matrix-visualization

July 13, 2023

1 Project / Task - 3

1.0.1 MATRICES / NUMPY —–

- Matrix is the tabular representation of the data
- Lot of datas are stored in table format,that is why Matrices is very very important topic in python
- as we working on dataframe so matrices are played a major rule
- List is one dimension & matrix is multidimension
- indexation is very important to plot the datapoints
- we will see tht & we gonna analyze the NBA players

- hear i have taken top 10 highest paid player in 2015-2016 season
- we will analyze how 10 players have been playing over the past 10 years & we had the data for past 10yrs yrs
- our main goal is to find trends,patterns & their performance for the past 10 yrs
- ultimately they haven't always been top 10 player & lets see how they improving, what actually secreates or patterns
- dont worry guys if you dont know anything about basket ball NBA
- I will explain indepth of everything
- lets analyze the statistics of the basket ball player
- gp - total games played,mpg - minutes per game,field goal(accuracy), ppg (points per game) – this is no of point player has scores in that season

- guys slowly i am bringing you into data analytics, jump into datavisualization using python
- i will give you the this code can everybody copy and paste your jupyter notebook
- Now i will explain with matrices

[1]:
```python
#Import numpy
import numpy as np
```

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

#Games
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]
#Matrix
Games = np.array([Sachin_G, Rahul_G, Smith_G, Sami_G, Pollard_G, Morris_G,␣
 ↪Samson_G, Dhoni_G, Kohli_G, Sky_G])

#Points
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]
#Matrix
Points = np.array([Sachin_PTS, Rahul_PTS, Smith_PTS, Sami_PTS, Pollard_PTS,␣
 ↪Morris_PTS, Samson_PTS, Dhoni_PTS, Kohli_PTS, Sky_PTS])
```

[2]:

```
Salary  # martrix format
```

[2]:

```
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]])
```

```
[3]: # Building your first matrix -
     Games
```

```
[3]: 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]])
```

```
[4]: Points
```

```
[4]: 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]])
```

```
[5]: 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]
```

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

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

```
[7]: mydata
```

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

```
[8]: #np.reshape(mydata,(5,4), order = 'c') #'C' means to read / write the elements␣
      ↪using C-like index order
     MATR1 = np.reshape(mydata, (5,4), order = 'c')
```

```
MATR1
```

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

```
[9]: MATR1
```

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

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

```
[10]: 19
```

```
[11]: MATR1[3,3]
```

```
[11]: 15
```

```
[12]: MATR1
```

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

```
[13]: MATR1[-3,-1]
```

```
[13]: 11
```

```
[14]: MATR1
```

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

```
[15]: mydata
```

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

```
[16]: MATR2 = np.reshape(mydata, (5,4), order = 'F') # reshape behaviour are  ␣
        ↪'C','F','A'
      MATR2
```

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

```
[17]: MATR2[4,3]
```

```
[17]: 19
```

```
[18]: MATR2[0,2]
```

```
[18]: 10
```

```
[19]: MATR2[0:2]
```

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

```
[20]: MATR2
```

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

```
[21]: MATR2[1:2]
```

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

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

```
[22]: 11
```

```
[23]: MATR2
```

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

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

```
[24]: 18
```

```
[25]: MATR2[-3,-3]
```

```
[25]: 7
```

```
[26]: MATR2
```

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

```
[27]: MATR2[0:2]
```

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

```
[28]: mydata
```

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

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

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

```
[30]: MATR2 ## F shaped
```

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

```
[31]: MATR1 # C shaped
```

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

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

```
[33]: [a1,a2,a3] # List same dataypte
```

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

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

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

```
[35]: Games
```

```
[35]: 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]])
```

```
[36]: Games[0]
```

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

```
[37]: Games[5]
```

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

```
[38]: Games[0:5]
```

```
[38]: 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]])
```

```
[39]: Games[0,5]
```

```
[39]: 82
```

```
[40]: Games[0,2]
```

```
[40]: 82
```

```
[41]: Games
```

```
[41]: 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]])
```

```
[42]: Games[0:2]
```

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

```
[ ]: Games
```

```
[ ]: Games[1:2]
```

```
[ ]: Games[2]
```

```
[ ]: Games
```

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

```
[43]: Games
```

```
[43]: 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]])
```

[44]:
```
Games[-3:-1]
```

[44]:
```
array([[35, 35, 80, 74, 82, 78, 66, 81, 81, 27],
       [40, 40, 40, 81, 78, 81, 39,  0, 10, 51]])
```

[45]:
```
Games[-3,-1]
```

[45]:
```
27
```

[46]:
```
Points
```

[46]:
```
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]])
```

[47]:
```
Points[0]
```

[47]:
```
array([2832, 2430, 2323, 2201, 1970, 2078, 1616, 2133,   83,  782])
```

[48]:
```
Points
```

[48]:
```
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]])
```

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

[49]:
```
1104
```

[50]:

```
Points[3:6]
```

[50]:

```
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]])
```

[52]:

```
Points
```

[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]])
```

[51]:

```
Points[-6,-1]
```

[51]:

```
646
```

[ ]:

```
#====== DICTIONARY =======#

# dict does not maintain the order

dict1 = {'key1':'val1', 'key2':'val2', 'key3':'val3'}
```

[ ]:

```
dict1
```

[ ]:

```
dict1['key2']
```

[ ]:

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

[ ]:

```
dict2
```

[ ]:

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

[ ]:

```
dict3
```

[ ]:

```
dict3['Germany']
```

[ ]:

```
# if you check theat 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 players are sitting
```

```
[53]: Games
```

```
[53]: 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]])
```

```
[54]: Pdict
```

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

```
[55]: # how do i know player kobebryant is at

      Pdict['Sachin']
```

```
[55]: 0
```

```
[56]: Games[0]
```

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

```
[57]: Games
```

```
[57]: 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]])
```

```
[58]: Pdict['Rahul']
```

```
[58]: 1
```

```
[59]: Games[1]
```

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

2 Games

```
[60]: Games[Pdict['Rahul']]
```

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

```
[ ]: Points
```

```
[61]: Salary
```

```
[61]: 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]])
```

```
[62]: Salary[2,4]
```

```
[62]: 15779912
```

```
[63]: Salary
```

```
[63]: 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]])
```

```
[65]: Salary[Pdict['Sky']][Sdict['2019']]
```

```
[65]: 15000000
```

```
[66]: Salary
```

```
[66]: 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]])
```

```
[67]: Games
```

```
[67]: 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]])
```

```
[68]: Salary/Games
```

```
C:\Users\kdata\anaconda3\lib\site-packages\ipykernel_launcher.py:1:
RuntimeWarning: divide by zero encountered in true_divide
  """Entry point for launching an IPython kernel.
```

```
[68]: 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]])
```

[69]:
```python
np.round(Salary/Games)
```

```
C:\Users\kdata\anaconda3\lib\site-packages\ipykernel_launcher.py:1:
RuntimeWarning: divide by zero encountered in true_divide
  """Entry point for launching an IPython kernel.
```

[69]:
```
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.]])
```

[ ]:
```python
import warnings
warnings.filterwarnings('ignore')
```

```
#np.round(FieldGoals/Games)
#FieldGoals/Games  # this matrix is lot of decimal points yo can not round
#round()
```

[ ]:

```
## --- First visualization ----##
```

[73]:

```
import numpy as np
import matplotlib.pyplot as plt
```

[74]:

```
%matplotlib inline # keep the plot inside jupyter nots insted of getting in␣
 ↪other screen
```

```
UsageError: unrecognized arguments: # keep the plot inside jupyter nots insted
of getting in other screen
```

[75]:

```
Salary
```

[75]:

```
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]])
```

[76]:

```
Salary[0]
```

[76]:

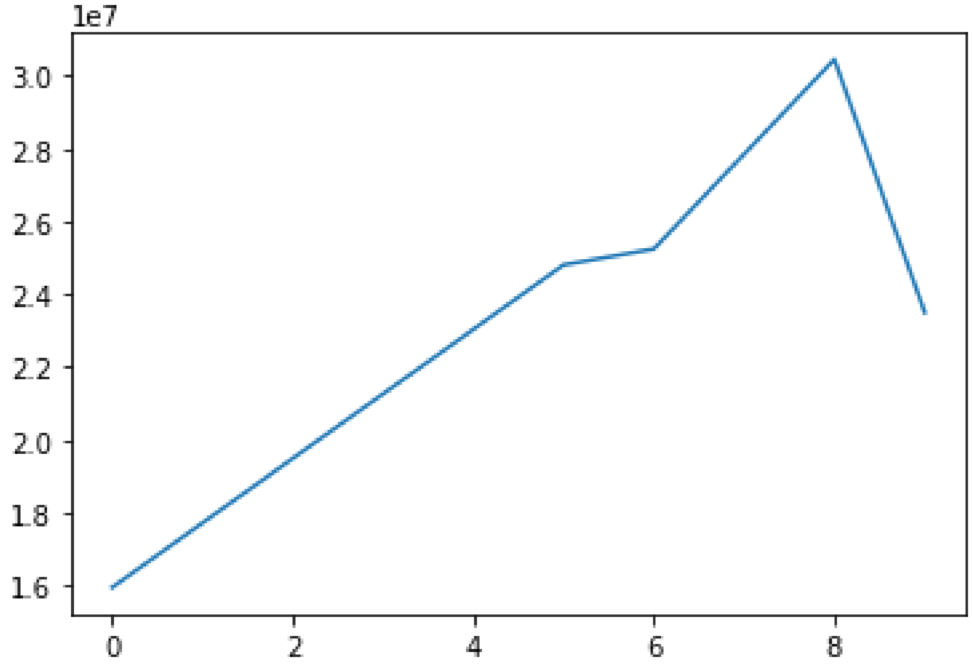
```
array([15946875, 17718750, 19490625, 21262500, 23034375, 24806250,
       25244493, 27849149, 30453805, 23500000])
```

[77]:

```
plt.plot(Salary[0])
```

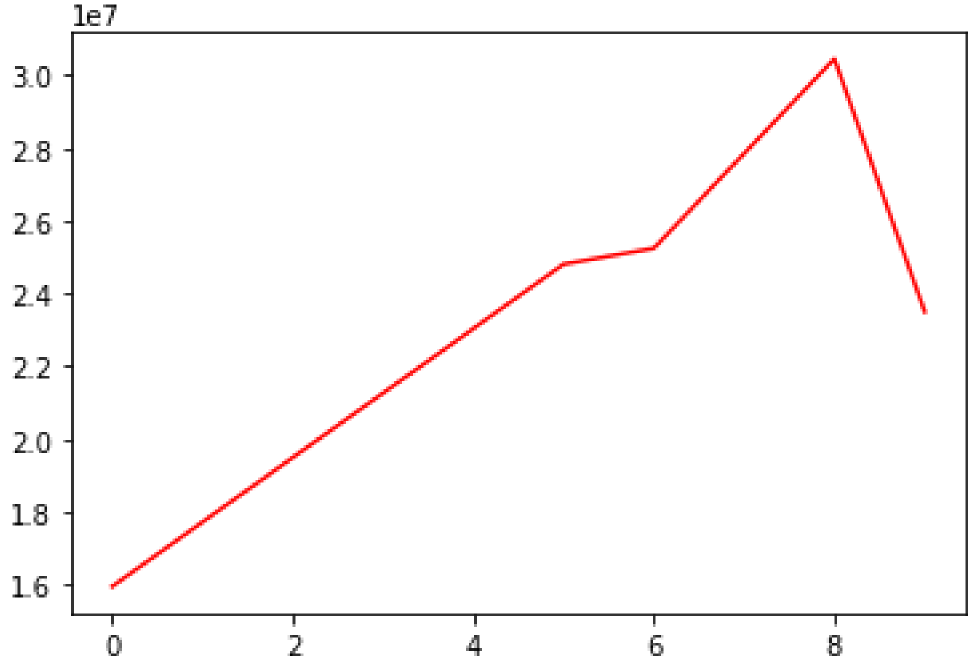
[77]:

```
[<matplotlib.lines.Line2D at 0x19ecb3017c8>]
```



[78]:
```
plt.plot(Salary[0], c='red')
```

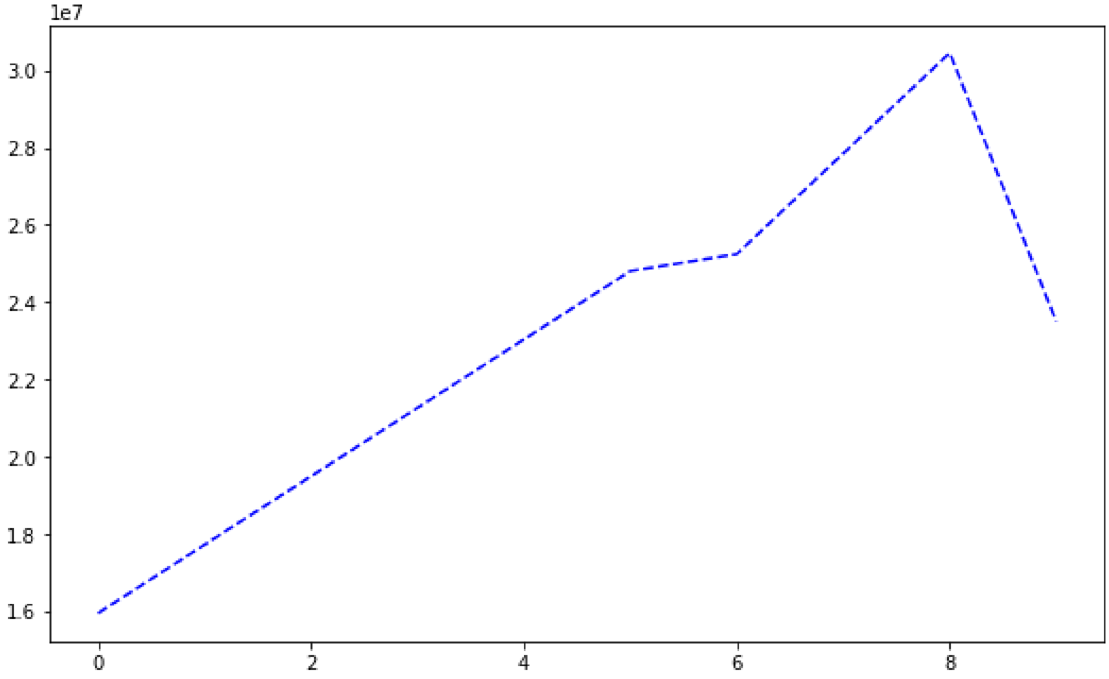
[78]:
```
[<matplotlib.lines.Line2D at 0x19ecec3f088>]
```



[80]:
```
%matplotlib inline
plt.rcParams['figure.figsize'] = 10,6
```

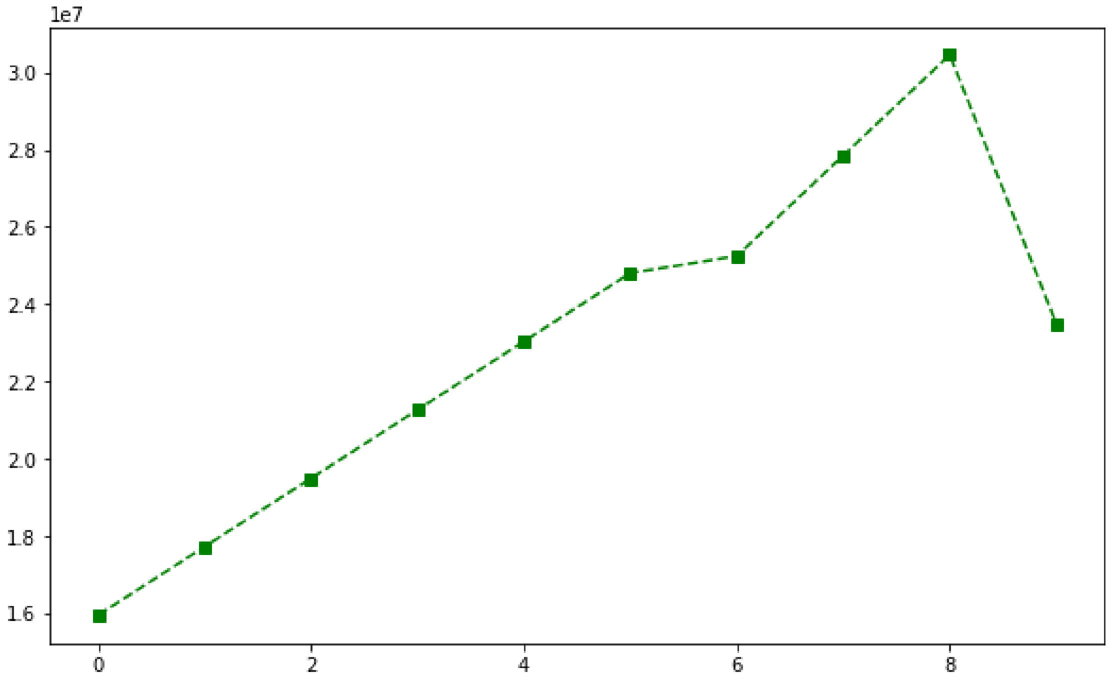
[88]:
```
plt.plot(Salary[0], c='Blue', ls = 'dashed')
```

[88]:
```
[<matplotlib.lines.Line2D at 0x19ecf0d98c8>]
```



[89]:
```
plt.plot(Salary[0], c='Green', ls = '--', marker = 's') # s - squares
```

[89]:
```
[<matplotlib.lines.Line2D at 0x19ecf144448>]
```

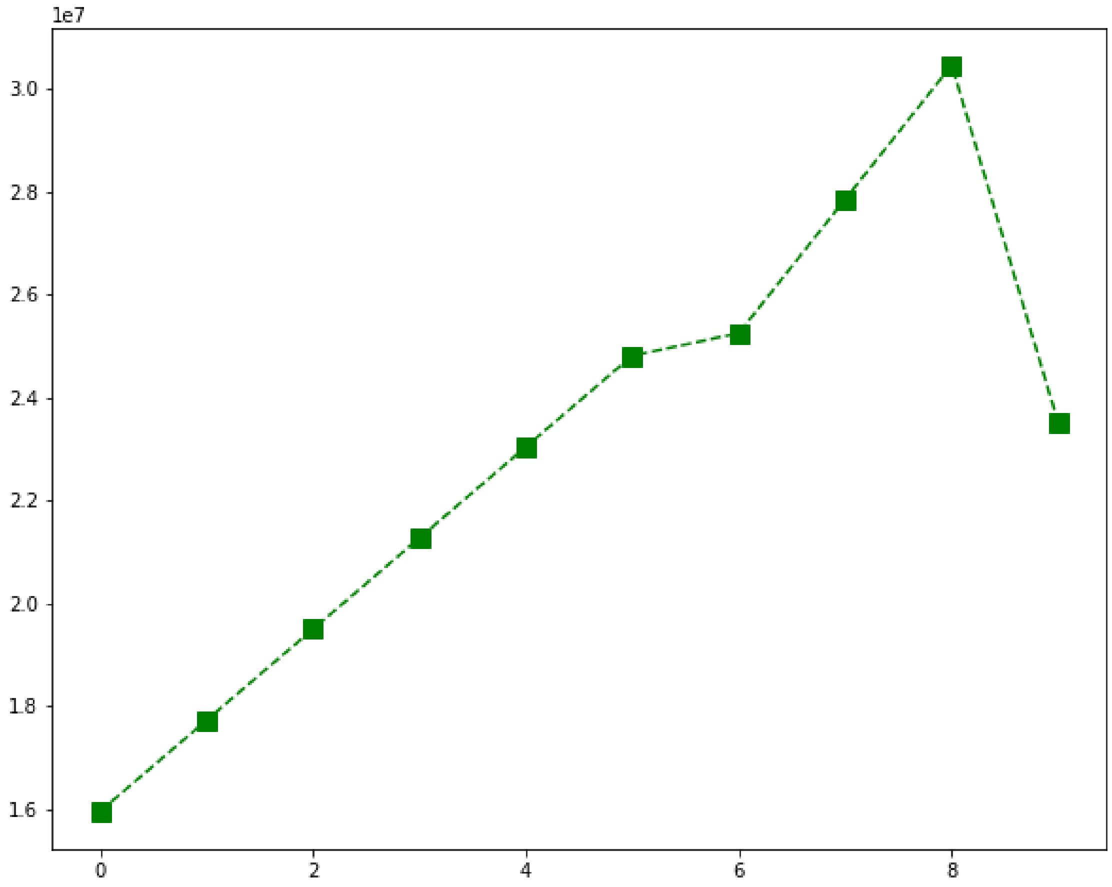


[90]:

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

[94]:

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



```
[95]: list(range(0,10))
```

```
[95]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
[96]: Sdict
```

```
[96]: {'2010': 0,
       '2011': 1,
       '2012': 2,
       '2013': 3,
       '2014': 4,
       '2015': 5,
       '2016': 6,
       '2017': 7,
       '2018': 8,
       '2019': 9}
```

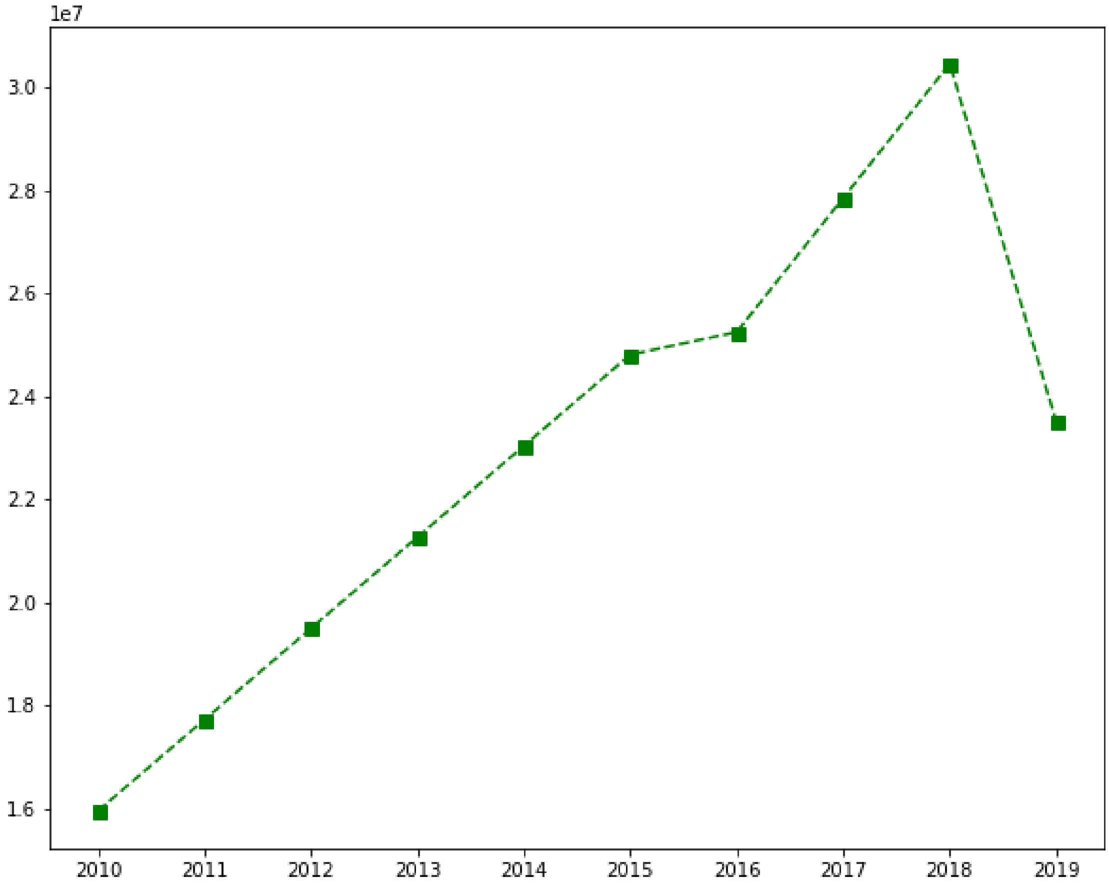
```
[97]: Pdict
```

[97]:

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

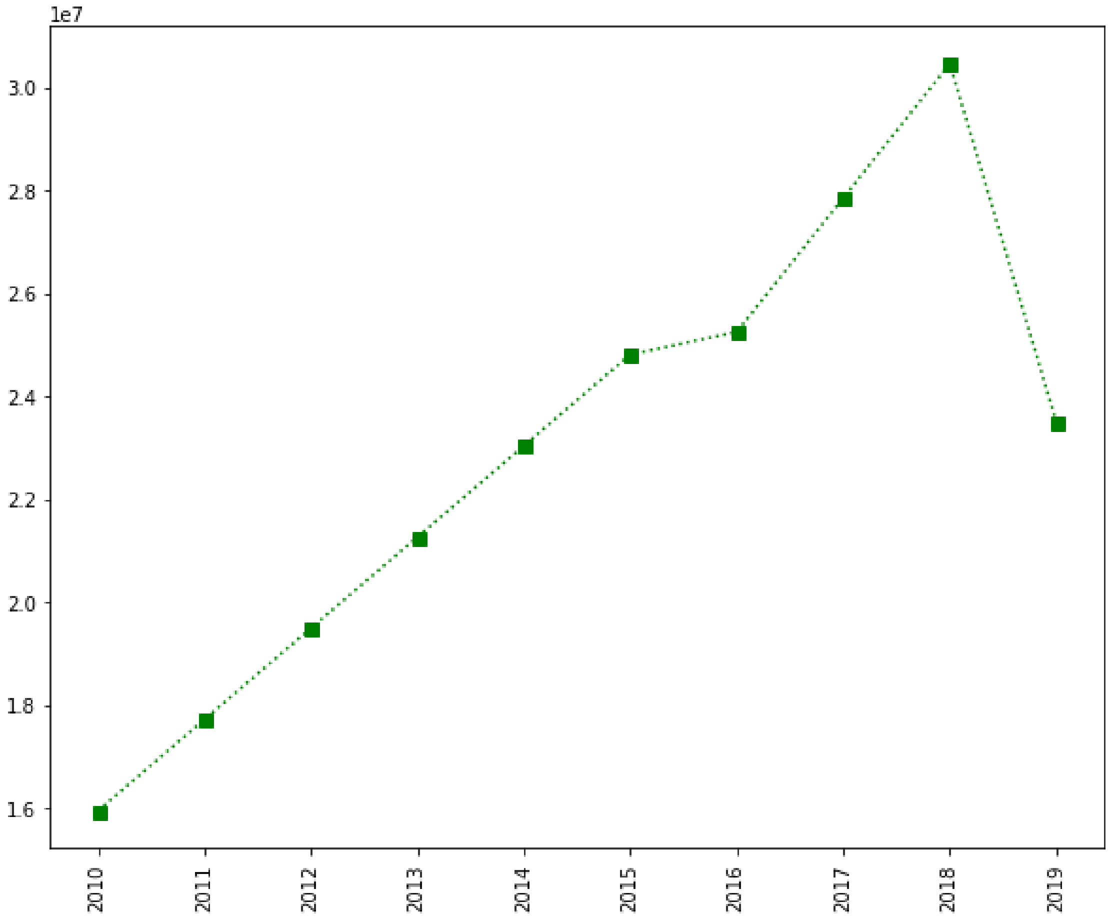
[98]:

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



[99]:

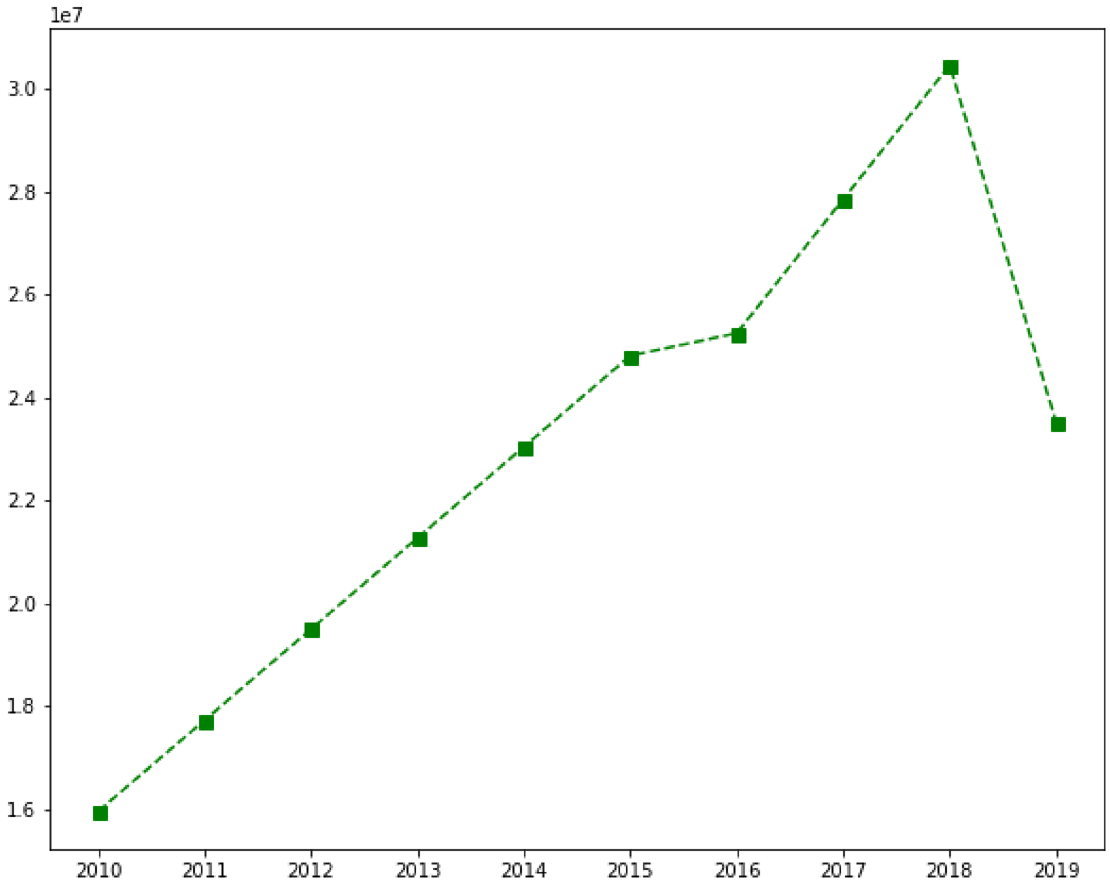
```
plt.plot(Salary[0], c='Green', ls = ':', marker = 's', ms = 7, label =␣
 ↪Players[0])
plt.xticks(list(range(0,10)), Seasons,rotation='vertical')
plt.show()
```



[100]:
```
Games
```

[100]:
```
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]])
```

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



```
[102]: Salary[0]
```

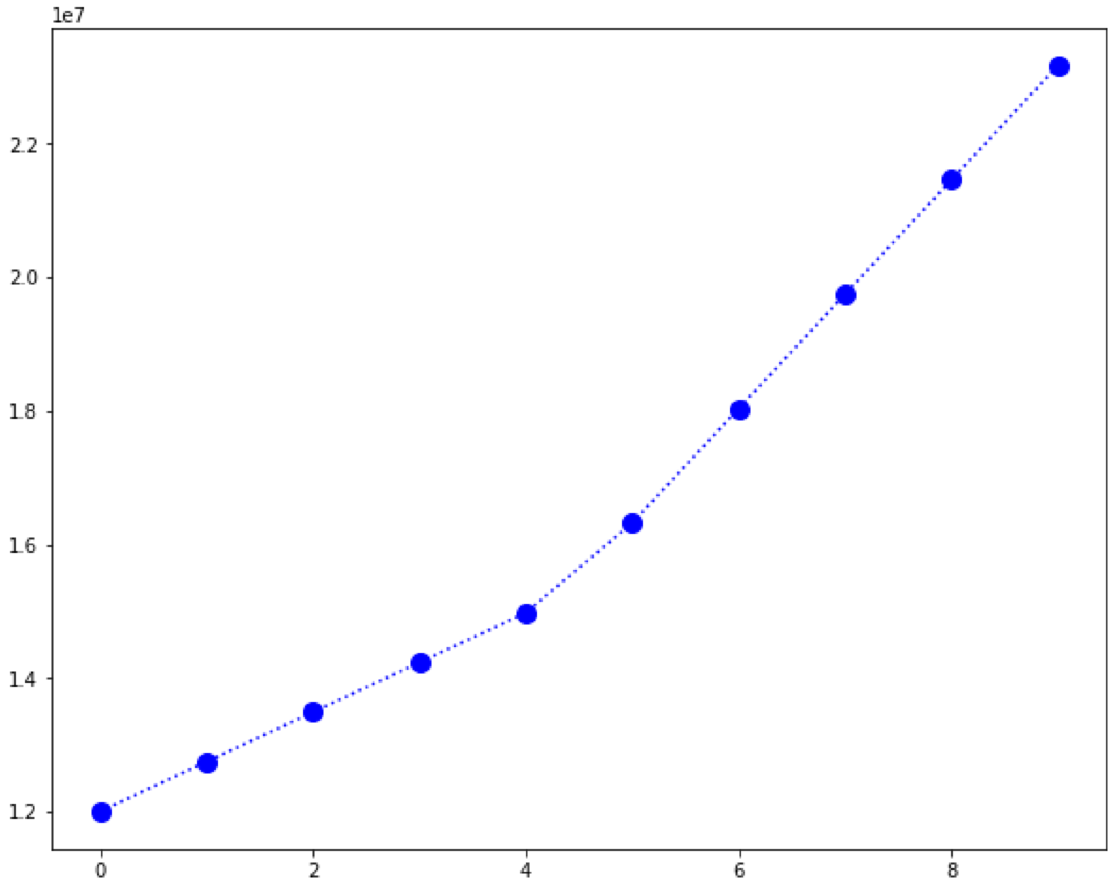
```
[102]: array([15946875, 17718750, 19490625, 21262500, 23034375, 24806250,
               25244493, 27849149, 30453805, 23500000])
```

```
[103]: Salary[1]
```

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

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

```
[105]: [<matplotlib.lines.Line2D at 0x19ecf09db48>]
```



[ ]:
```
# More visualization
```

[ ]:
```
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()
```

[ ]:
```
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()
```

[ ]:
```
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()
```

[ ]:
```
# 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()
```

[ ]:
```
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()
```

[ ]:
```
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()
```

[ ]:

```
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()
```

[ ]:

```
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 = 'lover right',bbox_to_anchor=(0.5,1) )
plt.xticks(list(range(0,10)), Seasons,rotation='vertical')

plt.show()
```

[ ]:

```
# 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 - 1>Matrices 2>Building matrices - np.reshape 3>Dictionaried in python (order doesnot mater) (keys & values) 4>visualizaing using pyplot 5>Basket ball analysis

[ ]:

```
```