='vertical')  
plt.show()
```



```
In [93]: Games
```

```
Out[93]: 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 [94]: 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 [95]: Salary[0]
```

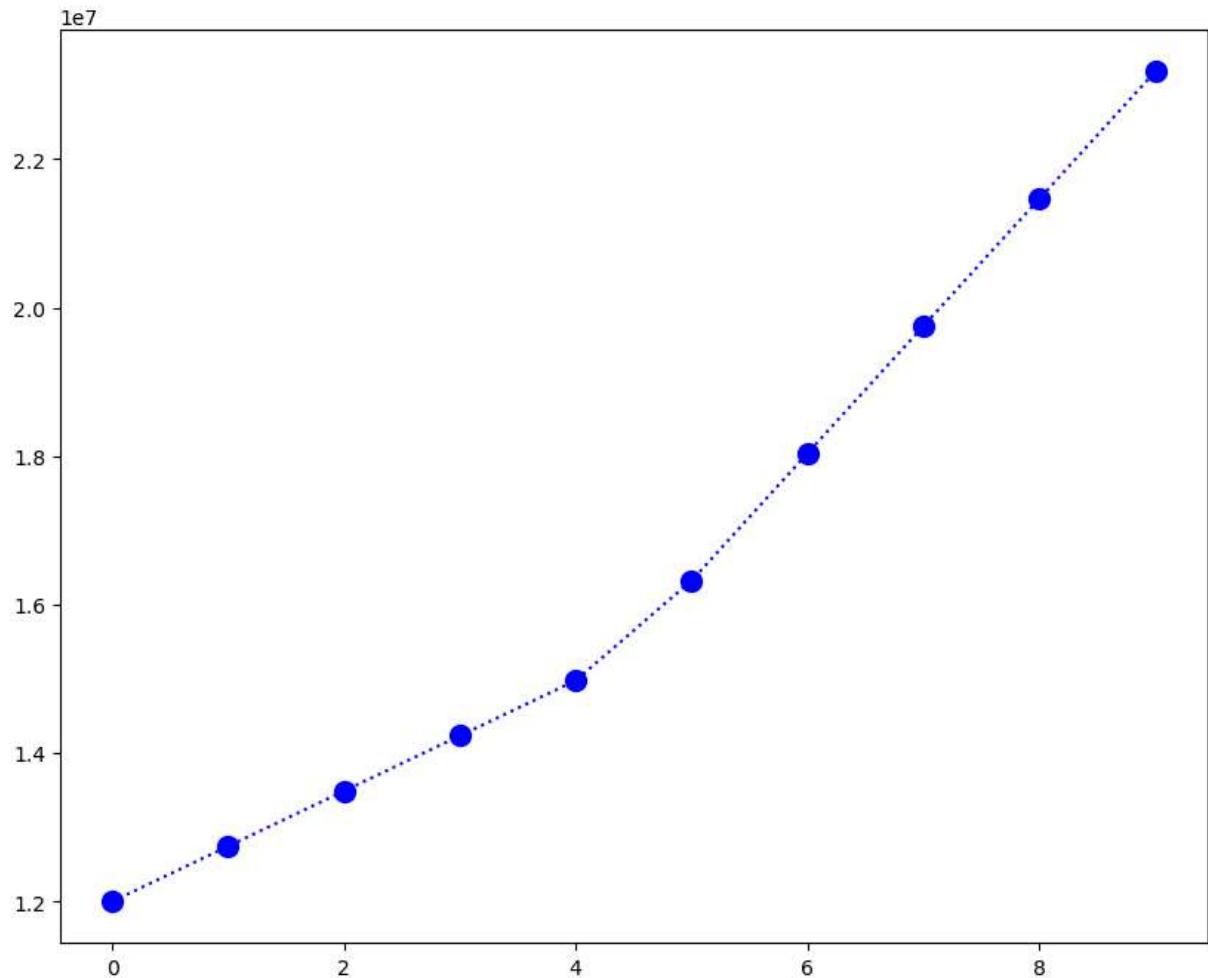
```
Out[95]: array([15946875, 17718750, 19490625, 21262500, 23034375, 24806250,
 25244493, 27849149, 30453805, 23500000])
```

```
In [96]: Salary[1]
```

```
Out[96]: array([12000000, 12744189, 13488377, 14232567, 14976754, 16324500,
 18038573, 19752645, 21466718, 23180790])
```

```
In [97]: plt.plot(Salary[1], c='Blue', ls = ':', marker = 'o', ms = 10, label = Players[1])
```

```
Out[97]: [
```

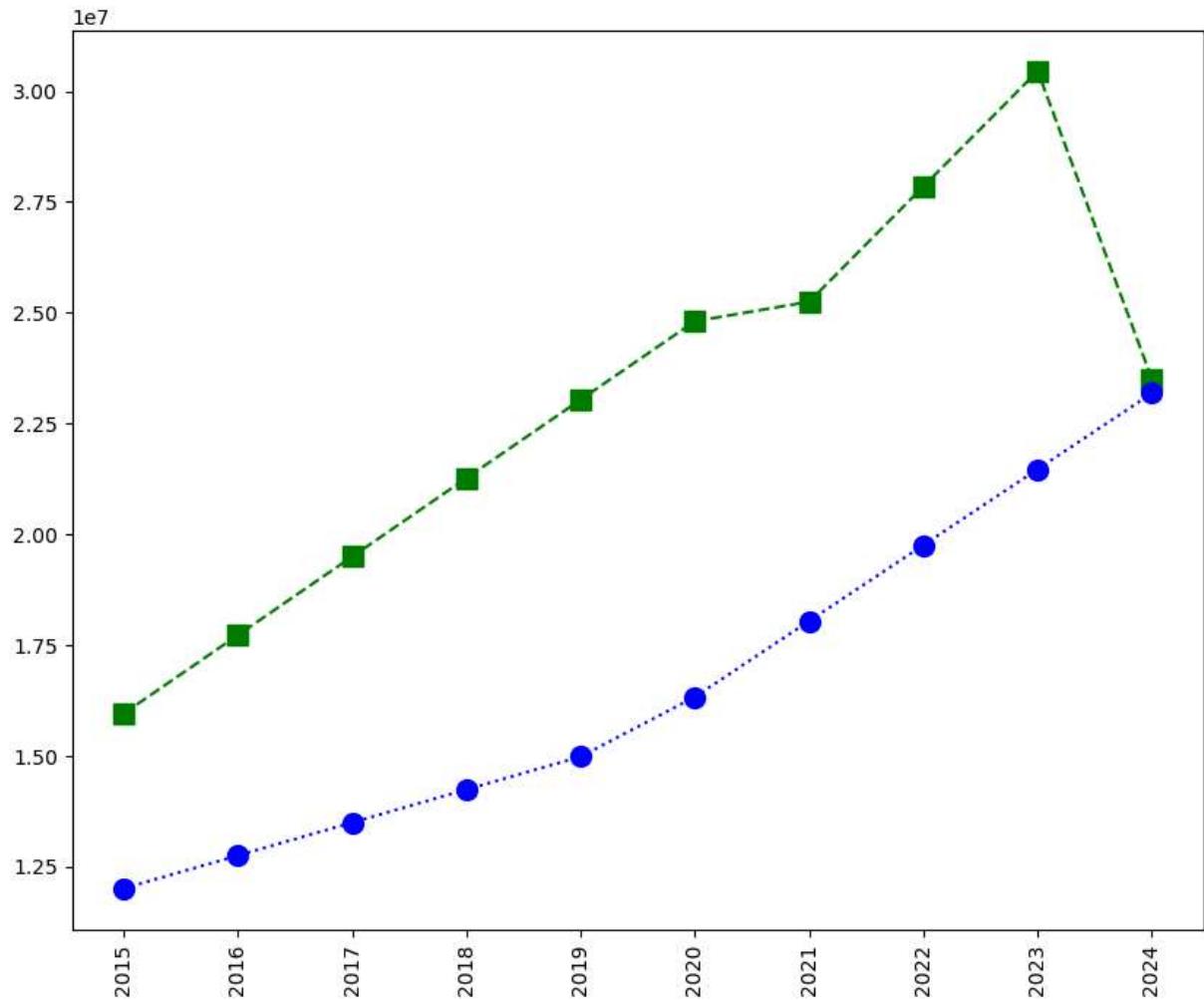


```
In [98]: # More Visualisation
```

```
In [99]: 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')

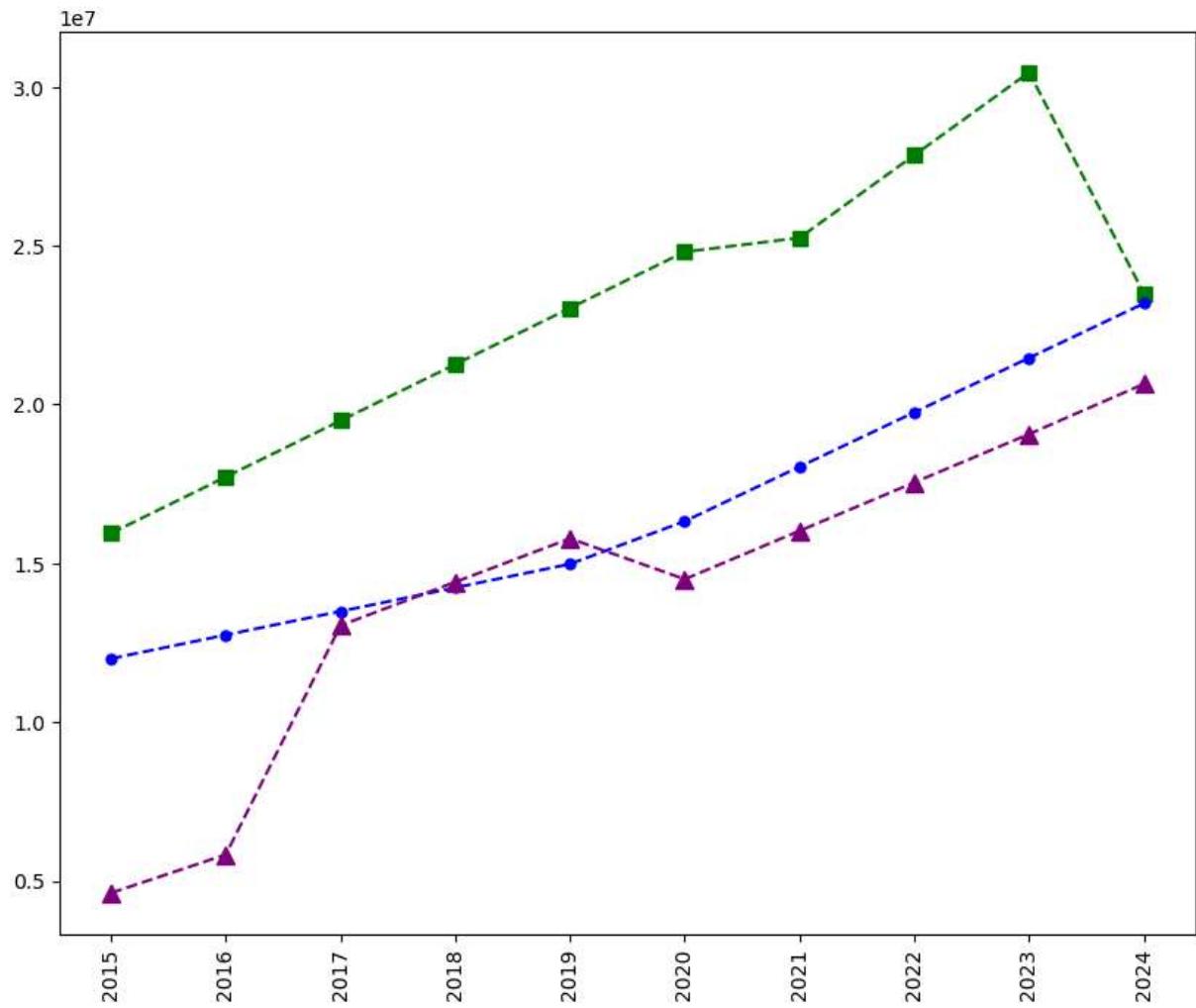
plt.show()
```



```
In [100]: 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')

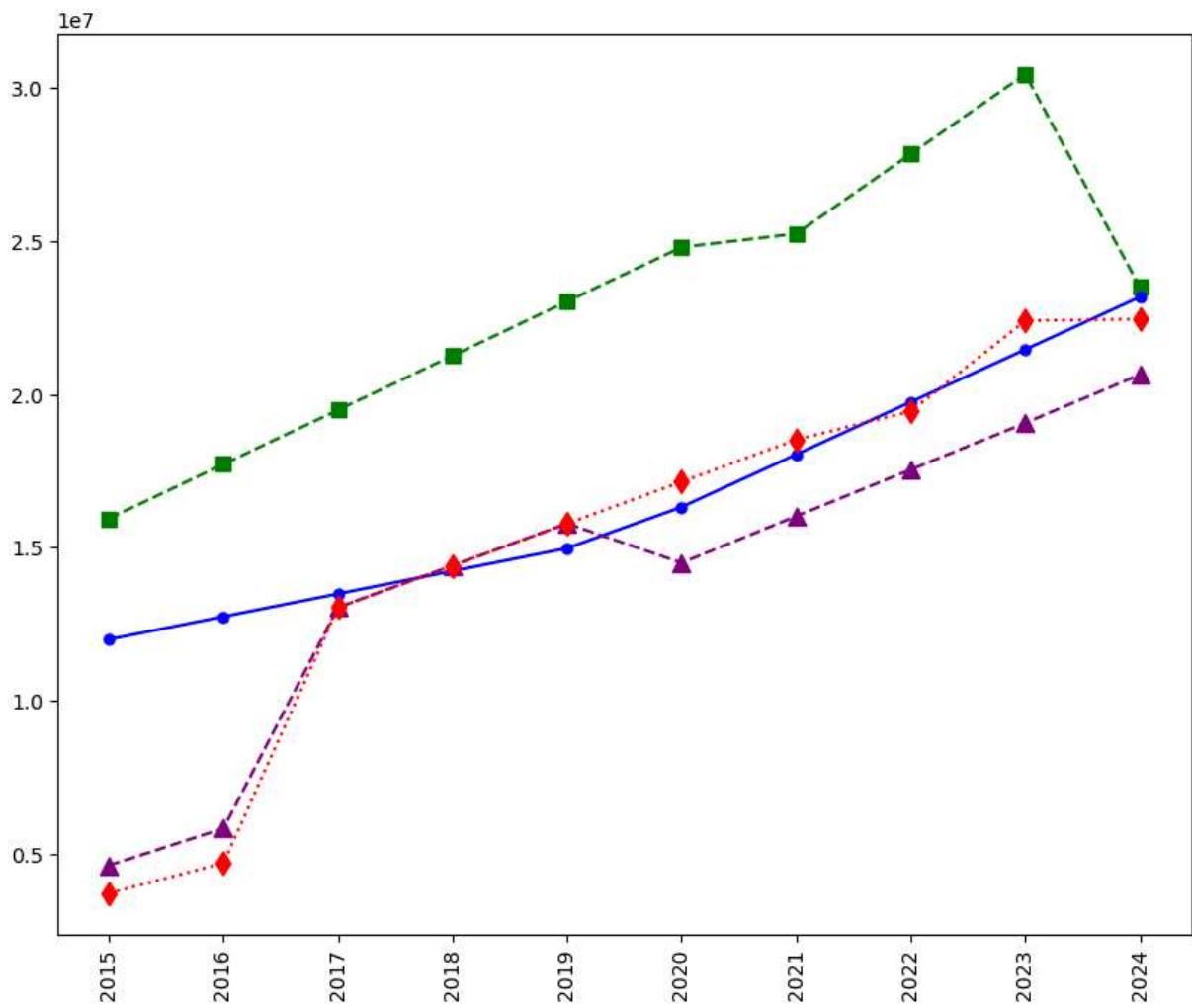
plt.show()
```



```
In [101]: 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()
```

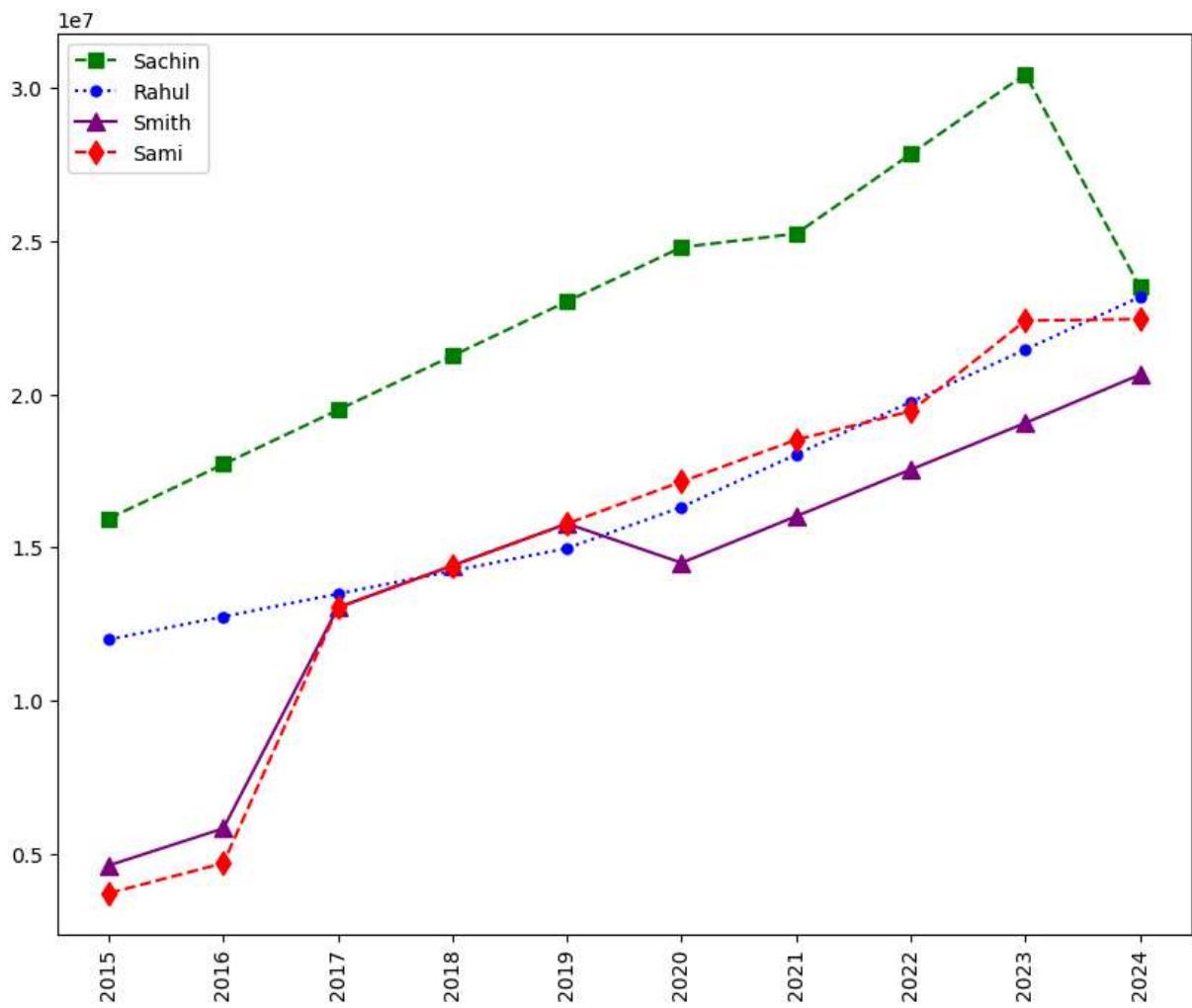


In [102]:

```
# how to add legend 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')

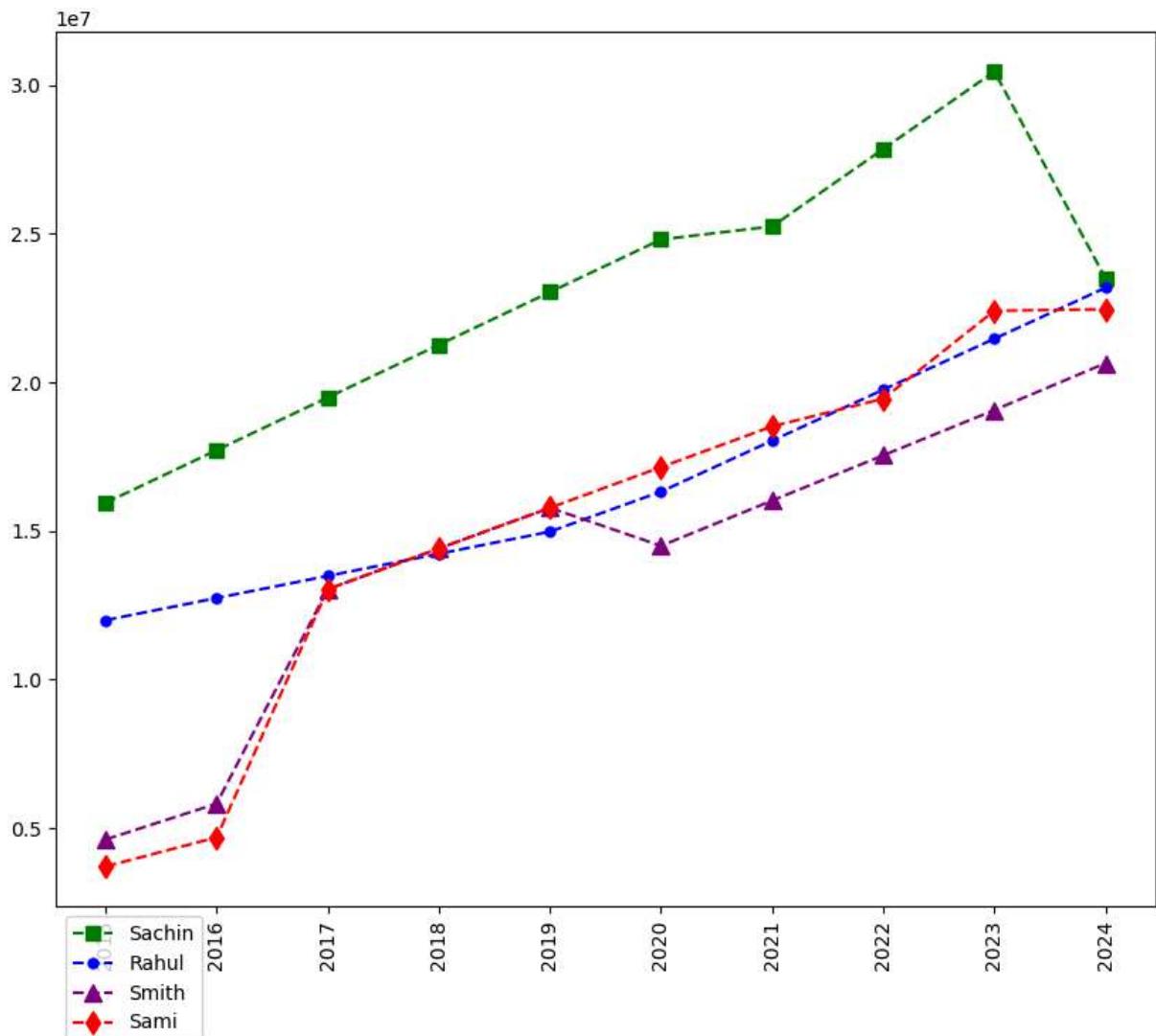
plt.show()
```



In [103]:

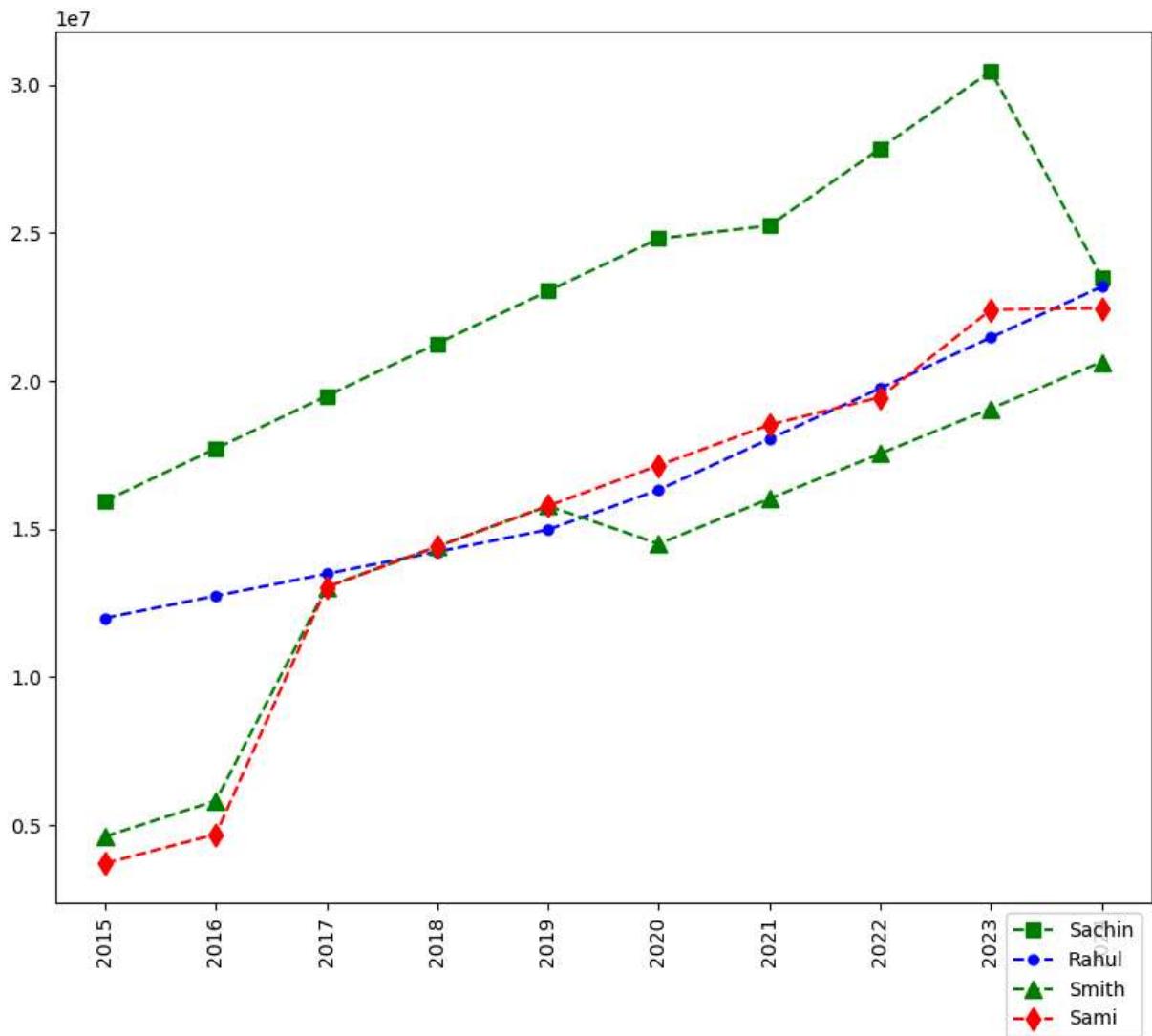
```
plt.plot(Salary[0], c='Green', ls = '--', marker = 's', ms = 7, label = Players[0])
plt.plot(Salary[1], c='Blue', ls = '--', marker = 'o', ms = 5, label = Players[1])
plt.plot(Salary[2], c='purple', ls = '--', marker = '^', ms = 8, label = Players[2])
plt.plot(Salary[3], c='Red', ls = '--', marker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



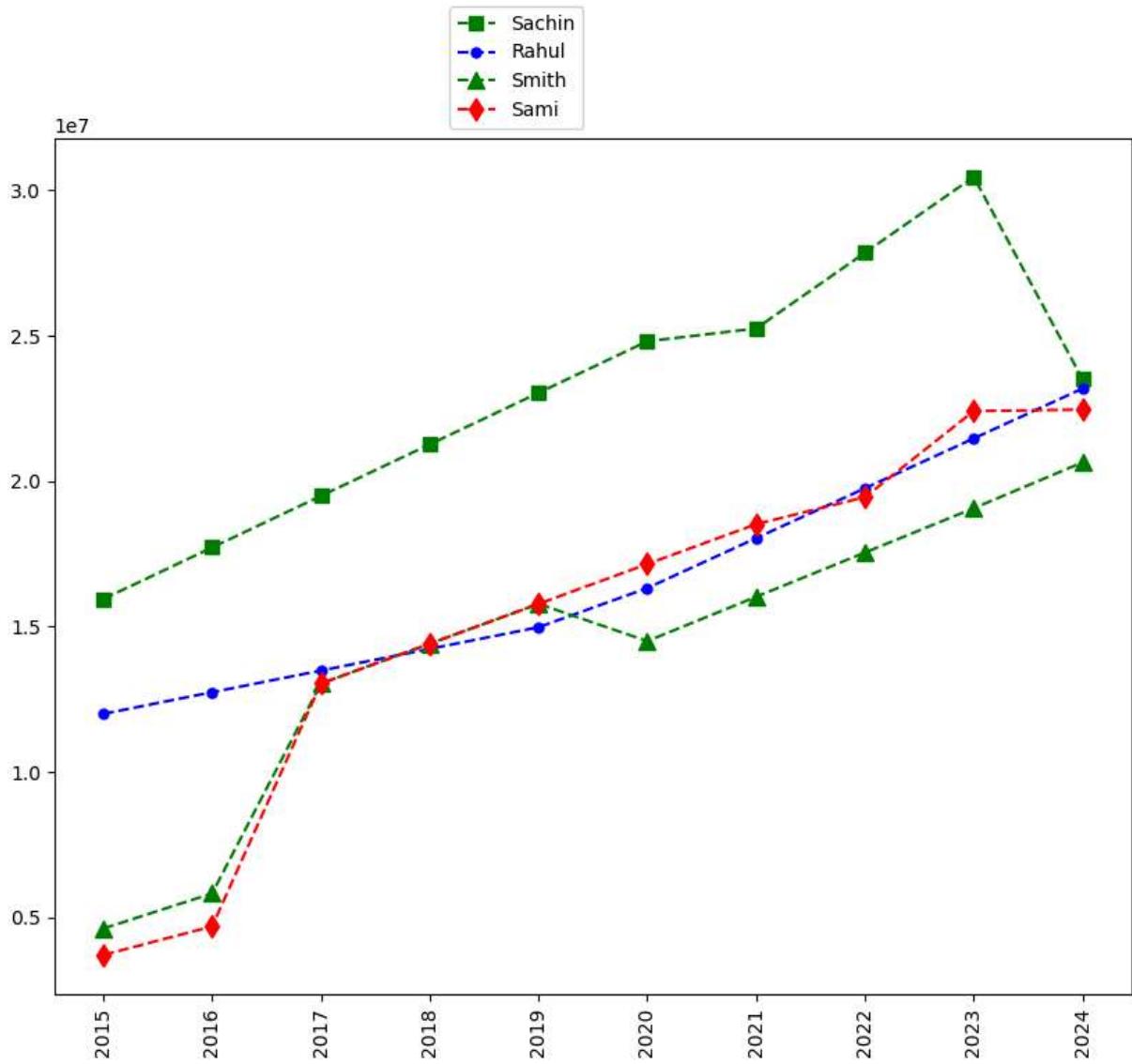
```
In [104]: plt.plot(Salary[0], c='Green', ls = '--', marker = 's', ms = 7, label = Players[0])
plt.plot(Salary[1], c='Blue', ls = '--', marker = 'o', ms = 5, label = Players[1])
plt.plot(Salary[2], c='Green', ls = '--', marker = '^', ms = 8, label = Players[2])
plt.plot(Salary[3], c='Red', ls = '--', 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 [105]: 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()
```



```
In [106...]: 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()
```

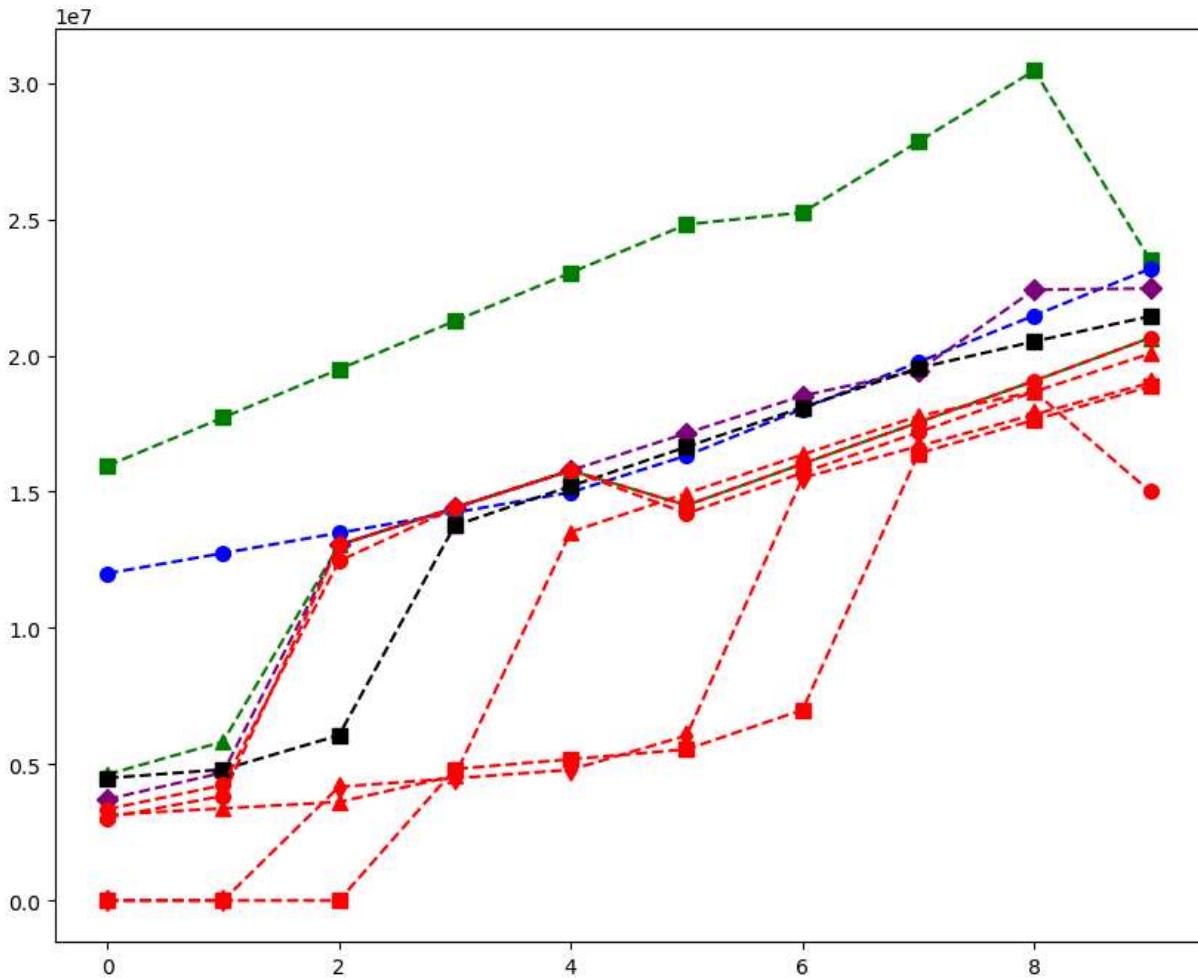
```
-----  
ValueError                                Traceback (most recent call last)  
Cell In[106], line 12  
      9 plt.plot(Salary[8], c='Red', ls = '--', marker = 's', ms = 7, label = Player  
     10 s[8])  
     11 plt.plot(Salary[9], c='Red', ls = '--', marker = 'o', ms = 7, label = Player  
     12 s[9])  
--> 13 plt.legend(loc = 'lower right',bbox_to_anchor=(0.5,1) )  
     14 plt.xticks(list(range(0,10)), Seasons, rotation='vertical')  
     15 plt.show()  
  
File ~\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 ~\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 ~\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, labelspacing, handlelength, handleheight, handletextpad, borderaxespad, columnspacing, ncols, 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)  
    564 # Set legend location  
--> 566 self.set_loc(loc)  
    567 # figure out title font properties:  
    568 if title_fontsize is not None and title_fontproperties is not None:  
  
File ~\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 ~\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 [ ]: # 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 [ ]: