In [52]:
```
import numpy as np

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

#Players
Players = ["Sachin","Rahul","Smith","Sami","Pollard","Morris","Samson","Dhoni","
Pdict = {"Sachin":0,"Rahul":1,"Smith":2,"Sami":3,"Pollard":4,"Morris":5,"Samson"

#Salaries
Sachin_Salary = [15946875,17718750,19490625,21262500,23034375,24806250,25244493,
Rahul_Salary = [12000000,12744189,13488377,14232567,14976754,16324500,18038573,1
Smith_Salary = [4621800,5828090,13041250,14410581,15779912,14500000,16022500,175
Sami_Salary = [3713640,4694041,13041250,14410581,15779912,17149243,18518574,1945
Pollard_Salary = [4493160,4806720,6061274,13758000,15202590,16647180,18091770,19
Morris_Salary = [3348000,4235220,12455000,14410581,15779912,14500000,16022500,17
Samson_Salary = [3144240,3380160,3615960,4574189,13520500,14940153,16359805,1777
Dhoni_Salary = [0,0,4171200,4484040,4796880,6053663,15506632,16669630,17832627,1
Kohli_Salary = [0,0,0,4822800,5184480,5546160,6993708,16402500,17632688,18862875
Sky_Salary = [3031920,3841443,13041250,14410581,15779912,14200000,15691000,17182
#Matrix
Salary = np.array([Sachin_Salary, Rahul_Salary, Smith_Salary, Sami_Salary, Polla

#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, Samsc

#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, Morr
```

In [54]:
```
Salary
```

Out[54]:
```
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 [56]:
```
Games
```

Out[56]:
```
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 [58]:
```
Games[1] #This prints the 1th indexed row
```

Out[58]:
```
array([82, 57, 82, 79, 76, 72, 60, 72, 79, 80])
```

In [60]:
```
Games[0] #This prints the 0th indexed row
```

Out[60]:
```
array([80, 77, 82, 82, 73, 82, 58, 78,  6, 35])
```

In [62]:
```
Games[0:5] #This prints top 5 rows
```

Out[62]:
```
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]])
```

In [64]:
```
Points
```

Out[64]:
```
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 [66]:
```
Points[0:5]
```

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

In [68]:
```
Games[0,5]  #0th row and 5th column
```

Out[68]:
```
82
```

In [70]:
```
Games[-3,-1]  #-3rd row and -1th column value
```

Out[70]:
```
27
```

In [72]:
```
Games[-3:-1]  #-3rd row to -2th row (n-1)th = -1-1 = -2th row printed
```

Out[72]:
```
array([[35, 35, 80, 74, 82, 78, 66, 81, 81, 27],
       [40, 40, 40, 81, 78, 81, 39,  0, 10, 51]])
```

In [74]:
```
Salary/Games # For calculating salary of 1 game
```

Out[74]:
```
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]])
```

For round off

In [77]:
```
np.round(Salary/Games)
```

Out[77]:
```
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 [79]:
```
np.round(Salary//Games) # 2 times // to get perfect divsion i.e round off
```

Out[79]:
```
array([[ 199335,  230113,  237690,  259298,  315539,  302515,  435249,
         357040, 5075634,  671428],
       [ 146341,  223582,  164492,  180159,  197062,  226729,  300642,
         274342,  271730,  289759],
       [  58503,   74719,  173883,  177908,  207630,  183544,  258427,
         230855,  247629,  299194],
       [  46420,   72216,  169366,  218342,  228694,  222717,  336701,
         290298,  291006,  561450],
       [  54794,   58618,   73917,  174151,  185397,  213425,  335032,
         257057,  288918,  522835],
       [  47828,   61380,  185895,  187150,  225427,  188311,  281096,
         237094,  241360,  469190],
       [  40310,   52815,   45199,   58643,  300455,  186751,  272663,
         253992,  301103,  244738],
       [      0,       0,   52140,   60595,   58498,   77611,  234948,
         205797,  220155,  703541],
       [      0,       0,       0,   59540,   66467,   68471,  179325,
               0, 1763268,  369860],
       [  40425,   75322,  255710,  182412,  204933,  186842,  320224,
         249014,  345796,  241935]])
```

In [81]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [83]:
```
import matplotlib.pyplot as plt
```

Read this document - https://pypi.org/project/matplotlib/

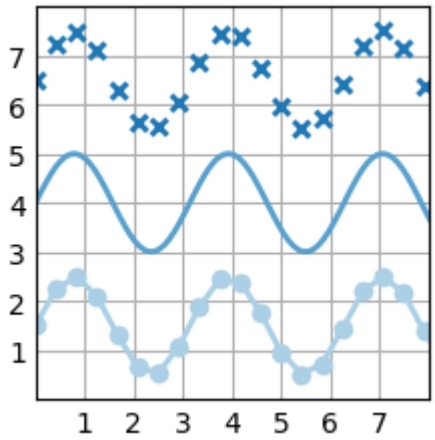
Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

Refer to github for whatever we are learning - https://github.com/matplotlib/matplotlib

Pasting below graph code from - https://matplotlib.org/stable/plot_types/basic/plot.html#sphx-glr-plot-types-basic-plot-

py

In [89]:
```
import matplotlib.pyplot as plt
import numpy as np

plt.style.use('_mpl-gallery')

# make data
x = np.linspace(0, 10, 100)
y = 4 + 1 * np.sin(2 * x)
x2 = np.linspace(0, 10, 25)
y2 = 4 + 1 * np.sin(2 * x2)

# plot
fig, ax = plt.subplots()

ax.plot(x2, y2 + 2.5, 'x', markeredgewidth=2)
ax.plot(x, y, linewidth=2.0)
ax.plot(x2, y2 - 2.5, 'o-', linewidth=2)

ax.set(xlim=(0, 8), xticks=np.arange(1, 8),
       ylim=(0, 8), yticks=np.arange(1, 8))

plt.show()
```



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]])
```

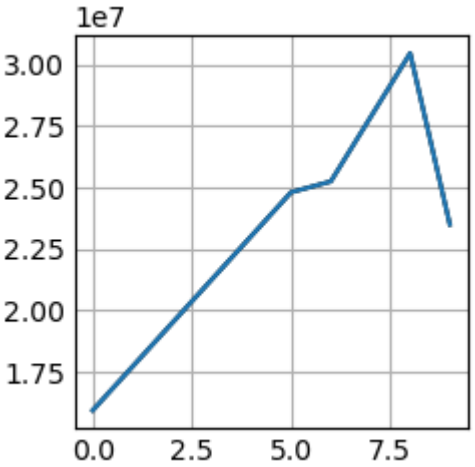
In [93]:
```
Salary[0]
```

Out[93]:
```
array([15946875, 17718750, 19490625, 21262500, 23034375, 24806250,
       25244493, 27849149, 30453805, 23500000])
```

In [95]:
```
plt.plot(Salary[0], color = 'black')
```

Out[95]:
```
[<matplotlib.lines.Line2D at 0x21b2cb9aab0>]
```

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

#If graph is not generated
```



These numbers are created automatically by Matplotlib. Matplotlib library is designed to inbuild these numbers based on the data

Insight-1 : Based on above graph, Sachin salary increased till 2023 and then it has reduced

In [101…
```
plt.plot(Salary[0], c = 'b') #Added 1 parameter, here r = red, b = blue, k = bla
```

Out[101… [<matplotlib.lines.Line2D at 0x21b2ceb2690>]

Markers

====== character description ======

- `'.'` point marker
- `','` pixel marker
- `'o'` circle marker
- `'v'` triangle_down marker
- `'^'` triangle_up marker
- `'<'` triangle_left marker
- `'>'` triangle_right marker
- `'1'` tri_down marker
- `'2'` tri_up marker
- `'3'` tri_left marker
- `'4'` tri_right marker
- `'8'` octagon marker
- `'s'` square marker
- `'p'` pentagon marker
- `'P'` plus (filled) marker
- `'*'` star marker
- `'h'` hexagon1 marker
- `'H'` hexagon2 marker
- `'+'` plus marker
- `'x'` x marker
- `'X'` x (filled) marker
- `'D'` diamond marker
- `'d'` thin_diamond marker
- `'|'` vline marker
- `'_'` hline marker

Line Styles

===== character description =======

- `'-'` solid line style
- `'--'` dashed line style
- `'-.'` dash-dot line style
- `':'` dotted line style

==========

Example format strings::

```
'b'    # blue markers with default shape
'or'   # red circles
'-g'   # green solid line
'--'   # dashed line with default color
```

```
'^k:'  # black triangle_up markers connected by a dotted line
```

Colors

The supported color abbreviations are the single letter codes

======= character color =======

- `'b'` blue
- `'g'` green
- `'r'` red
- `'c'` cyan
- `'m'` magenta
- `'y'` yellow
- `'k'` black
- `'w'` white

============= ================================

In [104…
```
plt.plot(Salary[0], c = 'k', marker = 'v') # added the 2nd parameter marker
```

Out[104…
```
[<matplotlib.lines.Line2D at 0x21b2cb9b050>]
```

Salary and year is highlighted

Markers

============= ============= character description =============
=============

- `'.'` point marker
- `','` pixel marker
- `'o'` circle marker
- `'v'` triangle_down marker
- `'^'` triangle_up marker
- `'<'` triangle_left marker
- `'>'` triangle_right marker
- `'1'` tri_down marker
- `'2'` tri_up marker
- `'3'` tri_left marker
- `'4'` tri_right marker
- `'8'` octagon marker
- `'s'` square marker
- `'p'` pentagon marker
- `'P'` plus (filled) marker
- `'*'` star marker
- `'h'` hexagon1 marker
- `'H'` hexagon2 marker

- `'+'` plus marker
- `'x'` x marker
- `'X'` x (filled) marker
- `'D'` diamond marker
- `'d'` thin_diamond marker
- `'|'` vline marker
- `'_'` hline marker

========== ==========

In [108…
```
plt.plot(Salary[0], c = 'b', marker = 'v', ls = '--') # added the 3rd parameter.
```

Out[108…
```
[<matplotlib.lines.Line2D at 0x21b2ceeddf0>]
```

[<matplotlib.lines.Line2D at 0x1d1cd8a3bc0>] -> This means at memory location <0x1d1cd8a3bc0>, the graph is plotted

le7 -> Label 7

In [111…
```
plt.plot(Salary[0], c = 'b', marker = 'v', ls = '*') # Error generated as '*' is
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[111], line 1
----> 1 plt.plot(Salary[0], c = 'b', marker = 'v', ls = '*')

File ~\anaconda3\Lib\site-packages\matplotlib\pyplot.py:3794, in plot(scalex, sca
ley, data, *args, **kwargs)
   3786 @_copy_docstring_and_deprecators(Axes.plot)
   3787 def plot(
   3788     *args: float | ArrayLike | str,
   (...)
   3792     **kwargs,
   3793 ) -> list[Line2D]:
-> 3794     return gca().plot(
   3795         *args,
   3796         scalex=scalex,
   3797         scaley=scaley,
   3798         **({"data": data} if data is not None else {}),
   3799         **kwargs,
   3800     )

File ~\anaconda3\Lib\site-packages\matplotlib\axes\_axes.py:1779, in Axes.plot(se
lf, scalex, scaley, data, *args, **kwargs)
   1536 """
   1537 Plot y versus x as lines and/or markers.
   1538
   (...)
   1776 (``'green'``) or hex strings (``'#008000'``).
   1777 """
   1778 kwargs = cbook.normalize_kwargs(kwargs, mlines.Line2D)
-> 1779 lines = [*self._get_lines(self, *args, data=data, **kwargs)]
   1780 for line in lines:
   1781     self.add_line(line)

File ~\anaconda3\Lib\site-packages\matplotlib\axes\_base.py:296, in _process_plot
_var_args.__call__(self, axes, data, *args, **kwargs)
    294     this += args[0],
    295     args = args[1:]
--> 296 yield from self._plot_args(
    297     axes, this, kwargs, ambiguous_fmt_datakey=ambiguous_fmt_datakey)

File ~\anaconda3\Lib\site-packages\matplotlib\axes\_base.py:534, in _process_plot
_var_args._plot_args(self, axes, tup, kwargs, return_kwargs, ambiguous_fmt_datake
y)
    532     return list(result)
    533 else:
--> 534     return [l[0] for l in result]

File ~\anaconda3\Lib\site-packages\matplotlib\axes\_base.py:527, in <genexpr>(.0)
    522 else:
    523     raise ValueError(
    524         f"label must be scalar or have the same length as the input "
    525         f"data, but found {len(label)} for {n_datasets} datasets.")
--> 527 result = (make_artist(axes, x[:, j % ncx], y[:, j % ncy], kw,
    528                       {**kwargs, 'label': label})
    529           for j, label in enumerate(labels))
    531 if return_kwargs:
    532     return list(result)

File ~\anaconda3\Lib\site-packages\matplotlib\axes\_base.py:335, in _process_plot
```

```
_var_args._makeline(self, axes, x, y, kw, kwargs)
    333 kw = {**kw, **kwargs}  # Don't modify the original kw.
    334 self._setdefaults(self._getdefaults(kw), kw)
--> 335 seg = mlines.Line2D(x, y, **kw)
    336 return seg, kw

File ~\anaconda3\Lib\site-packages\matplotlib\lines.py:372, in Line2D.__init__(self, xdata, ydata, linewidth, linestyle, color, gapcolor, marker, markersize, markeredgewidth, markeredgecolor, markerfacecolor, markerfacecoloralt, fillstyle, antialiased, dash_capstyle, solid_capstyle, dash_joinstyle, solid_joinstyle, pickradius, drawstyle, markevery, **kwargs)
    369 self._dash_pattern = (0, None)  # offset, dash (scaled by linewidth)
    371 self.set_linewidth(linewidth)
--> 372 self.set_linestyle(linestyle)
    373 self.set_drawstyle(drawstyle)
    375 self._color = None

File ~\anaconda3\Lib\site-packages\matplotlib\lines.py:1177, in Line2D.set_linestyle(self, ls)
   1175 if ls in [' ', '', 'none']:
   1176     ls = 'None'
-> 1177 _api.check_in_list([*self._lineStyles, *ls_mapper_r], ls=ls)
   1178 if ls not in self._lineStyles:
   1179     ls = ls_mapper_r[ls]

File ~\anaconda3\Lib\site-packages\matplotlib\_api\__init__.py:129, in check_in_list(values, _print_supported_values, **kwargs)
    127 if _print_supported_values:
    128     msg += f"; supported values are {', '.join(map(repr, values))}"
--> 129 raise ValueError(msg)

ValueError: '*' is not a valid value for ls; supported values are '-', '--', '-.', ':', 'None', ' ', '', 'solid', 'dashed', 'dashdot', 'dotted'
```

Line Styles

========== ========== character description ========== ==========

- `'-'` solid line style
- `'--'` dashed line style
- `'-.'` dash-dot line style
- `':'` dotted line style

Whatever inbuild functions are given as parameter, we can use that only

In [115…
```
plt.plot(Salary[0], c = 'g', marker = 'd', ls = ':')
```

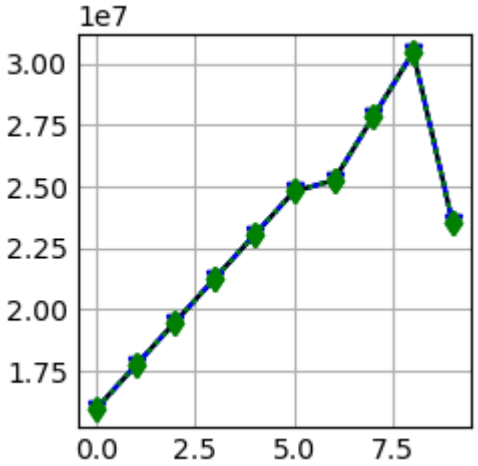
Out[115…
```
[<matplotlib.lines.Line2D at 0x21b2cbcbc20>]
```

To Re-size the graph

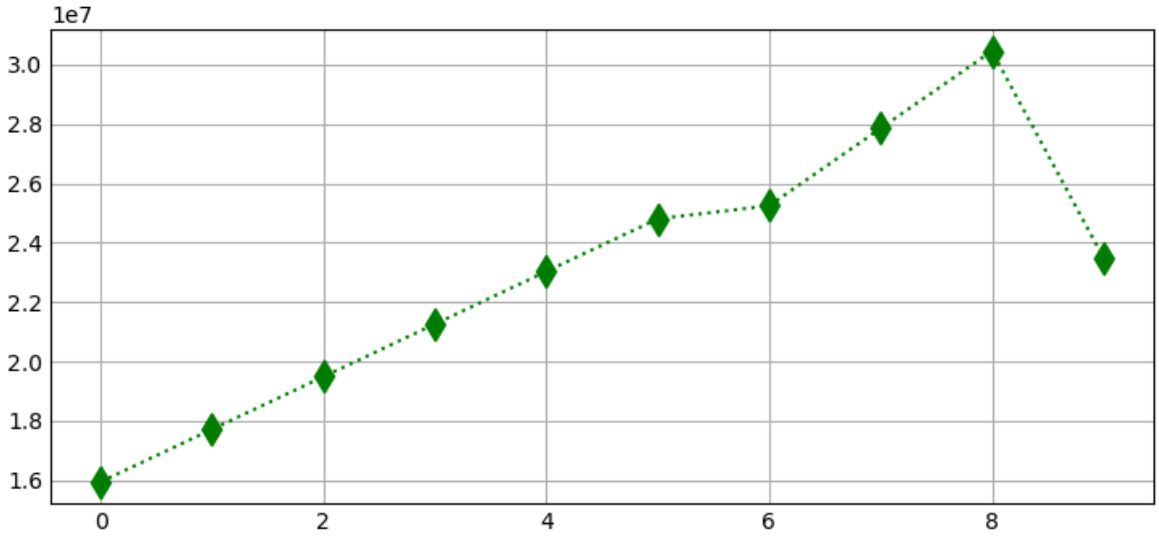
- If graph is big and it killed the windows - use command - >

In [117…
```
%matplotlib inline
plt.rcParams['figure.figsize'] = 7, 3 # Here 7 is the width and 3 is height
```

In [118…
```
plt.plot(Salary[0], c = 'g', marker = 'd', ls = ':')
plt.show()                          # use plt.show() if graph is not showing
```



In [119…
```
plt.plot(Salary[0], c = 'g', marker = 'd', ls = ':', ms = 10)  #ms = marker size
plt.show()
```



Diamond size is increased

plt.plot(Salary[0], c = 'g', marker = 'd', ls = ':', ms = 10) #ms = marker size plt.show()

- plt -> alias of matplotlib
- plot -> plot the graph
- Salary[0] -> 1st player salary
- c = 'g' -> Color = green color
- marker = 'd' -> Marker means diamond marker
- ls = ':' -> Linestyle is :
- ms = 10 -> Marker size is 10
- plt.show() -> Show the plot(graph)

In [122…
```
list (range(0,10))
```

Out[122… `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`

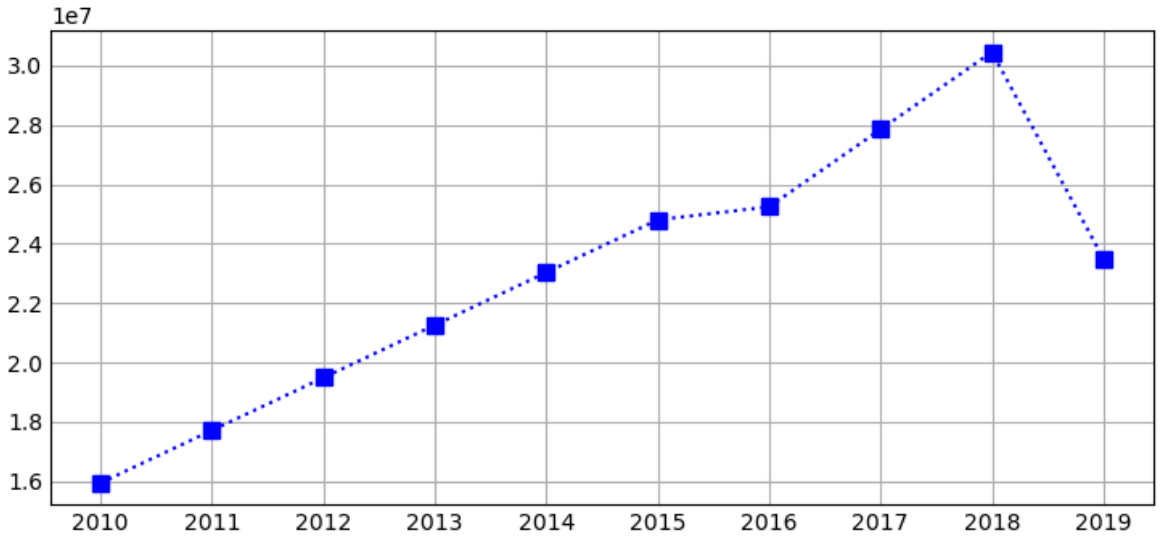
In [123…
```
Sdict  #Seasons dictionary
```

Out[123…
```
{'2010': 0,
 '2011': 1,
 '2012': 2,
 '2013': 3,
 '2014': 4,
 '2015': 5,
 '2016': 6,
 '2017': 7,
 '2018': 8,
 '2019': 9}
```

In [124…
```
Pdict  #Pdict adds value. Pdict means Players dictionary
```

Out[124…
```
{'Sachin': 0,
 'Rahul': 1,
 'Smith': 2,
 'Sami': 3,
 'Pollard': 4,
 'Morris': 5,
 'Samson': 6,
 'Dhoni': 7,
 'Kohli': 8,
 'Sky': 9}
```

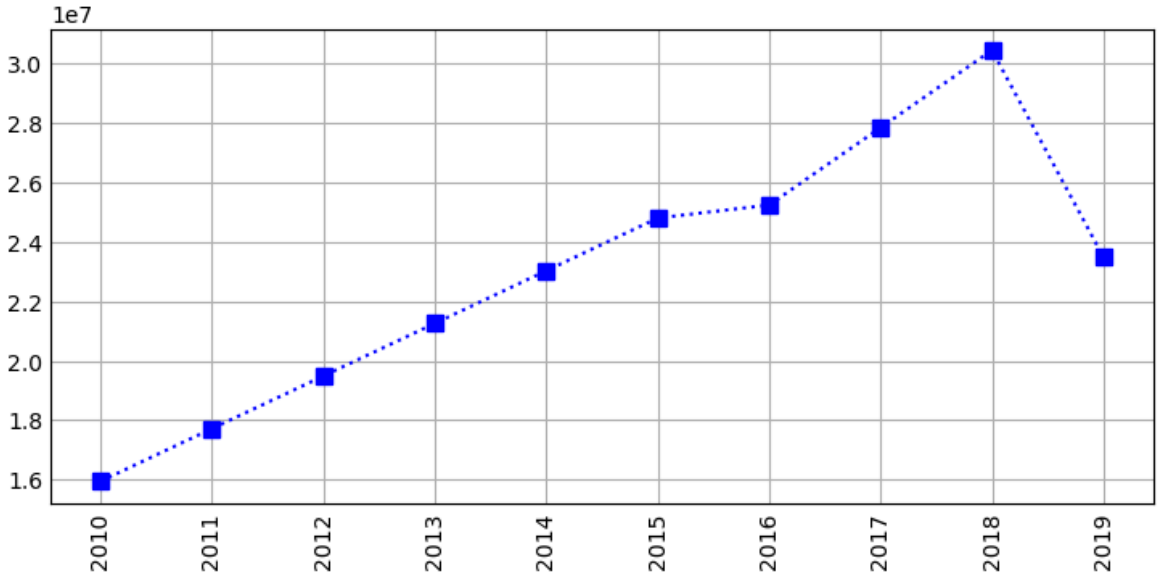
In [125…
```
%matplotlib inline
plt.rcParams['figure.figsize'] = 7, 3
plt.plot(Salary[0], c = 'blue', ls = ':', marker = 's', ms = 7)
plt.xticks(list(range(0,10)), Seasons)
plt.show()
```



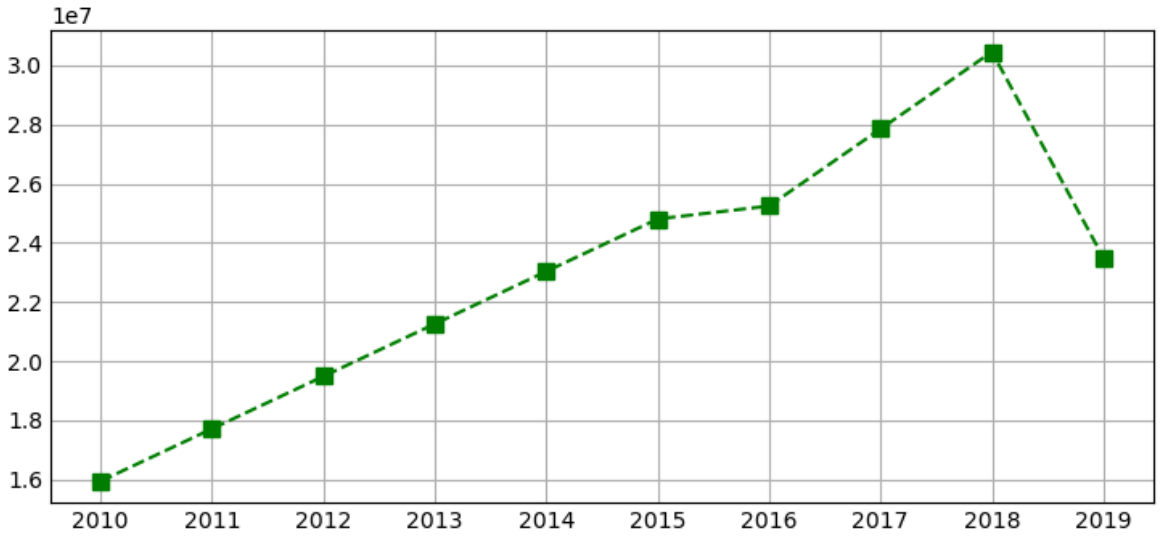
- xticks means x-axis
- yticks means y-axis
- This graph is more meaningful as compare to above graph
- This means that player's salary increased from 2010 to 2018, then reduced

To make the YEAR in vertical

In [128…
```
plt.plot(Salary[0], c = 'blue', ls = ':', marker = 's', ms = 7)
plt.xticks(list(range(0,10)), Seasons, rotation = 'vertical')
plt.show()
```



In [129…
```
plt.plot(Salary[0], c = 'g', ls = '--', marker = 's', ms = 7, label = Players[0]
plt.xticks(list(range(0,10)), Seasons, rotation = 'horizontal')
plt.show()
```



In [130…
```
Salary[0]
```

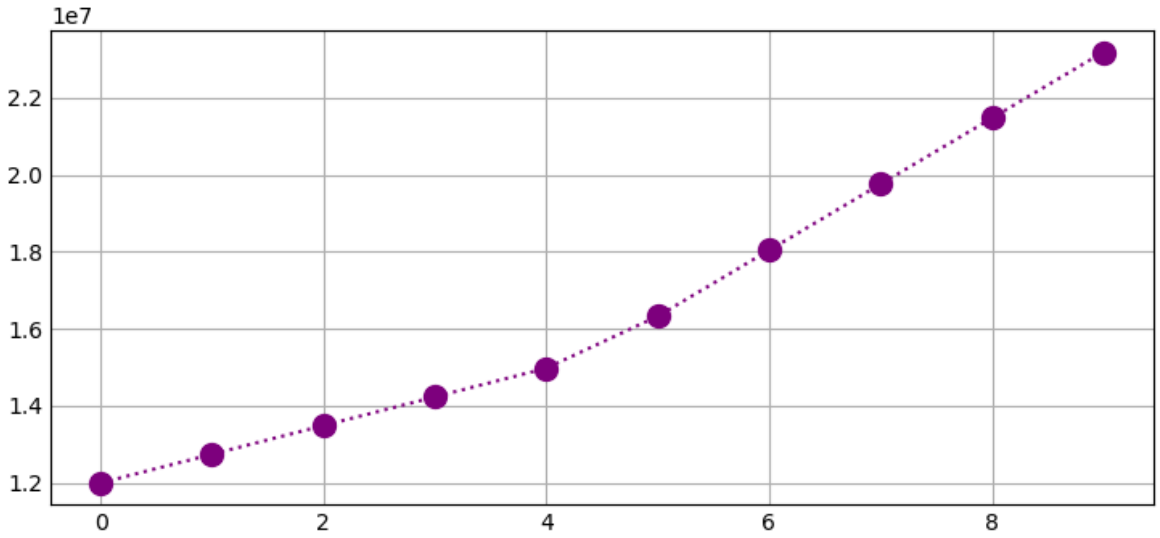
Out[130…
```
array([15946875, 17718750, 19490625, 21262500, 23034375, 24806250,
       25244493, 27849149, 30453805, 23500000])
```

In [131…
```
Salary[1]
```

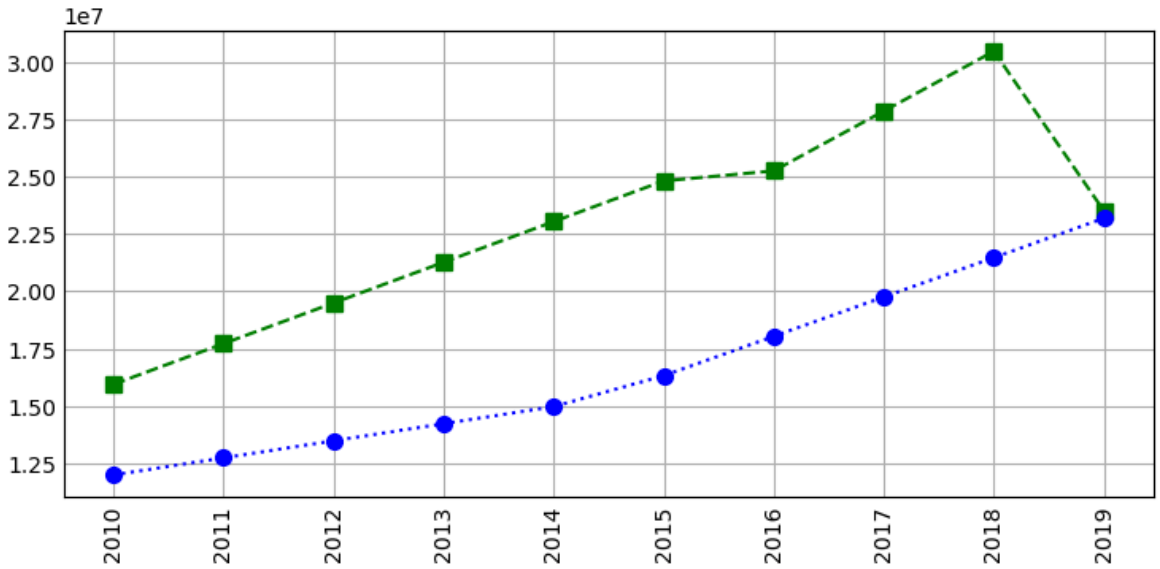
Out[131…
```
array([12000000, 12744189, 13488377, 14232567, 14976754, 16324500,
       18038573, 19752645, 21466718, 23180790])
```

To COMPARE 2 Players

In [133…
```
plt.plot(Salary[1], c = 'purple', ls = ':', marker = 'o', ms = 10, label = Playe
plt.show()
```



In [134…
```
plt.plot(Salary[0], c = 'green', ls = '--', marker = 's', ms = 7, label = Player
plt.plot(Salary[1], c = 'blue', ls = ':', marker = 'o', ms = 7, label = Players[
plt.xticks(list(range(0,10)), Seasons, rotation = 'vertical')
plt.show()
```



Insights :

- Out of the 2 players, salary of green is more than blue
- Blue is the best performer, as it never had downfall

In terms of salary, green has higher salary. In terms of performance, blue is the best performer.

Trend

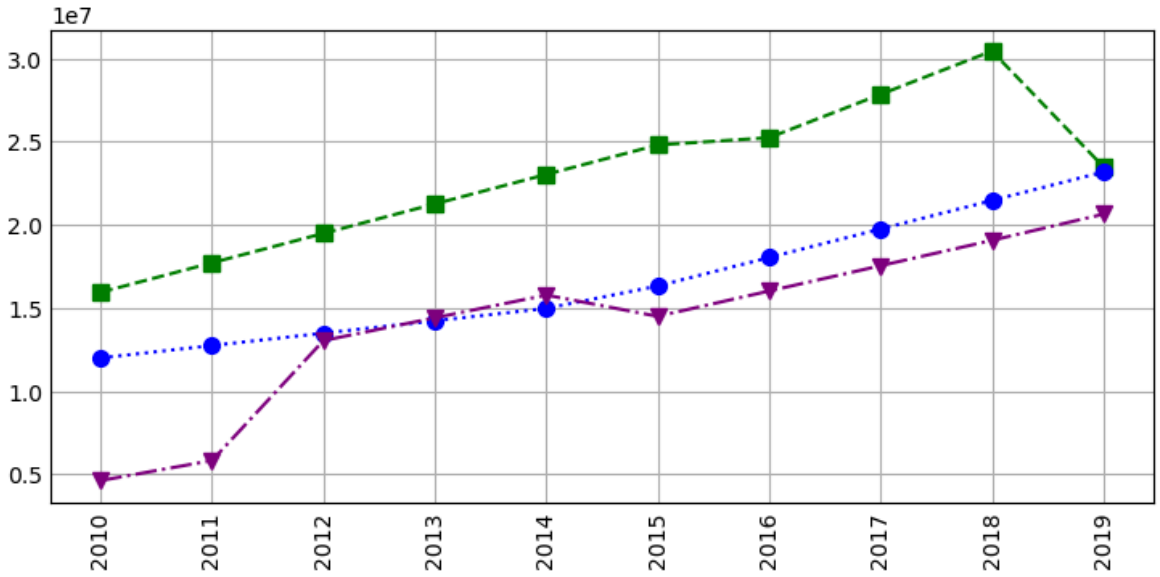
- Green is a negative trend.

- Blue is a positive trend.

Graphs are important! Entire stock market is a graph. Everywhere dashboard is required.

To COMPARE 3 Players

In [138…
```
plt.plot(Salary[0], c = 'green', ls = '--', marker = 's', ms = 7, label = Player
plt.plot(Salary[1], c = 'blue', ls = ':', marker = 'o', ms = 7, label = Players[
plt.plot(Salary[2], c = 'purple', ls = '-.', marker = 'v', ms = 7, label = Playe
plt.xticks(list(range(0,10)), Seasons, rotation = 'vertical')
plt.show()
```



For purple player, 2011 to 2012 there is a big jump in the salary.

If we write an email to the client, it will be -

Dear___,

If we compare the 3 players, there is a drastic hike in the 3rd player from 2011 to 2012. Why?

shoot questions to the client, they will like it. We always have to listen to the client, but we can raise a question.

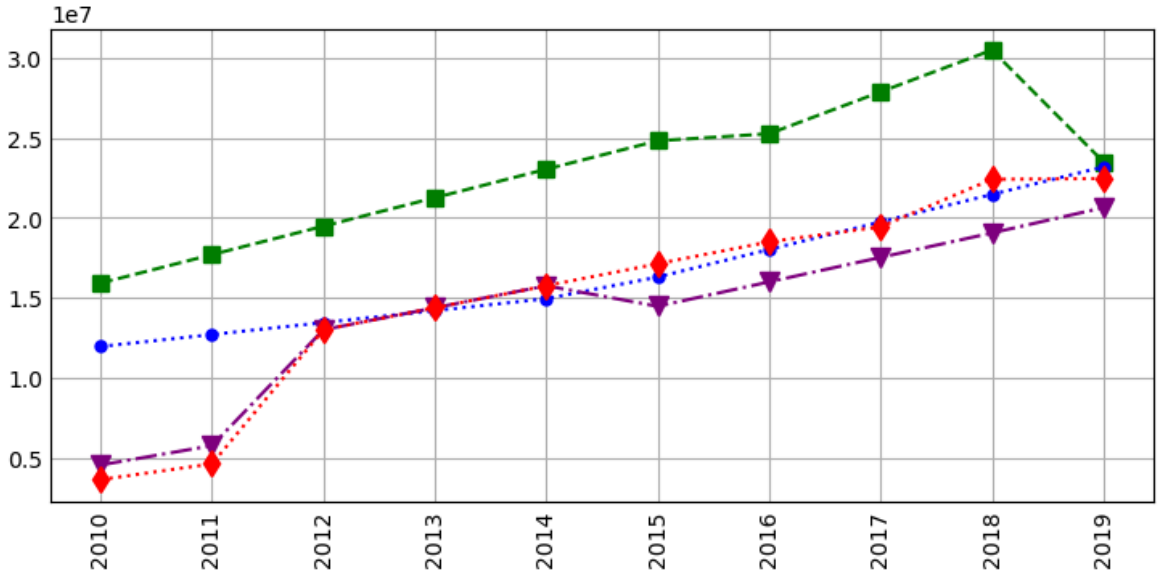
Every player has a coach/manager. Best player has best manager/coach, not others. If he is retired, may be he slows down. As thos didn't happened with other players, so he jumped up and not others. And it never happened again in rest of the years.

We need to tell client, what is green, blue and purple line

To COMPARE 4 Players

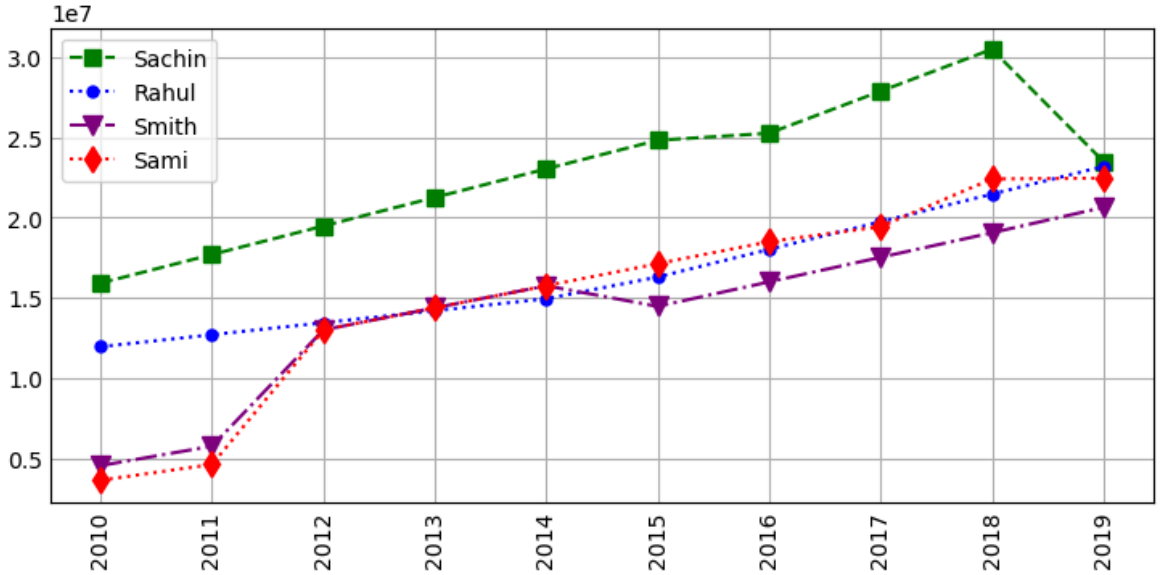
In [141…
```
plt.plot(Salary[0], c = 'green', ls = '--', marker = 's', ms = 7, label = Player
plt.plot(Salary[1], c = 'blue', ls = ':', marker = 'o', ms = 5, label = Players[
```

```
plt.plot(Salary[2], c = 'purple', ls = '-.', marker = 'v', ms = 8, label = Playe
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 [142…

```
# How to add legend in Visualisation
plt.plot(Salary[0], c = 'green', ls = '--', marker = 's', ms = 7, label = Player
plt.plot(Salary[1], c = 'blue', ls = ':', marker = 'o', ms = 5, label = Players[
plt.plot(Salary[2], c = 'purple', ls = '-.', marker = 'v', ms = 8, label = Playe
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()
```



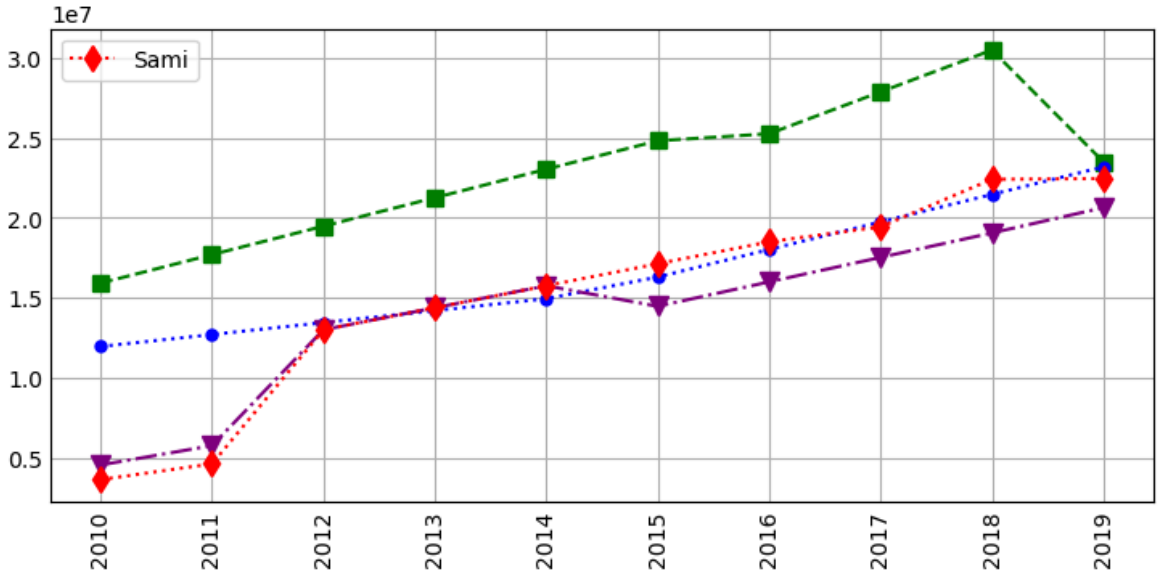
Legend creates a box, with player names for color reference. Easy for clent to understand.

plt.legend() -> Displays the color labels

legend() box should be kept outside the graph. If we will remove labels then legend won't be created. Labels and legend, both are compatible. To do that ->

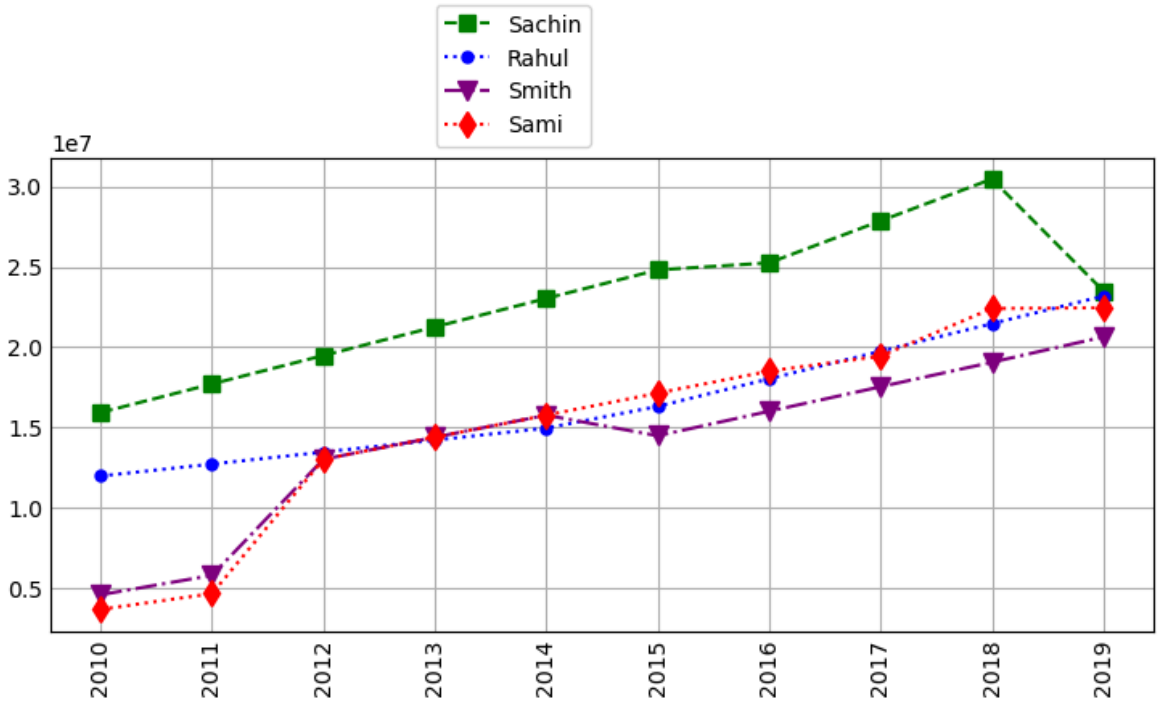
In [145…

```
# If labels are deleted then legend won't be displayed, ex. only players[3] is d
plt.plot(Salary[0], c = 'green', ls = '--', marker = 's', ms = 7) #s means squar
plt.plot(Salary[1], c = 'blue', ls = ':', marker = 'o', ms = 5) #o means round m
plt.plot(Salary[2], c = 'purple', ls = '-.', marker = 'v', ms = 8) #v means v ma
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()
```



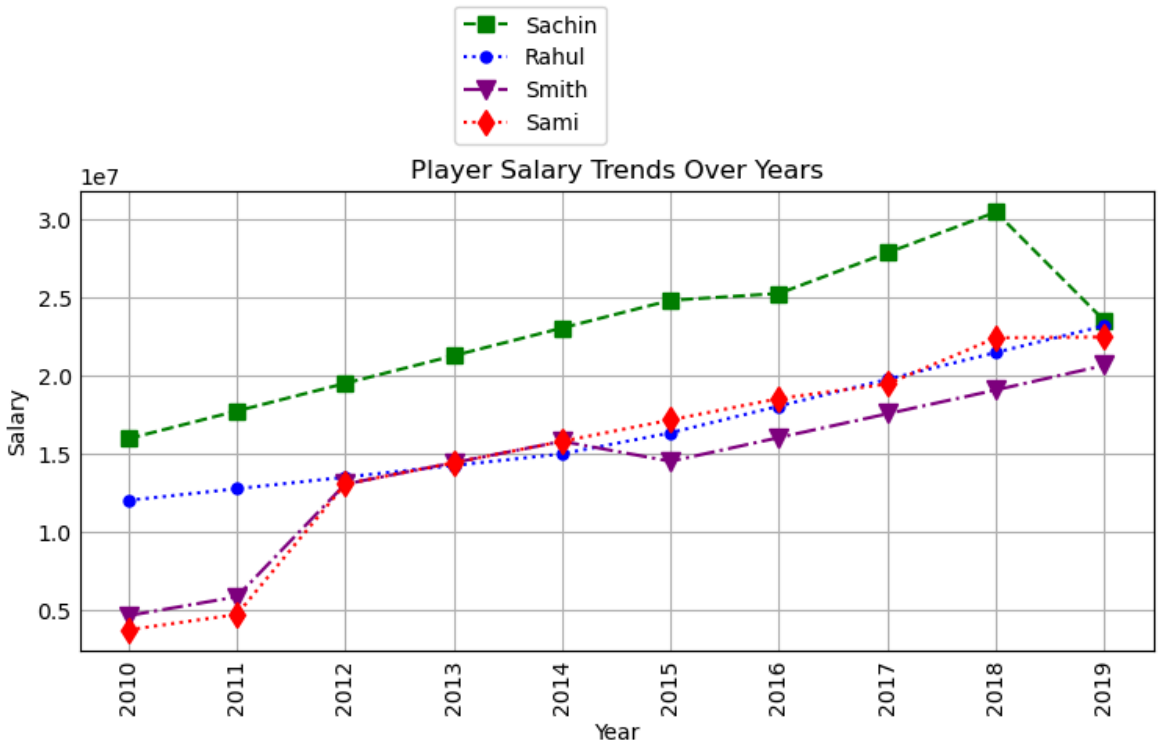
In [146…

```
# To keep legend() box outside the graph
plt.plot(Salary[0], c = 'green', ls = '--', marker = 's', ms = 7, label = Player
plt.plot(Salary[1], c = 'blue', ls = ':', marker = 'o', ms = 5, label = Players[
plt.plot(Salary[2], c = 'purple', ls = '-.', marker = 'v', ms = 8, label = Playe
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()
```



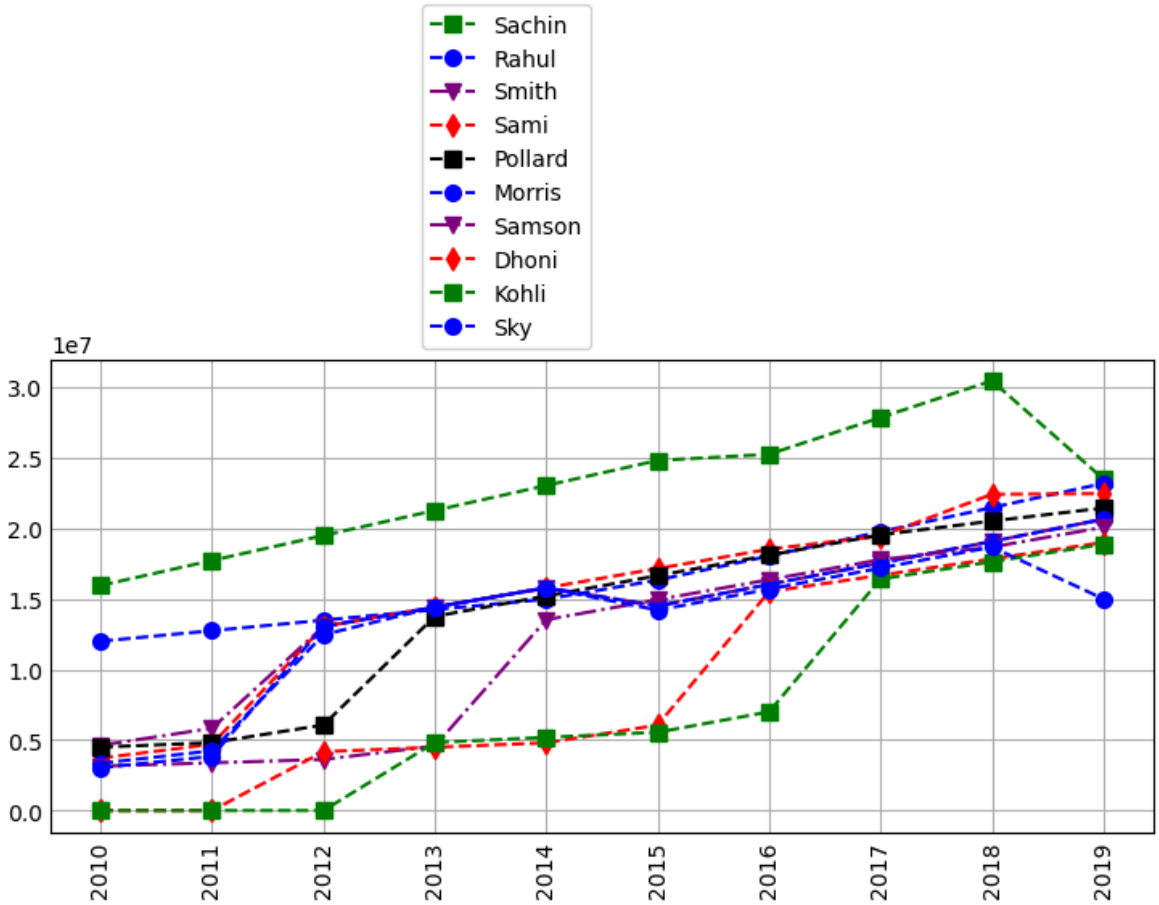
In [147…

```
# To keep legend() box outside the graph
plt.plot(Salary[0], c = 'green', ls = '--', marker = 's', ms = 7, label = Player
plt.plot(Salary[1], c = 'blue', ls = ':', marker = 'o', ms = 5, label = Players[
plt.plot(Salary[2], c = 'purple', ls = '-.', marker = 'v', ms = 8, label = Playe
plt.plot(Salary[3], c = 'red', ls = ':', marker = 'd', ms = 8, label = Players[3

plt.legend(loc = 'lower right', bbox_to_anchor=(0.5, 1.08))
plt.xticks(list(range(0,10)), Seasons, rotation = 'vertical')

plt.xlabel('Year')
plt.ylabel('Salary')
plt.title('Player Salary Trends Over Years')

plt.show()
```



In [148…

```
# To keep legend() box outside the graph
plt.plot(Salary[0], c = 'green', ls = '--', marker = 's', ms = 7, label = Player
plt.plot(Salary[1], c = 'blue', ls = '--', marker = 'o', ms = 7, label = Players
plt.plot(Salary[2], c = 'purple', ls = '-.', marker = 'v', ms = 7, label = Playe
plt.plot(Salary[3], c = 'red', ls = '--', marker = 'd', ms = 7, label = Players[
plt.plot(Salary[4], c = 'black', ls = '--', marker = 's', ms = 7, label = Player
plt.plot(Salary[5], c = 'blue', ls = '--', marker = 'o', ms = 7, label = Players
plt.plot(Salary[6], c = 'purple', ls = '-.', marker = 'v', ms = 7, label = Playe
plt.plot(Salary[7], c = 'red', ls = '--', marker = 'd', ms = 7, label = Players[
plt.plot(Salary[8], c = 'green', ls = '--', marker = 's', ms = 7, label = Player
plt.plot(Salary[9], c = 'blue', ls = '--', marker = 'o', ms = 7, label = Players

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

plt.show()
```



- If all the players are added, the graph is dirty.
- If the data visualisation has more data, then Python program can't visualise properly. Thus, we need to introduce BUSINESS INTELLIGENCE Tools such as Power BI/Tableau
- If the data is BIG, python is OK for data cleaning but NOT for dashboards. That's why we need to refer to Tableau and Power BI.
- Python is a Programming language. In Programming language, we can visualise limited data.

In today's session, we learned -

- Matrices
- Building Matrices - np.reshape
- Dictionary in Python
- Visualising using Pyplot
- IPL Data Analysis