# Python Libraries (Numpy, Pandas, Matplotlib)

June 19, 2018

## 1 Bibliotecas de Análise de Dados

Mais informações em: Ipython Notebook website

```
pip install ipython
```

### 1.1 Numpy

Mais informações em: Numpy website

```
pip install numpy
```

```
In [5]: import numpy as np

In [6]: 2.1 * np.array([1, 2, 3])

Out[6]: array([ 2.1,  4.2,  6.3])

In [7]: # help(np.array)
        array = list(range(10))
        np_array = np.array(array)
        np_array

Out[7]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [11]: array = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12.]]
         np_array = np.array(array)
         np_array

Out[11]: array([[  1.,   2.,   3.,   4.],
                [  5.,   6.,   7.,   8.],
                [  9.,  10.,  11.,  12.]])

In [12]: # help(np.arange)
         np.arange(2, 10, 2)

Out[12]: array([2, 4, 6, 8])

In [14]: # array de zeros
         np.zeros(10)
```

```
Out[14]: array([ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.])

In [16]: # array de uns
         np.ones(10)

Out[16]: array([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.])

In [17]: # array de 2's
         2 * np.ones(10)

Out[17]: array([ 2.,  2.,  2.,  2.,  2.,  2.,  2.,  2.,  2.,  2.])

In [19]: # array de 2 dimensões
         np.ones((4, 2)) # ou de mais dimensões

Out[19]: array([[ 1.,  1.],
                [ 1.,  1.],
                [ 1.,  1.],
                [ 1.,  1.]])

In [27]: # array de números aleatórios
         np.random.rand(5)

Out[27]: array([ 0.89592011,  0.13217349,  0.265455  ,  0.46261907,  0.48048668])

In [32]: # array de números inteiros aleatórios
         np.random.randint(1, 100, 5)

Out[32]: array([16, 17, 17,  8, 73])

In [39]: # exemplo 1
         array = np.random.randint(1, 100, (4, 3))
         array

Out[39]: array([[99, 69,  6],
                [34, 69, 56],
                [77, 45, 33],
                [78, 59, 55]])

In [40]: array.shape

Out[40]: (4, 3)

In [43]: array.reshape(3, 4)

Out[43]: array([[99, 69,  6, 34],
                [69, 56, 77, 45],
                [33, 78, 59, 55]])

In [44]: array.max() # array.min()
```

```
Out[44]: 99

In [45]: array.argmax()

Out[45]: 0

In [50]: array.reshape(4, 3)

Out[50]: array([[99, 69,  6],
                [34, 69, 56],
                [77, 45, 33],
                [78, 59, 55]])

In [49]: array2 = np.eye(5)
         array2

Out[49]: array([[ 1.,  0.,  0.,  0.,  0.],
                [ 0.,  1.,  0.,  0.,  0.],
                [ 0.,  0.,  1.,  0.,  0.],
                [ 0.,  0.,  0.,  1.,  0.],
                [ 0.,  0.,  0.,  0.,  1.]])

In [51]: array[1, 2]

Out[51]: 56

In [54]: array[2:]

Out[54]: array([[77, 45, 33],
                [78, 59, 55]])

In [55]: array[2:] = 0
         array

Out[55]: array([[99, 69,  6],
                [34, 69, 56],
                [ 0,  0,  0],
                [ 0,  0,  0]])

In [56]: array[1:, 2:]

Out[56]: array([[56],
                [ 0],
                [ 0]])

In [57]: # exemplo 2
         array = np.linspace(9, 10, 11) # [9, 10] dividido em 11 pontos
         array

Out[57]: array([  9. ,   9.1,   9.2,   9.3,   9.4,   9.5,   9.6,   9.7,   9.8,
                  9.9,  10. ])
```

```
In [81]: array > 9.5

Out[81]: array([False, False, False, False, False, False,  True,  True,  True,
                True,  True], dtype=bool)

In [58]: array[array > 9.5]

Out[58]: array([  9.6,   9.7,   9.8,   9.9,  10. ])

In [61]: # array
         array + array

Out[61]: array([ 18. ,  18.2,  18.4,  18.6,  18.8,  19. ,  19.2,  19.4,  19.6,
                19.8,  20. ])

In [62]: 3 * array

Out[62]: array([ 27. ,  27.3,  27.6,  27.9,  28.2,  28.5,  28.8,  29.1,  29.4,
                29.7,  30. ])

In [63]: array * array

Out[63]: array([  81.  ,   82.81,   84.64,   86.49,   88.36,   90.25,   92.16,
                 94.09,   96.04,   98.01,  100.  ])

In [64]: array / array

Out[64]: array([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.])

In [65]: 1 / array

Out[65]: array([ 0.11111111,  0.10989011,  0.10869565,  0.10752688,  0.10638298,
                0.10526316,  0.10416667,  0.10309278,  0.10204082,  0.1010101 ,
                0.1       ])

In [66]: np.sqrt(array)

Out[66]: array([ 3.        ,  3.01662063,  3.03315018,  3.04959014,  3.06594194,
                3.082207  ,  3.09838668,  3.1144823 ,  3.13049517,  3.14642654,
                3.16227766])

In [91]: np.exp(array)

Out[91]: array([  8103.08392758,   8955.29270348,   9897.12905874,  10938.01920817,
                12088.38073022,  13359.72682966,  14764.78156558,  16317.60719802,
                18033.74492783,  19930.37043823,  22026.46579481])

In [92]: np.sin(array)

Out[92]: array([ 0.41211849,  0.31909836,  0.22288991,  0.12445442,  0.02477543,
               -0.07515112, -0.17432678, -0.27176063, -0.36647913, -0.45753589,
               -0.54402111])
```

```
In [67]: # o que faz?
         np.arange(1, 101).reshape(10, 10) / 100
```

```
Out[67]: array([[ 0.01,  0.02,  0.03,  0.04,  0.05,  0.06,  0.07,  0.08,  0.09,  0.1 ],
                [ 0.11,  0.12,  0.13,  0.14,  0.15,  0.16,  0.17,  0.18,  0.19,  0.2 ],
                [ 0.21,  0.22,  0.23,  0.24,  0.25,  0.26,  0.27,  0.28,  0.29,  0.3 ],
                [ 0.31,  0.32,  0.33,  0.34,  0.35,  0.36,  0.37,  0.38,  0.39,  0.4 ],
                [ 0.41,  0.42,  0.43,  0.44,  0.45,  0.46,  0.47,  0.48,  0.49,  0.5 ],
                [ 0.51,  0.52,  0.53,  0.54,  0.55,  0.56,  0.57,  0.58,  0.59,  0.6 ],
                [ 0.61,  0.62,  0.63,  0.64,  0.65,  0.66,  0.67,  0.68,  0.69,  0.7 ],
                [ 0.71,  0.72,  0.73,  0.74,  0.75,  0.76,  0.77,  0.78,  0.79,  0.8 ],
                [ 0.81,  0.82,  0.83,  0.84,  0.85,  0.86,  0.87,  0.88,  0.89,  0.9 ],
                [ 0.91,  0.92,  0.93,  0.94,  0.95,  0.96,  0.97,  0.98,  0.99,  1.  ]])
```

## 1.2 Pandas

Mais informações em: Pandas website

```
pip install pandas
```

```
In [69]: import pandas as pd
```

```
In [74]: # leitura .csv
         df = pd.read_csv('example3.csv', index_col=0) # index_col=0
         df
```

```
Out[74]:     a   b   c   d
         0   0   1   2   3
         1   4   5   6   7
         2   8   9  10  11
         3  12  13  14  15
```

```
In [73]: # escrita .csv
         df.to_csv('example3.csv') # index=False
```

```
In [101]: # help(pd.DataFrame)
          df = pd.DataFrame(
              np.random.randint(1, 100, (5, 4)),
              index=['A', 'B', 'C', 'D', 'E'],
              columns=['W', 'X', 'Y', 'Z'])
          # verificar df
          df
```

```
Out[101]:     W   X   Y   Z
          A  80  90  21  48
          B  87   2  14  53
          C  36  32  56  66
          D  11  97  14  69
          E  20  59  46   3
```

```
In [79]: # acesso pela coluna
         df['W']

Out[79]: A    73
         B    87
         C    36
         D    16
         E    85
         Name: W, dtype: int64

In [83]: df[['W', 'Z', 'X']]

Out[83]:     W   Z   X
         A  73  71  67
         B  87  89  48
         C  36  93  18
         D  16  89  35
         E  85  88  87

In [85]: # df
         df.loc['A']

Out[85]: W    73
         X    67
         Y    81
         Z    71
         Name: A, dtype: int64

In [86]: df.loc[['A', 'B']]

Out[86]:     W   X   Y   Z
         A  73  67  81  71
         B  87  48  69  89

In [88]: #df
         df.loc['A', 'W']

Out[88]: 73

In [89]: df[['W', 'Y']].loc[['A', 'B']]

Out[89]:     W   Y
         A  73  81
         B  87  69

In [90]: # adicionar colunas
         df['NEW'] = df['X'] + df['W']
         df
```

```
Out[90]:     W   X   Y   Z  NEW
         A  73  67  81  71  140
         B  87  48  69  89  135
         C  36  18  49  93   54
         D  16  35  10  89   51
         E  85  87  47  88  172

In [91]: # remover colunas
         df.drop('NEW', axis = 1)

Out[91]:     W   X   Y   Z
         A  73  67  81  71
         B  87  48  69  89
         C  36  18  49  93
         D  16  35  10  89
         E  85  87  47  88

In [94]: df
         # df.drop('NEW', axis=1, inplace=True)

Out[94]:     W   X   Y   Z
         A  73  67  81  71
         B  87  48  69  89
         C  36  18  49  93
         D  16  35  10  89
         E  85  87  47  88

In [99]: # df.drop(labels=['A', 'B', 'C'], axis=0, inplace=True)
         df

Out[99]:     W   X   Y   Z
         D  16  35  10  89
         E  85  87  47  88

In [137]: # df
          df > 50

Out[137]:        W      X      Y      Z
          A   True   True   True   True
          B  False  False  False   True
          C   True  False   True   True
          D   True  False  False  False
          E   True   True   True  False

In [140]: # df
          # df[df > 50]

In [141]: # df
          df [[True, True, False, False, False]]
```

```
Out[141]:     W   X   Y   Z
          A  74  63  95  70
          B  32  12  29  74

In [103]: # df
          df[df['W'] > 50]

Out[103]:     W   X   Y   Z
          A  80  90  21  48
          B  87   2  14  53

In [104]: df[(df['W'] > 40) & (df['X'] > 50)]

Out[104]:     W   X   Y   Z
          A  80  90  21  48

In [105]: df[df['W'] > 50][['Y', 'Z']]

Out[105]:     Y   Z
          A  21  48
          B  14  53

In [108]: df[df > 50]
          # df1 = df[df > 50].fillna(value=0.0)
          df1

Out[108]:       W     X     Y     Z
          A  80.0  90.0   0.0   0.0
          B  87.0   0.0   0.0  53.0
          C   0.0   0.0  56.0  66.0
          D   0.0  97.0   0.0  69.0
          E   0.0  59.0   0.0   0.0

In [112]: # exemplo 2 pandas
          df2 = pd.DataFrame(
              np.random.randint(1, 100, (5, 4)),
              index=['A2', 'B2', 'C2', 'D2', 'E2'],
              columns=['W', 'X', 'Y', 'Z'])
          df2

Out[112]:     W   X   Y   Z
          A2  86  22  79  30
          B2  44  76   6  91
          C2  17  20  13  69
          D2  88  74  60  17
          E2  37  47  35  21

In [113]: # concatenar dataframes
          pd.concat([df1, df2]) # default
```

Out[113]:

| | W | X | Y | Z |
|---|---|---|---|---|
| A | 80.0 | 90.0 | 0.0 | 0.0 |
| B | 87.0 | 0.0 | 0.0 | 53.0 |
| C | 0.0 | 0.0 | 56.0 | 66.0 |
| D | 0.0 | 97.0 | 0.0 | 69.0 |
| E | 0.0 | 59.0 | 0.0 | 0.0 |
| A2 | 86.0 | 22.0 | 79.0 | 30.0 |
| B2 | 44.0 | 76.0 | 6.0 | 91.0 |
| C2 | 17.0 | 20.0 | 13.0 | 69.0 |
| D2 | 88.0 | 74.0 | 60.0 | 17.0 |
| E2 | 37.0 | 47.0 | 35.0 | 21.0 |

In [114]:
```
# df2
df2.columns = ['W2', 'X2', 'Y2', 'Z2']
df2.index = ['A', 'B', 'C', 'D', 'E']
df2
```

Out[114]:

| | W2 | X2 | Y2 | Z2 |
|---|---|---|---|---|
| A | 86 | 22 | 79 | 30 |
| B | 44 | 76 | 6 | 91 |
| C | 17 | 20 | 13 | 69 |
| D | 88 | 74 | 60 | 17 |
| E | 37 | 47 | 35 | 21 |

In [115]:
```
pd.concat([df1, df2], axis=1)
```

Out[115]:

| | W | X | Y | Z | W2 | X2 | Y2 | Z2 |
|---|---|---|---|---|---|---|---|---|
| A | 80.0 | 90.0 | 0.0 | 0.0 | 86 | 22 | 79 | 30 |
| B | 87.0 | 0.0 | 0.0 | 53.0 | 44 | 76 | 6 | 91 |
| C | 0.0 | 0.0 | 56.0 | 66.0 | 17 | 20 | 13 | 69 |
| D | 0.0 | 97.0 | 0.0 | 69.0 | 88 | 74 | 60 | 17 |
| E | 0.0 | 59.0 | 0.0 | 0.0 | 37 | 47 | 35 | 21 |

In [116]:
```
df = pd.DataFrame({
    'A': [1, 2, 2, 2, 1, 5, 5, 6],
    'B': [1, 2, 1, 2, 1, 2, 1, 2]
})
df
```

Out[116]:

| | A | B |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 2 | 2 |
| 2 | 2 | 1 |
| 3 | 2 | 2 |
| 4 | 1 | 1 |
| 5 | 5 | 2 |
| 6 | 5 | 1 |
| 7 | 6 | 2 |

In [117]:
```
df.describe() # describe, max, min, count, mean
```

```
Out[117]:            A         B
          count  8.00  8.000000
          mean   3.00  1.500000
          std    2.00  0.534522
          min    1.00  1.000000
          25%    1.75  1.000000
          50%    2.00  1.500000
          75%    5.00  2.000000
          max    6.00  2.000000

In [217]: df['A'].sum() # sum, max, min, count, std, mean, count, describe

Out[217]: 24

In [118]: # funções data frames
          df['A']

Out[118]: 0    1
          1    2
          2    2
          3    2
          4    1
          5    5
          6    5
          7    6
          Name: A, dtype: int64

In [119]: df['A'].unique()

Out[119]: array([1, 2, 5, 6])

In [193]: df['A'].nunique()

Out[193]: 4

In [120]: df['A'].value_counts()

Out[120]: 2    3
          5    2
          1    2
          6    1
          Name: A, dtype: int64
```

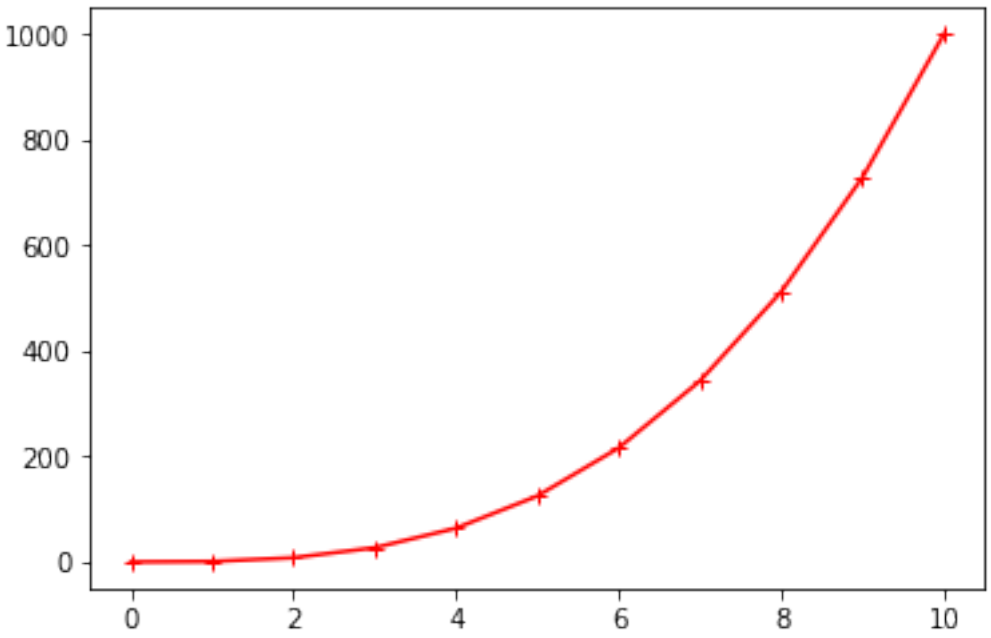
Mais informações em: Pandas 10min

### 1.3 Matplotlib

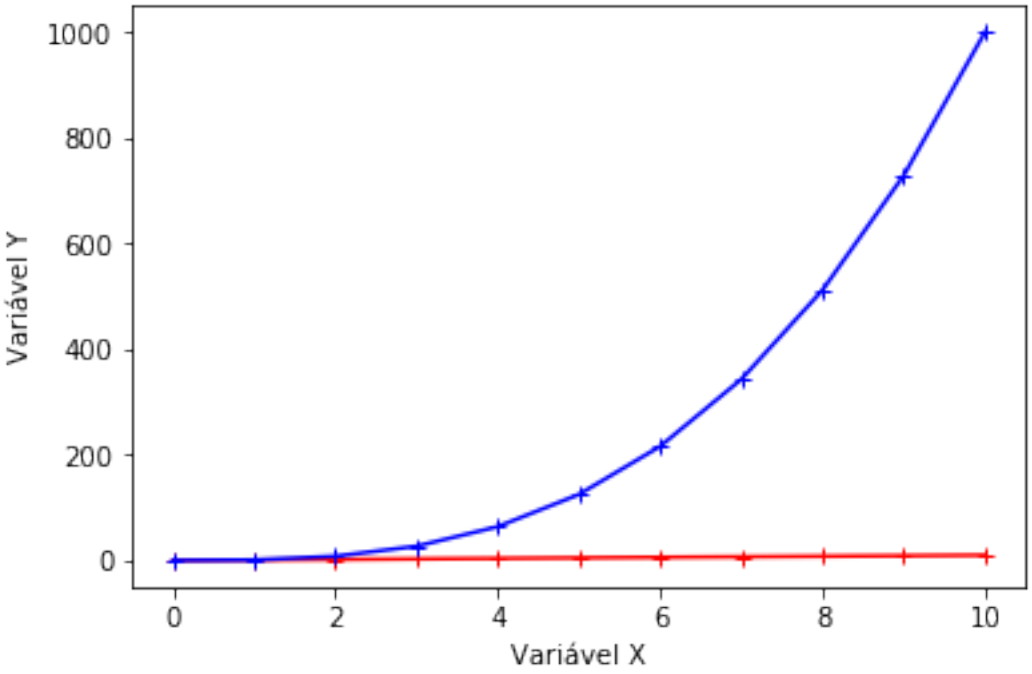
Mais informações em: Matplotlib website

```
pip install matplotlib
```

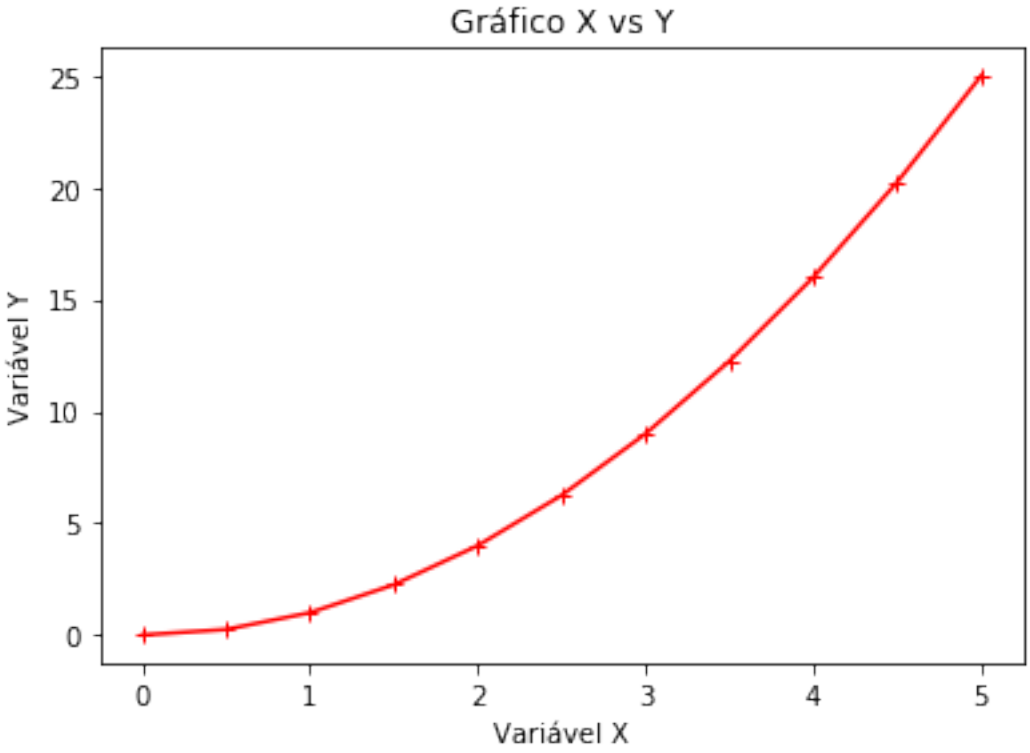
```
In [121]: import matplotlib.pyplot as plt
          %matplotlib inline

In [124]: x = np.linspace(0, 10, 11)
          x

Out[124]: array([  0.,   1.,   2.,   3.,   4.,   5.,   6.,   7.,   8.,   9.,  10.])

In [127]: y = x**3
          y

Out[127]: array([   0.,    1.,    8.,   27.,   64.,  125.,  216.,  343.,
                  512.,  729., 1000.])

In [132]: plt.plot(x, y, 'r+-') # bo, b-, bx, r+

Out[132]: [<matplotlib.lines.Line2D at 0x7fddf6a2cc88>]
```




```
In [139]: plt.xlabel('Variável X')
          plt.ylabel('Variável Y')
          plt.plot(x, x, 'r+-')
          plt.plot(x, y, 'b+-')
          # plt.show()

Out[139]: [<matplotlib.lines.Line2D at 0x7fddf67fbd30>]
```

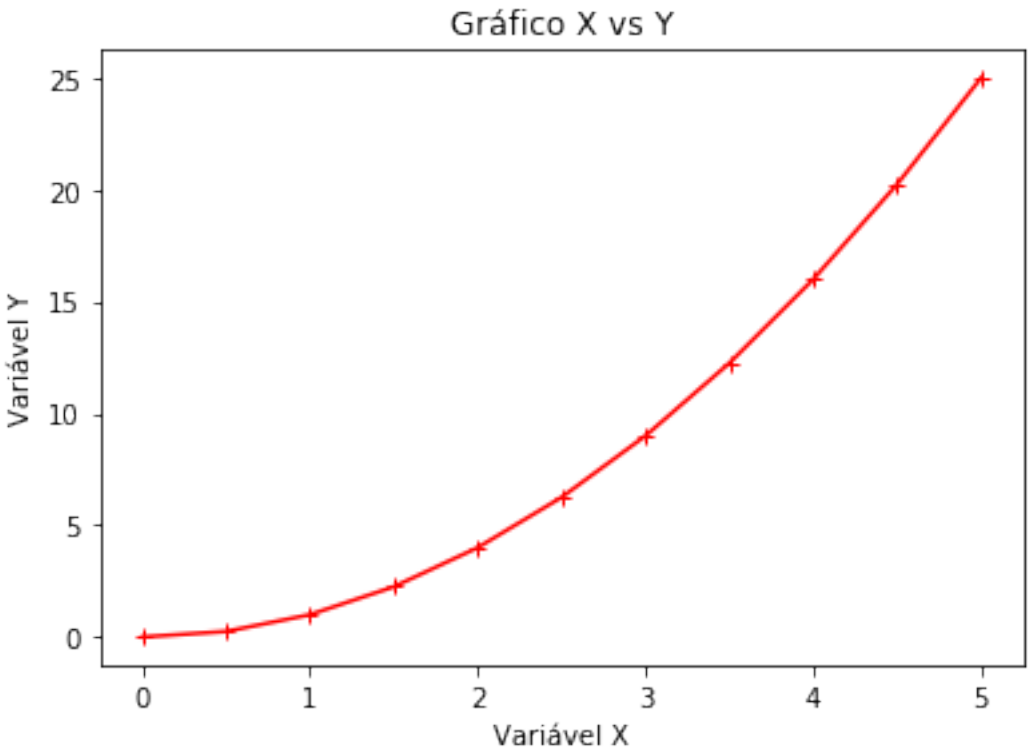



```
In [257]: fig = plt.figure()
          axes = fig.add_axes([0.1, 0.1, 0.8, 0.8]) # left, bottom, width, height
          axes.plot(x, y, 'r+-')
          axes.set_xlabel('Variável X')
          axes.set_ylabel('Variável Y')
          axes.set_title('Gráfico X vs Y')
          # fig.show()

Out[257]: Text(0.5,1,'Gráfico X vs Y')
```

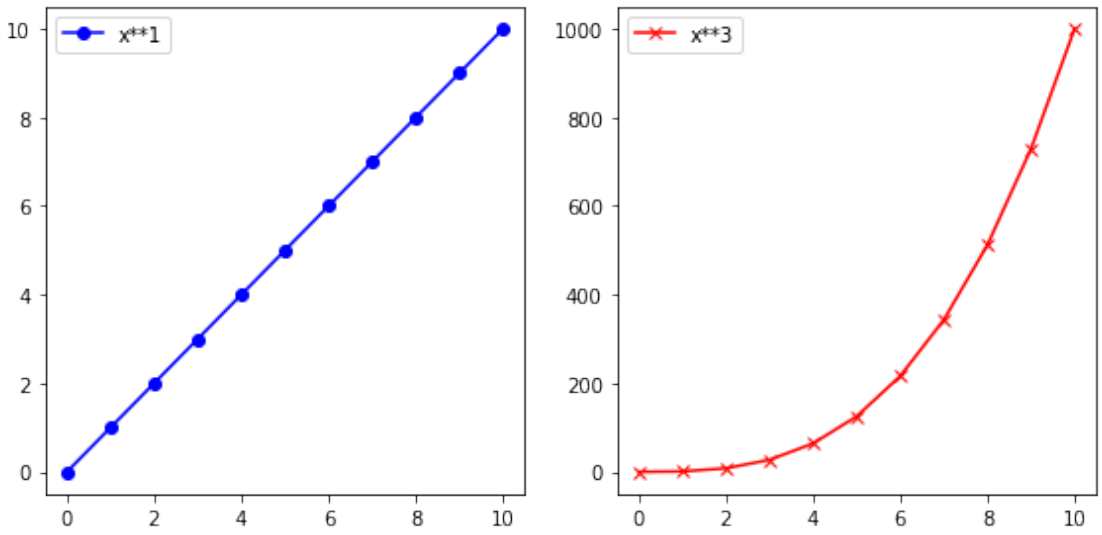



```
In [258]: fig
```

```
Out[258]:
```




```
In [140]: fig, axes = plt.subplots(nrows = 1, ncols = 2, figsize=(8, 4)) # figsize=(8, 4)
          ax1, ax2 = axes
          ax1.plot(x, x, 'bo-', label='x**1')
          ax2.plot(x, y, 'rx-', label='x**3')
          plt.tight_layout()
          ax1.legend(loc=2) #
          ax2.legend(loc=2)

Out[140]: <matplotlib.legend.Legend at 0x7fddf673de48>
```




```
In [141]: fig.savefig('figura.png')
```

## 1.4 Pandas (Visualização de Dados)

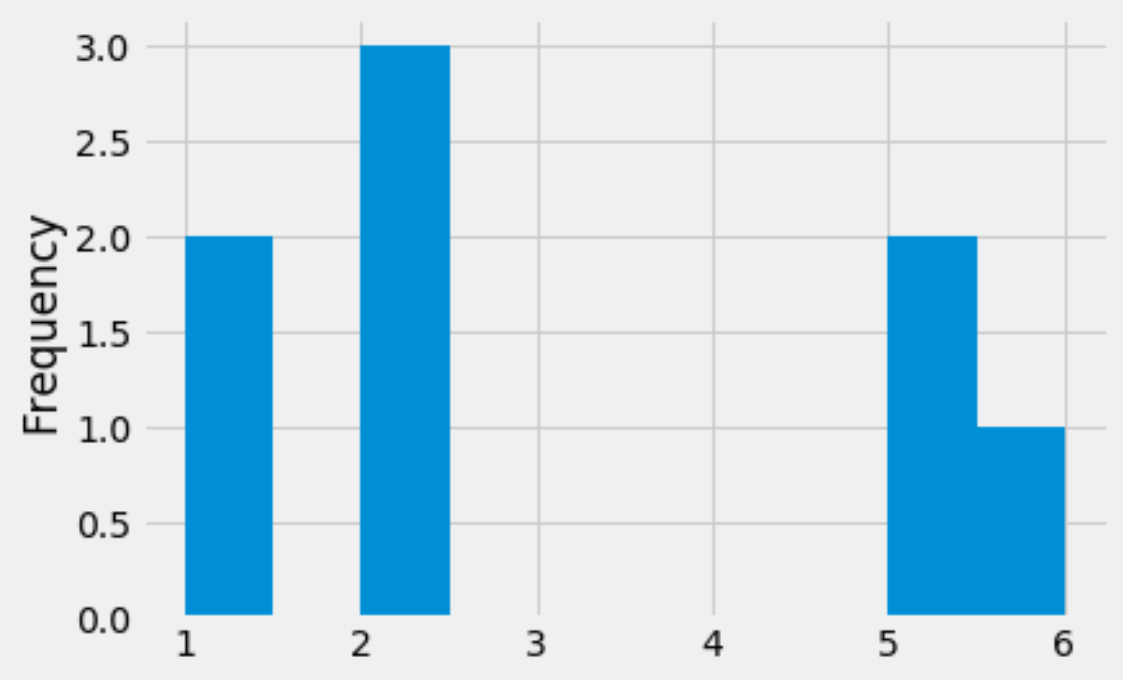
```
In [301]: # visualização de dados com pandas
          df
          # df['A']
          # df['A'].value_counts()
```

```
Out[301]:    A  B
          0  1  1
          1  2  2
          2  2  1
          3  2  2
          4  1  1
          5  5  2
          6  5  1
          7  6  2
```

```
In [149]: # vários estilos
          plt.style.use('fivethirtyeight') # ggplot, bmh, fivethirtyeight
```

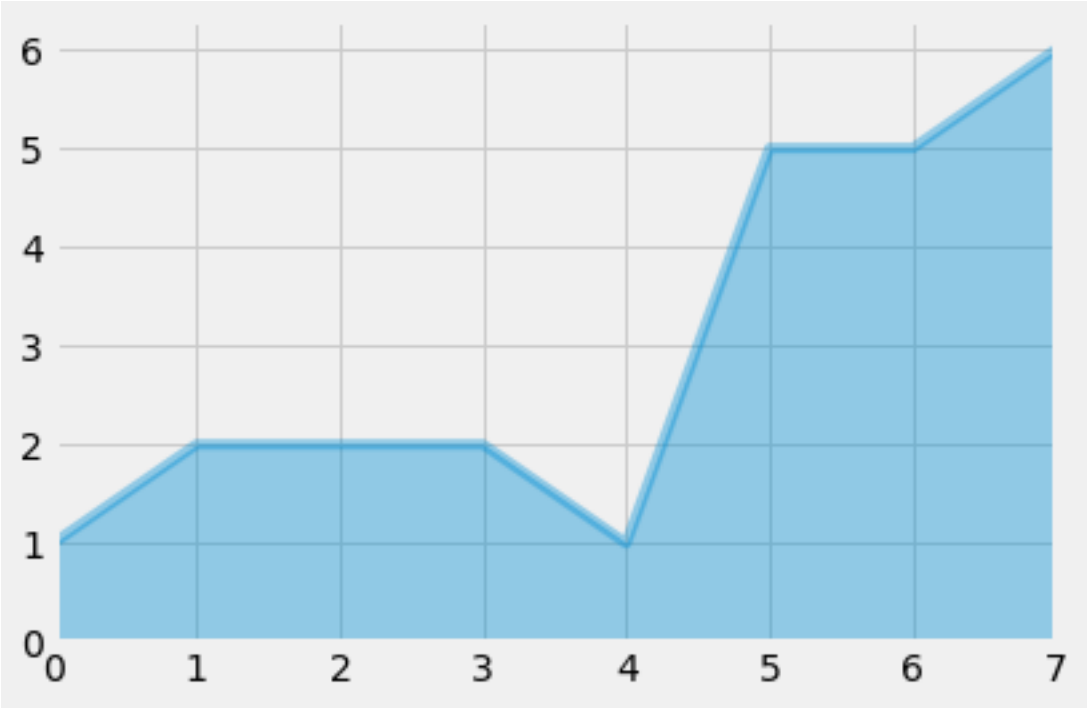
```
In [146]: df['A'].plot.hist()
```

```
Out[146]: <matplotlib.axes._subplots.AxesSubplot at 0x7fddf6692198>
```



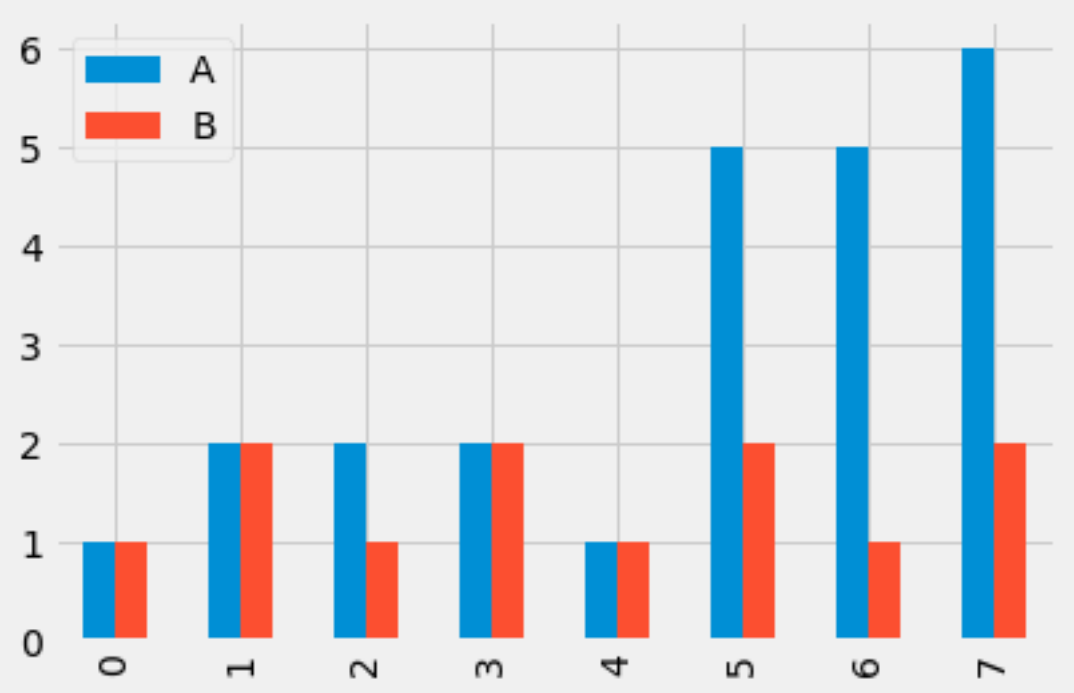

```
In [147]: df['A'].plot.area(alpha=0.4)
```

```
Out[147]: <matplotlib.axes._subplots.AxesSubplot at 0x7fddf65ff470>
```



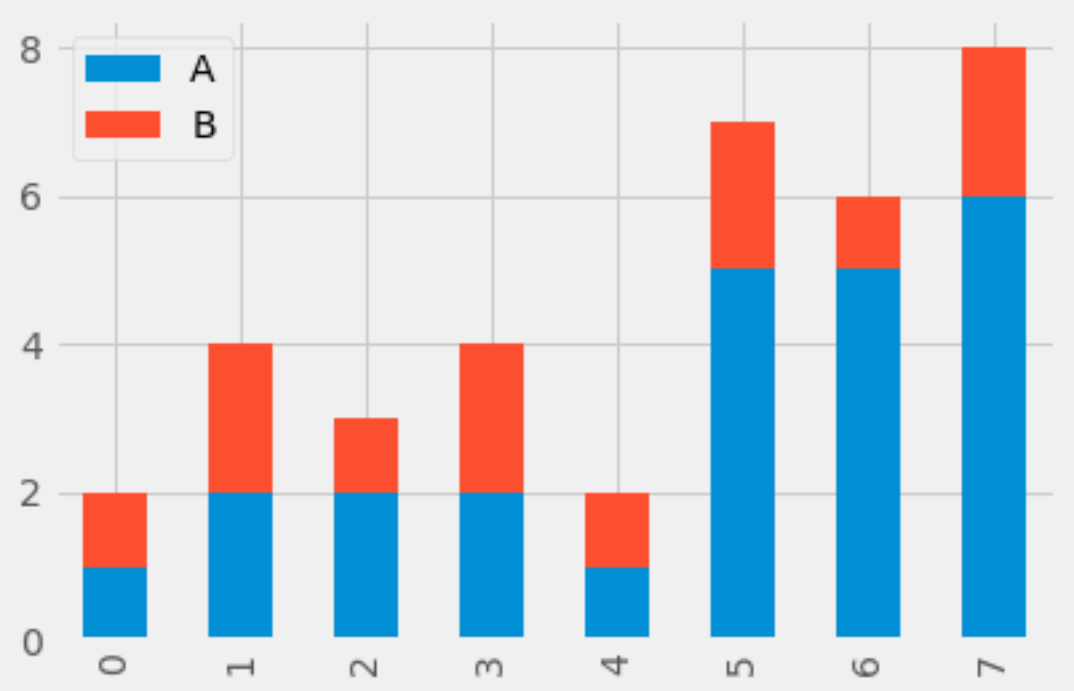
```
In [148]: df.plot.bar()
```

```
Out[148]: <matplotlib.axes._subplots.AxesSubplot at 0x7fddf65e1cc0>
```

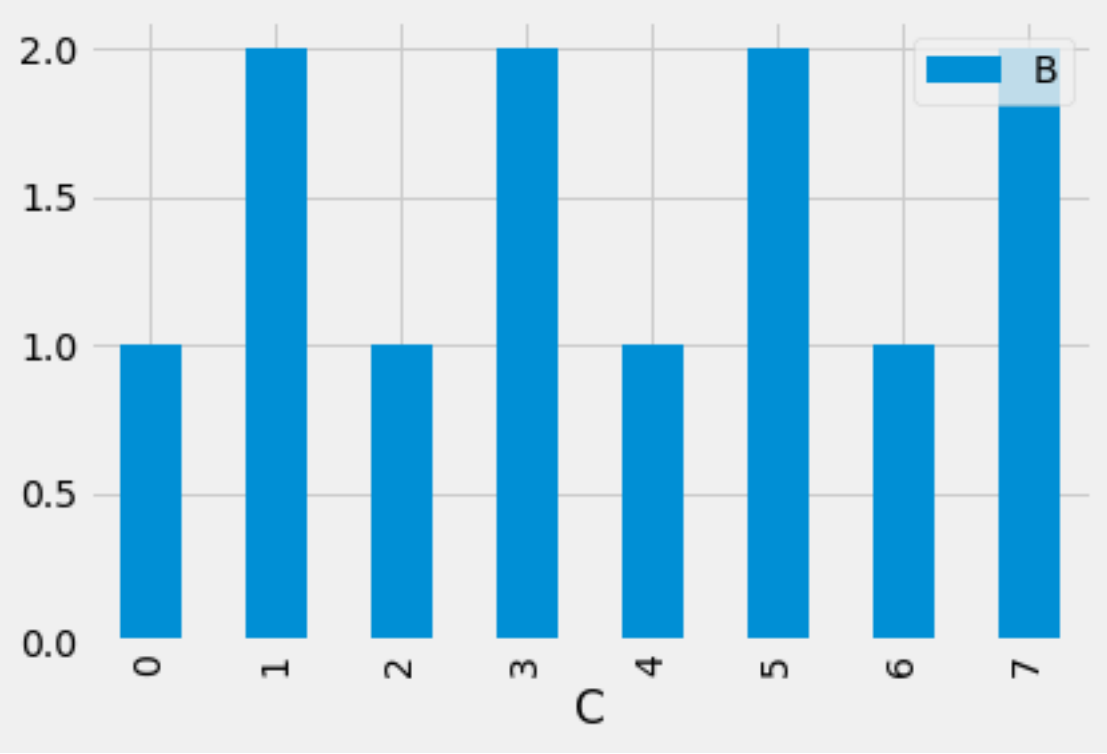



```
In [312]: df.plot.bar(stacked=True)
```

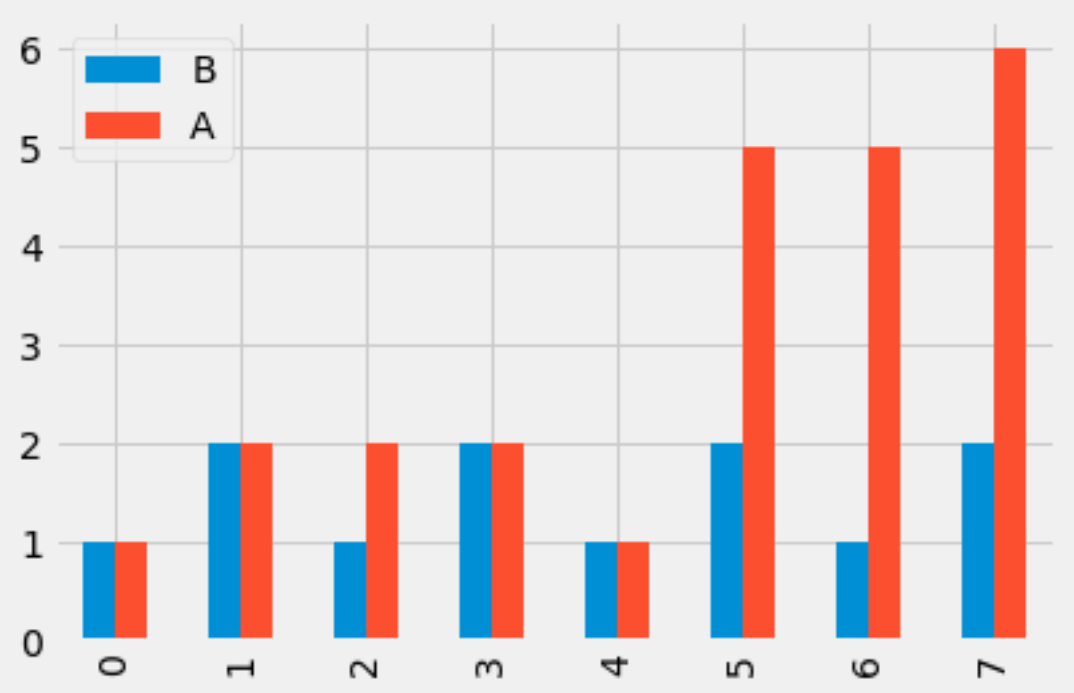
```
Out[312]: <matplotlib.axes._subplots.AxesSubplot at 0x7f17d1746588>
```



```
In [150]: # df
          df['C'] = np.arange(0, 8)
          df

Out[150]:    A  B  C
          0  1  1  0
          1  2  2  1
          2  2  1  2
          3  2  2  3
          4  1  1  4
          5  5  2  5
          6  5  1  6
          7  6  2  7

In [151]: df.plot.bar(x='C', y='B')

Out[151]: <matplotlib.axes._subplots.AxesSubplot at 0x7fddf65b8208>
```




```
In [152]: df.plot.bar(x=df.index, y=['B', 'A'])

Out[152]: <matplotlib.axes._subplots.AxesSubplot at 0x7fddf655ccc0>
```



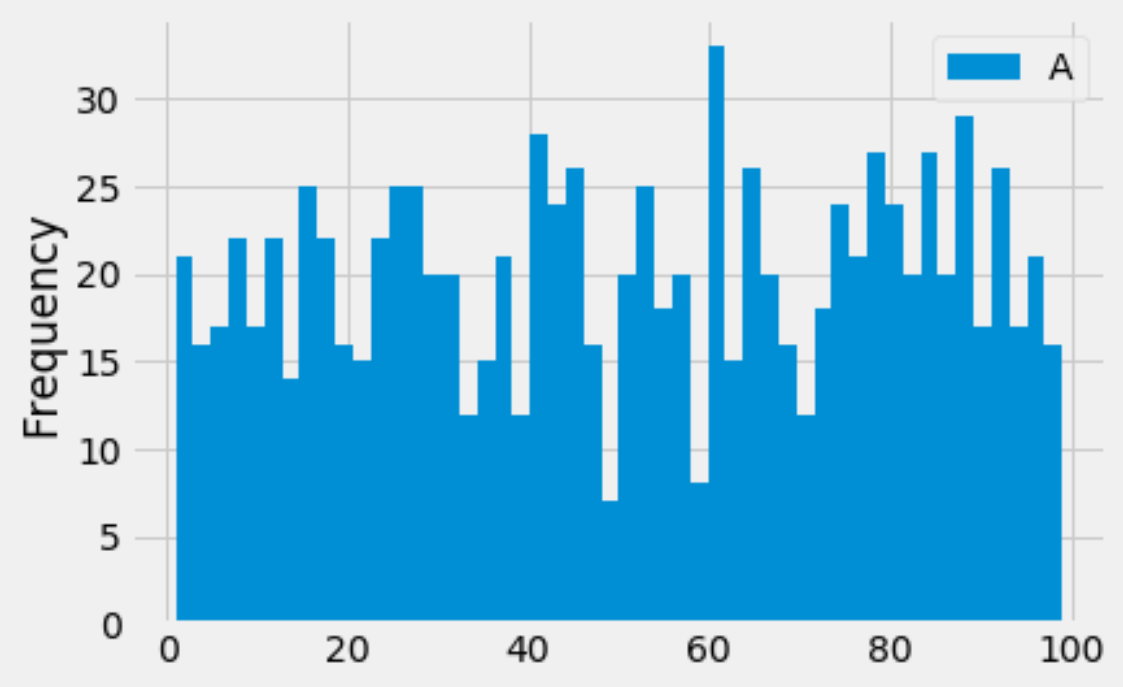

```
In [153]: # hist novamente
          df = pd.DataFrame(np.random.randint(1, 100, 1000), columns=['A'])
          df.head(n=10)
```

Out[153]:

| | A |
|---|---|
| 0 | 1 |
| 1 | 76 |
| 2 | 25 |
| 3 | 33 |
| 4 | 98 |
| 5 | 27 |
| 6 | 27 |
| 7 | 3 |
| 8 | 62 |
| 9 | 90 |

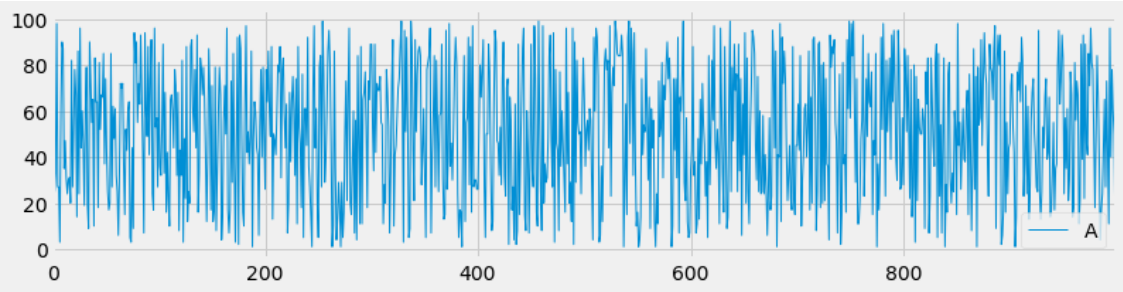
```
In [154]: df.plot.hist(bins=50) #
```

```
Out[154]: <matplotlib.axes._subplots.AxesSubplot at 0x7fddf64e0dd8>
```



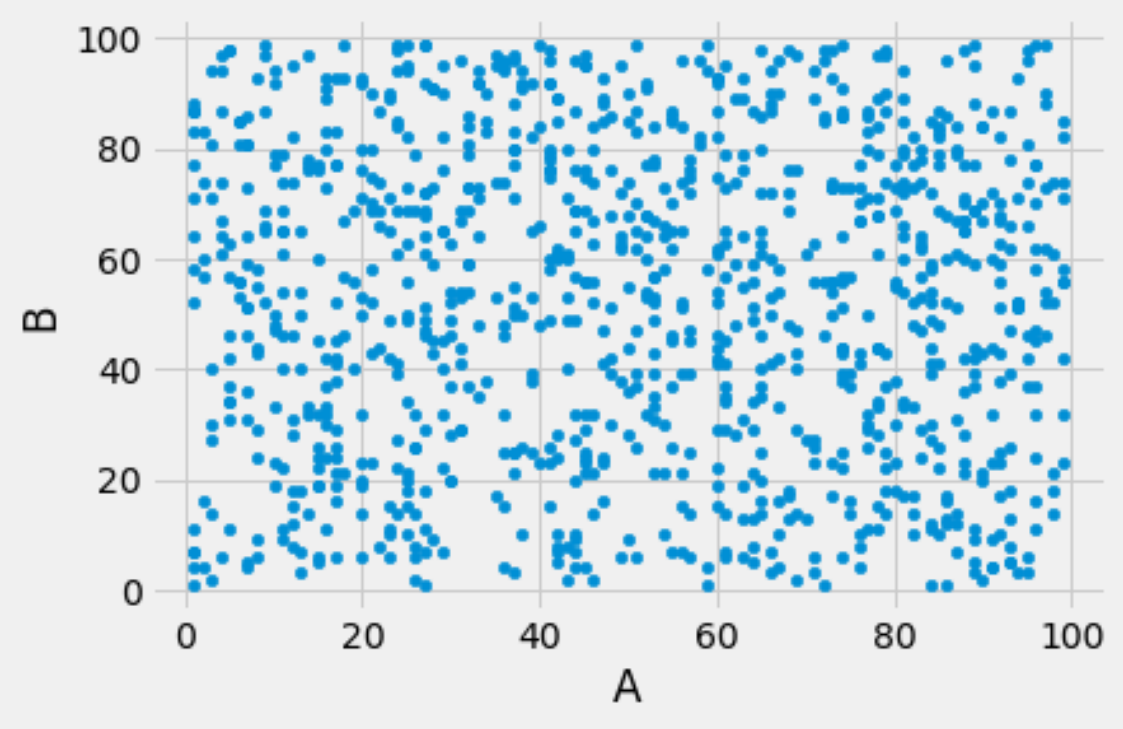

In [155]:
```
# df2 = df.loc[:100, 'A']
df.plot.line(x=df.index, y='A', figsize=(12, 3), lw=1)
```

Out[155]: <matplotlib.axes._subplots.AxesSubplot at 0x7fddf64adf60>



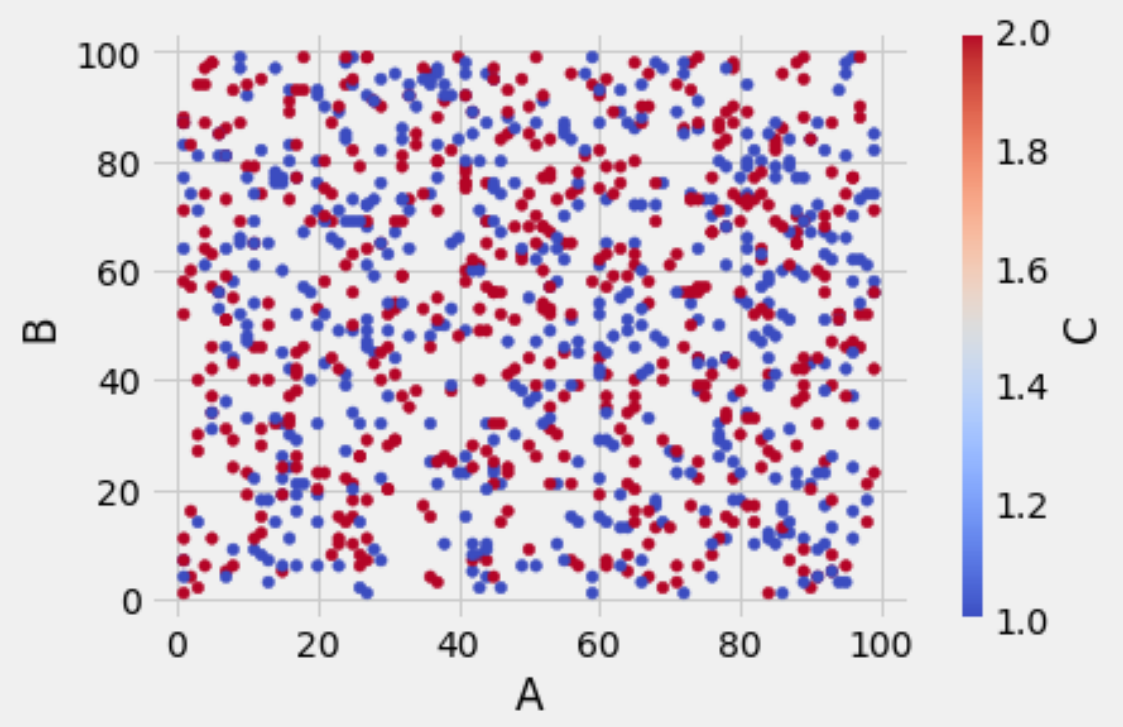

In [156]:
```
df['B'] = np.random.randint(1, 100, 1000)
df.plot.scatter(x='A', y='B')
```

Out[156]: <matplotlib.axes._subplots.AxesSubplot at 0x7fddf63be6a0>




```
In [157]: df['C'] = np.random.randint(1, 3, 1000)
          df.plot.scatter(x='A', y='B', c='C', cmap='coolwarm')
```

```
Out[157]: <matplotlib.axes._subplots.AxesSubplot at 0x7fddf63f2cc0>
```



```
In [158]: df.plot.box()
```

```
Out[158]: <matplotlib.axes._subplots.AxesSubplot at 0x7fddf4b3c390>
```