Chapter1_async

March 14, 2022

1 Chapter1

```
[1]: import pandas as pd
```

```
[2]: scores = {"name":['Ray','Japhy','Zosa'],
               "city":['San Francisco','San Francisco','Denver'],
               "score":[75,92,94]
              }
```

```
[3]: df = pd.DataFrame(scores)
```

```
[4]: df
```

```
[4]:     name           city  score
     0    Ray  San Francisco     75
     1  Japhy  San Francisco     92
     2   Zosa         Denver     94
```

```
[5]: df['score']
```

```
[5]: 0    75
     1    92
     2    94
     Name: score, dtype: int64
```

```
[6]: df['name_city'] = df['name'] + '_' + df['city']
```

```
[7]: df[df['score']>90]
```

```
[7]:     name           city  score            name_city
     1  Japhy  San Francisco     92  Japhy_San Francisco
     2   Zosa         Denver     94          Zosa_Denver
```

```
[8]: iris = pd.read_csv('./iris.csv')
```

```
[9]: iris.shape
```

```
[9]: (150, 5)
```

```
[10]: iris.head(3)
```

```
[10]:    sepal_length  sepal_width  petal_length  petal_width species
      0           5.1          3.5           1.4          0.2  setosa
      1           4.9          3.0           1.4          0.2  setosa
      2           4.7          3.2           1.3          0.2  setosa
```

```
[11]: iris.tail(3)
```

```
[11]:      sepal_length  sepal_width  petal_length  petal_width    species
      147           6.5          3.0           5.2          2.0  virginica
      148           6.2          3.4           5.4          2.3  virginica
      149           5.9          3.0           5.1          1.8  virginica
```

```
[12]: iris.dtypes
```

```
[12]: sepal_length    float64
      sepal_width     float64
      petal_length    float64
      petal_width     float64
      species          object
      dtype: object
```

```
[13]: iris.loc[3:5]
```

```
[13]:    sepal_length  sepal_width  petal_length  petal_width species
      3           4.6          3.1           1.5          0.2  setosa
      4           5.0          3.6           1.4          0.2  setosa
      5           5.4          3.9           1.7          0.4  setosa
```

```
[14]: iris.loc[3,'sepal_length']
```

```
[14]: 4.6
```

```
[15]: iris.iloc[3,0]
```

```
[15]: 4.6
```

```
[16]: # iris.to_csv('iris-output.csv',index=False)
```

```
[17]: emissions = pd.DataFrame({"country":['China','United States','India'],
                "year":['2018','2018','2018'],
                "co2_emissions":[10060000000.0,5410000000.0,2650000000.0]})
```

```
[18]: emissions
```

```
[18]:          country  year  co2_emissions
      0          China  2018   1.006000e+10
```

```
1  United States  2018   5.410000e+09
2          India  2018   2.650000e+09
```

```
[19]: # pd.set_option('display.max_rows', 2)
      pd.reset_option('^display.', silent=True)

      emissions
```

```
[19]:          country  year  co2_emissions
      0          China  2018   1.006000e+10
      1  United States  2018   5.410000e+09
      2          India  2018   2.650000e+09
```

```
[20]: # pd.set_option('display.max_columns', 2)
      pd.reset_option('^display.', silent=True)
      emissions
```

```
[20]:          country  year  co2_emissions
      0          China  2018   1.006000e+10
      1  United States  2018   5.410000e+09
      2          India  2018   2.650000e+09
```

```
[21]: pd.options.display.float_format = '{:,.2f}'.format
```

```
[22]: emissions
```

```
[22]:          country  year     co2_emissions
      0          China  2018 10,060,000,000.00
      1  United States  2018  5,410,000,000.00
      2          India  2018  2,650,000,000.00
```

2 Chapter2

```
[23]: planets = pd.read_csv('planets.csv')
```

```
[24]: planets.head(3)
```

```
[24]:             method  number  orbital_period  mass  distance  year
      0  Radial Velocity       1          269.30  7.10     77.40  2006
      1  Radial Velocity       1          874.77  2.21     56.95  2008
      2  Radial Velocity       1          763.00  2.60     19.84  2011
```

```
[25]: planets.dtypes
```

```
[25]: method             object
      number              int64
      orbital_period    float64
```

```
mass                float64
distance            float64
year                  int64
dtype: object
```

```
[26]: planets.mean()
```

```
C:\Users\aadar\AppData\Local\Temp\ipykernel_11968\656747818.py:1: FutureWarning:
Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None')
is deprecated; in a future version this will raise TypeError.  Select only valid
columns before calling the reduction.
  planets.mean()
```

```
[26]: number                1.79
      orbital_period    2,002.92
      mass                  2.64
      distance            264.07
      year              2,009.07
      dtype: float64
```

```
[27]: planets['number'][0]/planets['mass'][0]
```

```
[27]: 0.14084507042253522
```

```
[28]: planets['number'][0].astype(float)
```

```
[28]: 1.0
```

```
[29]: planets['mass'][0].astype(int)
      # note not a rounded value
```

```
[29]: 7
```

```
[30]: planets['year'][0].astype(str)
```

```
[30]: '2006'
```

```
[31]: planets['year_dt'] = pd.to_datetime(planets['year'], format='%Y')
      planets['year_dt']
```

```
[31]: 0       2006-01-01
      1       2008-01-01
      2       2011-01-01
      3       2007-01-01
      4       2009-01-01
                 …
      1030    2006-01-01
      1031    2007-01-01
```

```
1032   2007-01-01
1033   2008-01-01
1034   2008-01-01
Name: year_dt, Length: 1035, dtype: datetime64[ns]
```

[32]:
```python
names = pd.Series(['Pomeray, CODY ',' Wagner; Jarry','smith, Ray'])
```

[33]:
```python
names = names.str.replace(';',',')
names
```

[33]:
```
0    Pomeray, CODY 
1     Wagner, Jarry
2        smith, Ray
dtype: object
```

[34]:
```python
names.str.len()
```

[34]:
```
0    14
1    14
2    10
dtype: int64
```

[35]:
```python
names = names.str.strip()
names.str.len()
```

[35]:
```
0    13
1    13
2    10
dtype: int64
```

[36]:
```python
names = names.str.lower()
names
```

[36]:
```
0    pomeray, cody
1    wagner, jarry
2       smith, ray
dtype: object
```

[37]:
```python
names = names.str.split(', ')
names
```

[37]:
```
0    [pomeray, cody]
1    [wagner, jarry]
2       [smith, ray]
dtype: object
```

[38]:
```python
names = pd.Series([i[::-1] for i in names])
names
```

```
[38]: 0    [cody, pomeray]
      1    [jarry, wagner]
      2       [ray, smith]
      dtype: object
```

```
[39]: names = [' '.join(i) for i in names]
      names
```

```
[39]: ['cody pomeray', 'jarry wagner', 'ray smith']
```

3 Chapter3

```
[40]: iris = pd.read_csv('./iris.csv')
      iris.head(5)
```

```
[40]:    sepal_length  sepal_width  petal_length  petal_width species
      0          5.10         3.50          1.40         0.20  setosa
      1          4.90         3.00          1.40         0.20  setosa
      2          4.70         3.20          1.30         0.20  setosa
      3          4.60         3.10          1.50         0.20  setosa
      4          5.00         3.60          1.40         0.20  setosa
```

```
[41]: # can flatten hierarchical index with reset_index()
      iris.groupby(['species']).max()
```

```
[41]:             sepal_length  sepal_width  petal_length  petal_width
      species
      setosa              5.80         4.40          1.90         0.60
      versicolor          7.00         3.40          5.10         1.80
      virginica           7.90         3.80          6.90         2.50
```

```
[42]: df = iris.groupby(['species']).agg({'sepal_length':
       ↪['mean','min','max'],'sepal_width':'count'})
      df
```

```
[42]:           sepal_length           sepal_width
                        mean  min  max       count
      species
      setosa            5.01 4.30 5.80          50
      versicolor        5.94 4.90 7.00          50
      virginica         6.59 4.90 7.90          50
```

```
[43]: df['sepal_length']
```

```
[43]:             mean  min  max
      species
      setosa      5.01 4.30 5.80
```

```
versicolor  5.94 4.90 7.00
virginica   6.59 4.90 7.90
```

[44]:
```
df.columns = ['_'.join(col).strip() for col in df.columns.values]
df.reset_index()
df
```

[44]:
```
            sepal_length_mean  sepal_length_min  sepal_length_max  \
species
setosa                   5.01              4.30              5.80
versicolor               5.94              4.90              7.00
virginica                6.59              4.90              7.90

            sepal_width_count
species
setosa                     50
versicolor                 50
virginica                  50
```

[45]:
```
groupings = iris.groupby(['species'])
```

[46]:
```
groupings.get_group('setosa').head()
```

[46]:
```
   sepal_length  sepal_width  petal_length  petal_width species
0          5.10         3.50          1.40         0.20  setosa
1          4.90         3.00          1.40         0.20  setosa
2          4.70         3.20          1.30         0.20  setosa
3          4.60         3.10          1.50         0.20  setosa
4          5.00         3.60          1.40         0.20  setosa
```

[47]:
```
groupings.max()
```

[47]:
```
            sepal_length  sepal_width  petal_length  petal_width
species
setosa              5.80         4.40          1.90         0.60
versicolor          7.00         3.40          5.10         1.80
virginica           7.90         3.80          6.90         2.50
```

[48]:
```
groupings.apply(lambda x: x.max())
```

[48]:
```
            sepal_length  sepal_width  petal_length  petal_width     species
species
setosa              5.80         4.40          1.90         0.60      setosa
versicolor          7.00         3.40          5.10         1.80  versicolor
virginica           7.90         3.80          6.90         2.50   virginica
```

[49]:
```
groupings.filter(lambda x: x['petal_length'].max() <5)
```

```
[49]:     sepal_length  sepal_width  petal_length  petal_width species
      0           5.10         3.50          1.40         0.20  setosa
      1           4.90         3.00          1.40         0.20  setosa
      2           4.70         3.20          1.30         0.20  setosa
      3           4.60         3.10          1.50         0.20  setosa
      4           5.00         3.60          1.40         0.20  setosa
      5           5.40         3.90          1.70         0.40  setosa
      6           4.60         3.40          1.40         0.30  setosa
      7           5.00         3.40          1.50         0.20  setosa
      8           4.40         2.90          1.40         0.20  setosa
      9           4.90         3.10          1.50         0.10  setosa
      10          5.40         3.70          1.50         0.20  setosa
      11          4.80         3.40          1.60         0.20  setosa
      12          4.80         3.00          1.40         0.10  setosa
      13          4.30         3.00          1.10         0.10  setosa
      14          5.80         4.00          1.20         0.20  setosa
      15          5.70         4.40          1.50         0.40  setosa
      16          5.40         3.90          1.30         0.40  setosa
      17          5.10         3.50          1.40         0.30  setosa
      18          5.70         3.80          1.70         0.30  setosa
      19          5.10         3.80          1.50         0.30  setosa
      20          5.40         3.40          1.70         0.20  setosa
      21          5.10         3.70          1.50         0.40  setosa
      22          4.60         3.60          1.00         0.20  setosa
      23          5.10         3.30          1.70         0.50  setosa
      24          4.80         3.40          1.90         0.20  setosa
      25          5.00         3.00          1.60         0.20  setosa
      26          5.00         3.40          1.60         0.40  setosa
      27          5.20         3.50          1.50         0.20  setosa
      28          5.20         3.40          1.40         0.20  setosa
      29          4.70         3.20          1.60         0.20  setosa
      30          4.80         3.10          1.60         0.20  setosa
      31          5.40         3.40          1.50         0.40  setosa
      32          5.20         4.10          1.50         0.10  setosa
      33          5.50         4.20          1.40         0.20  setosa
      34          4.90         3.10          1.50         0.20  setosa
      35          5.00         3.20          1.20         0.20  setosa
      36          5.50         3.50          1.30         0.20  setosa
      37          4.90         3.60          1.40         0.10  setosa
      38          4.40         3.00          1.30         0.20  setosa
      39          5.10         3.40          1.50         0.20  setosa
      40          5.00         3.50          1.30         0.30  setosa
      41          4.50         2.30          1.30         0.30  setosa
      42          4.40         3.20          1.30         0.20  setosa
      43          5.00         3.50          1.60         0.60  setosa
      44          5.10         3.80          1.90         0.40  setosa
      45          4.80         3.00          1.40         0.30  setosa
```

```
46            5.10          3.80           1.60          0.20  setosa
47            4.60          3.20           1.40          0.20  setosa
48            5.30          3.70           1.50          0.20  setosa
49            5.00          3.30           1.40          0.20  setosa
```

[50]:

```
df = pd.DataFrame({"Region":
 ↪['North','West','East','South','North','West','East','South'],
          "Team":['One','One','One','One','Two','Two','Two','Two'],
          "Revenue":[7500,5500,2750,6400,2300,3750,1900,575],
            "Cost":[5200,5100,4400,5300,1250,1300,2100,50]})
df
```

[50]:

```
  Region Team  Revenue  Cost
0  North  One     7500  5200
1   West  One     5500  5100
2   East  One     2750  4400
3  South  One     6400  5300
4  North  Two     2300  1250
5   West  Two     3750  1300
6   East  Two     1900  2100
7  South  Two      575    50
```

[51]:

```
df.pivot(index='Region',columns='Team',values='Revenue')
```

[51]:

```
Team     One   Two
Region
East    2750  1900
North   7500  2300
South   6400   575
West    5500  3750
```

[52]:

```
df2 = df.set_index(['Region','Team'])
```

[53]:

```
stacked = pd.DataFrame(df2.stack())
stacked
```

[53]:

```
                         0
Region Team
North  One  Revenue  7500
            Cost     5200
West   One  Revenue  5500
            Cost     5100
East   One  Revenue  2750
            Cost     4400
South  One  Revenue  6400
            Cost     5300
North  Two  Revenue  2300
```

```
                Cost      1250
West    Two   Revenue   3750
                Cost      1300
East    Two   Revenue   1900
                Cost      2100
South   Two   Revenue    575
                Cost        50
```

[54]:

```
stacked.unstack('Region')
```

[54]:

```
                     0
Region          East  North  South   West
Team
One   Revenue   2750   7500   6400   5500
      Cost      4400   5200   5300   5100
Two   Revenue   1900   2300    575   3750
      Cost      2100   1250     50   1300
```

[55]:

```
df.head(3)
```

[55]:

```
  Region Team  Revenue  Cost
0  North  One     7500  5200
1   West  One     5500  5100
2   East  One     2750  4400
```

[56]:

```
df.melt(id_vars=['Region','Team'], var_name='value type')
```

[56]:

```
   Region Team value type  value
0   North  One    Revenue   7500
1    West  One    Revenue   5500
2    East  One    Revenue   2750
3   South  One    Revenue   6400
4   North  Two    Revenue   2300
5    West  Two    Revenue   3750
6    East  Two    Revenue   1900
7   South  Two    Revenue    575
8   North  One       Cost   5200
9    West  One       Cost   5100
10   East  One       Cost   4400
11  South  One       Cost   5300
12  North  Two       Cost   1250
13   West  Two       Cost   1300
14   East  Two       Cost   2100
15  South  Two       Cost     50
```

[57]:

```
# mean by default
df.pivot_table(index='Team',columns='Region',values='Revenue')
```

```
[57]: Region  East  North  South  West
      Team
      One     2750   7500   6400  5500
      Two     1900   2300    575  3750
```

```
[58]: df1 = pd.DataFrame({'letter': ['A', 'B', 'C', 'D'],
                          'number': [1, 2, 3, 4]})
      df2 = pd.DataFrame({'letter': ['C', 'D', 'E', 'F'],
                          'number': [3, 4, 5, 6]})
```

```
[59]: df1.merge(df2,how='left',on='number')
```

```
[59]:   letter_x  number letter_y
      0        A       1      NaN
      1        B       2      NaN
      2        C       3        C
      3        D       4        D
```

```
[60]: df1.merge(df2,how='inner',left_on='number',right_on='number')
```

```
[60]:   letter_x  number letter_y
      0        C       3        C
      1        D       4        D
```

```
[61]: df1.merge(df2,how='right',on='number',suffixes=('','_right'))
```

```
[61]:   letter  number letter_right
      0      C       3            C
      1      D       4            D
      2    NaN       5            E
      3    NaN       6            F
```

```
[62]: # drop duplicates with .drop_duplicates()
      df3 = pd.concat([df1,df2]).drop_duplicates().reset_index(drop=True)
      df3
```

```
[62]:   letter  number
      0      A       1
      1      B       2
      2      C       3
      3      D       4
      4      E       5
      5      F       6
```

```
[63]: df4 = pd.concat([df1,df2],axis=1)
      df4
```

```
[63]:   letter  number letter  number
      0      A       1      C       3
      1      B       2      D       4
      2      C       3      E       5
      3      D       4      F       6
```

```
[64]: new_row = pd.Series(['Z',26],index=df3.columns)
      df3.append(new_row,ignore_index=True)
```

```
C:\Users\aadar\AppData\Local\Temp\ipykernel_11968\696788980.py:2: FutureWarning:
The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.
  df3.append(new_row,ignore_index=True)
```

```
[64]:   letter  number
      0      A       1
      1      B       2
      2      C       3
      3      D       4
      4      E       5
      5      F       6
      6      Z      26
```

```
[65]: join_df = pd.DataFrame({'letter': ['F','G', 'H', 'I'],
                              'number': [6, 7, 8, 9]})
```

```
[66]: df2.join(join_df, rsuffix='_right')
```

```
[66]:   letter  number letter_right  number_right
      0      C       3            F             6
      1      D       4            G             7
      2      E       5            H             8
      3      F       6            I             9
```

```
[67]: df = pd.DataFrame({"Species":['Chinook','Chum','Coho','Steelhead','Bull Trout'],
                   "Population":['Skokomish','Lower␣
       ↪Skokomish','Skokomish','Skokomish','SF Skokomish'],
                   "Count":[1208,2396,3220,6245,8216]})
      df
```

```
[67]:      Species       Population  Count
      0     Chinook        Skokomish   1208
      1        Chum  Lower Skokomish   2396
      2        Coho        Skokomish   3220
      3   Steelhead        Skokomish   6245
      4  Bull Trout     SF Skokomish   8216
```

[68]:

```
import numpy as np
bins = [0, 2000, 4000, 6000, 8000, np.inf]
labels = ['Low Return', 'Below Avg Return', 'Avg Return', 'Above Avg Return',␣
 ↪'High Return']
```

[69]:

```
df['Count Category'] = pd.cut(df['Count'], bins, labels=labels)
df
```

[69]:

```
      Species       Population  Count     Count Category
0     Chinook         Skokomish   1208         Low Return
1        Chum   Lower Skokomish   2396   Below Avg Return
2        Coho         Skokomish   3220   Below Avg Return
3   Steelhead         Skokomish   6245   Above Avg Return
4  Bull Trout      SF Skokomish   8216        High Return
```

[70]:

```
fed_status ={"Chinook":"Threatened",
"Chum":"Not Warranted",
"Coho":"Not Warranted",
"Steelhead":"Threatened"}
```

[71]:

```
df['Federal Status'] = df['Species'].map(fed_status)
df
```

[71]:

```
      Species       Population  Count     Count Category Federal Status
0     Chinook         Skokomish   1208         Low Return     Threatened
1        Chum   Lower Skokomish   2396   Below Avg Return  Not Warranted
2        Coho         Skokomish   3220   Below Avg Return  Not Warranted
3   Steelhead         Skokomish   6245   Above Avg Return     Threatened
4  Bull Trout      SF Skokomish   8216        High Return            NaN
```

[72]:

```
df['Count Category'] = pd.Categorical(df['Count Category'],
               ordered=True,
               categories=labels)
df['Count Category']
```

[72]:

```
0          Low Return
1    Below Avg Return
2    Below Avg Return
3    Above Avg Return
4         High Return
Name: Count Category, dtype: category
Categories (5, object): ['Low Return' < 'Below Avg Return' < 'Avg Return' <
'Above Avg Return' < 'High Return']
```

[73]:

```
df.sort_values(by=['Count Category'],ascending=False)
```

[73]:

```
      Species       Population  Count     Count Category Federal Status
4  Bull Trout      SF Skokomish   8216        High Return            NaN
```

```
3   Steelhead         Skokomish   6245  Above Avg Return     Threatened
1        Chum   Lower Skokomish   2396  Below Avg Return  Not Warranted
2        Coho         Skokomish   3220  Below Avg Return  Not Warranted
0     Chinook         Skokomish   1208        Low Return     Threatened
```

[74]:

```
pd.get_dummies(df['Count Category'])
```

[74]:

```
   Low Return  Below Avg Return  Avg Return  Above Avg Return  High Return
0           1                 0           0                 0            0
1           0                 1           0                 0            0
2           0                 1           0                 0            0
3           0                 0           0                 1            0
4           0                 0           0                 0            1
```

4 Chapter4

[75]:

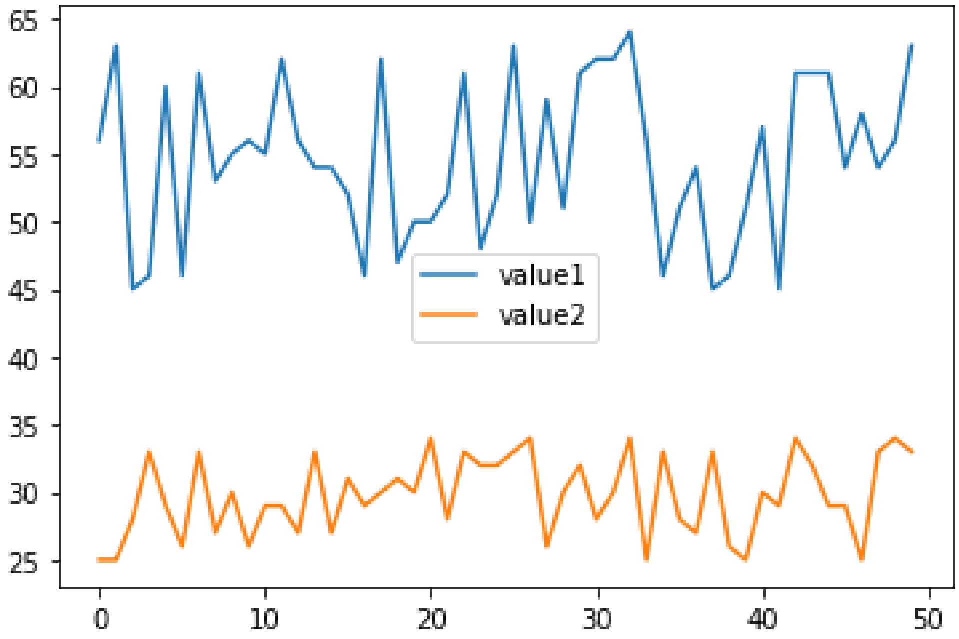
```
import numpy as np
daterange = pd.period_range('1/1/1950', freq='1d', periods=50)
date_df = pd.DataFrame(data=daterange,columns=['day'])
date_df['value1'] = np.random.randint(45,65,size=(len(date_df)))
date_df['value2'] = np.random.randint(25,35,size=(len(date_df)))
date_df.head(3)
```

[75]:

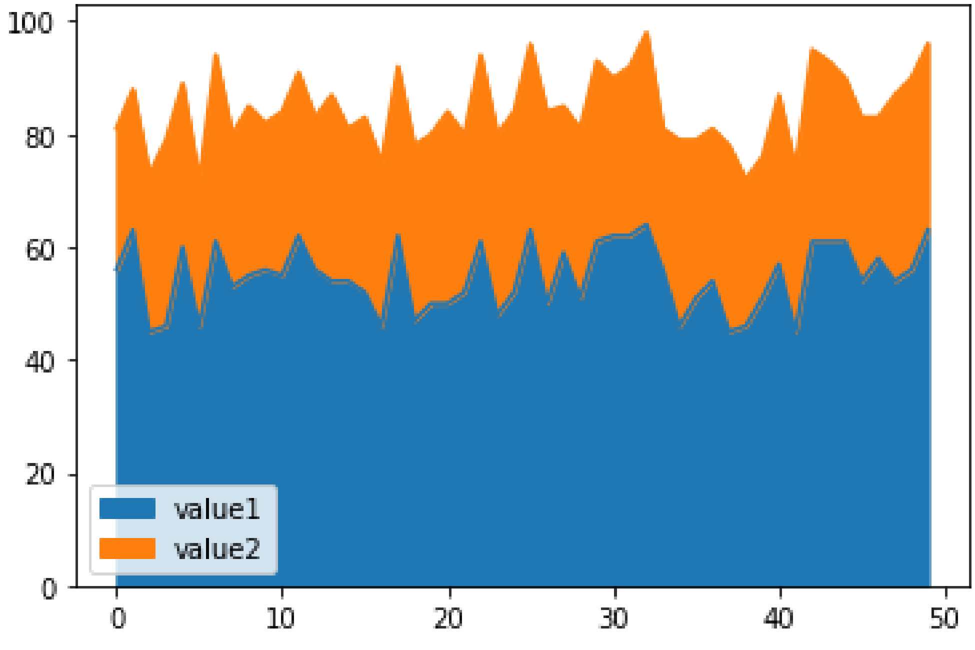
```
          day  value1  value2
0  1950-01-01      56      25
1  1950-01-02      63      25
2  1950-01-03      45      28
```

[76]:

```
ax = date_df.plot();
```



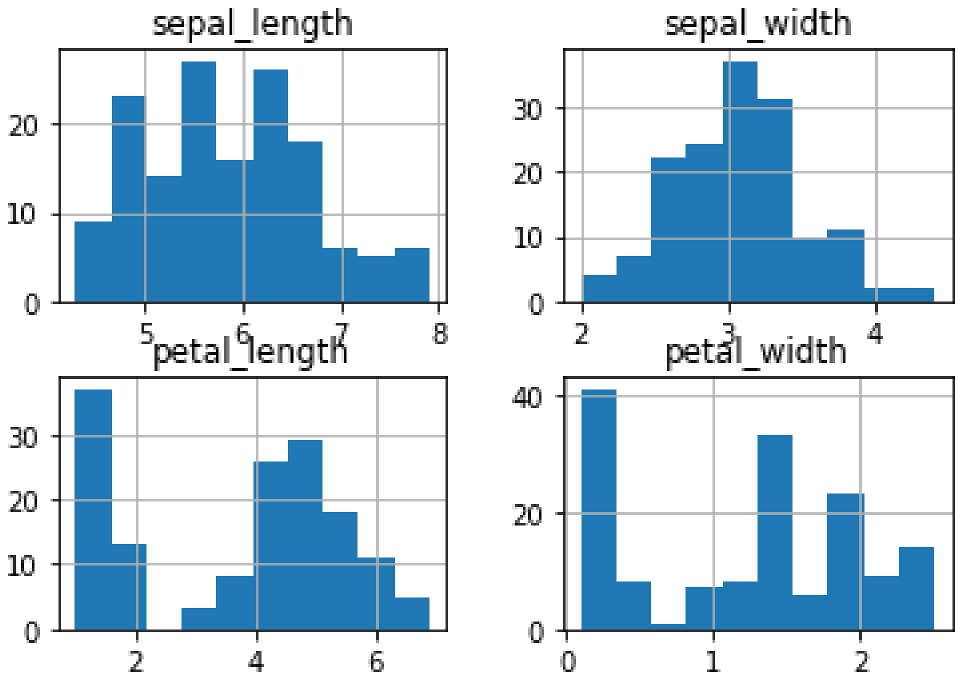
[77]:
```
date_df.plot.area(stacked=True);
```



[78]:
```
iris = pd.read_csv('./iris.csv')
```

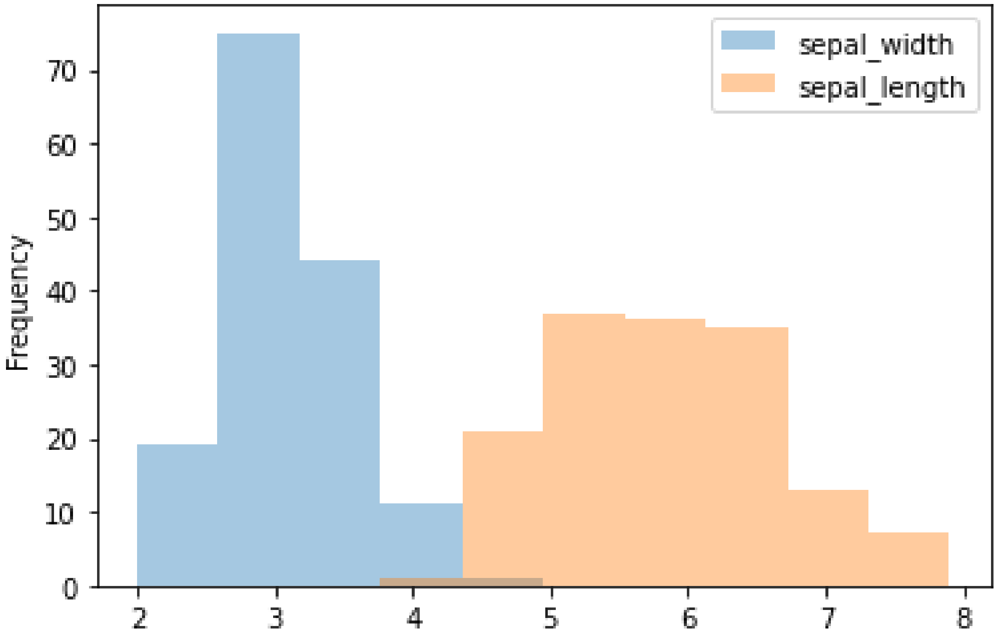
[79]:

```
iris.hist();
```



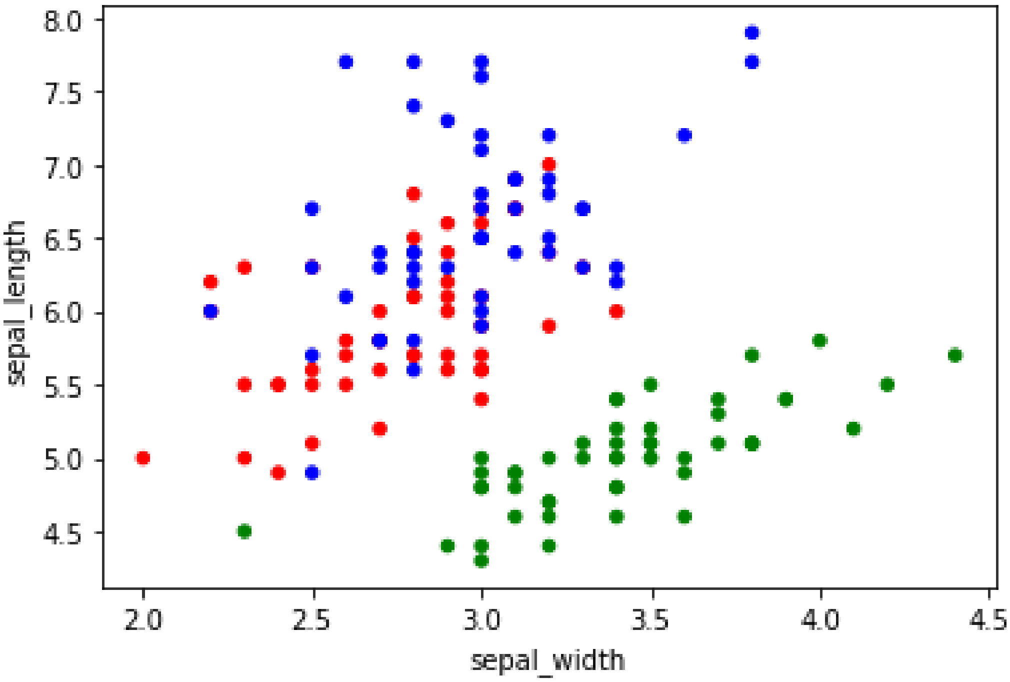
[80]:

```
iris[['sepal_width','sepal_length']].plot.hist(alpha=0.4);
```



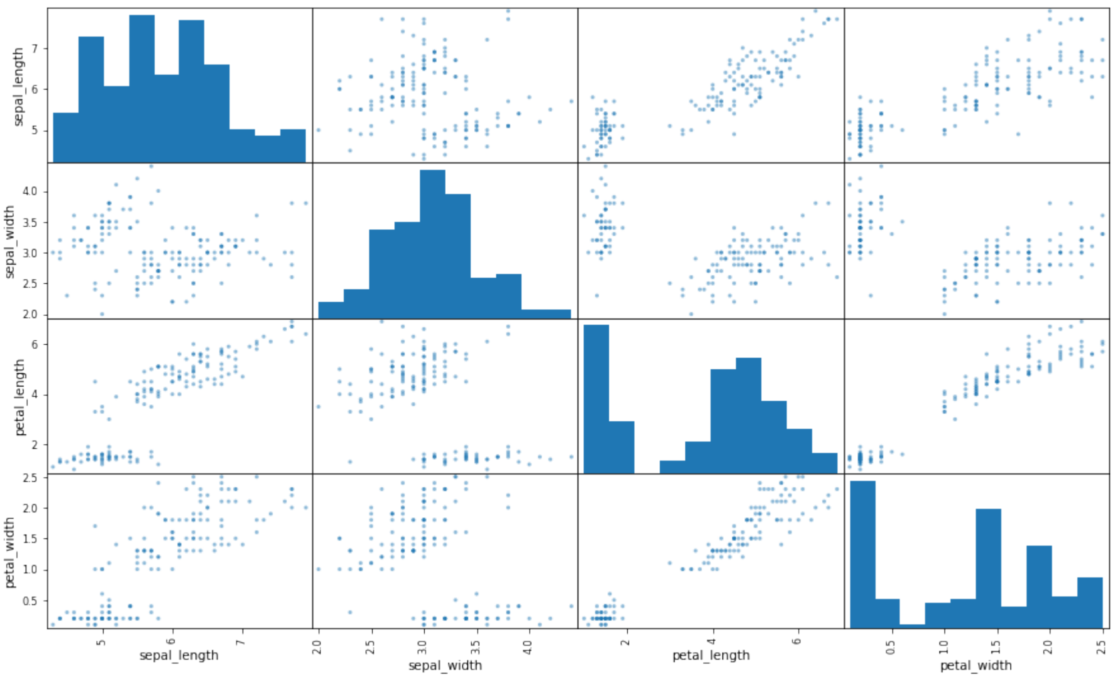
[81]:

```
colors = {"versicolor":"red","setosa":"green","virginica":"blue"}
iris['colors'] = iris['species'].map(colors)
iris.plot.scatter(x='sepal_width', y='sepal_length', color=iris['colors']);
```



[82]:

```
from pandas.plotting import scatter_matrix
scatter_matrix(iris,figsize=(15, 9),);
```



[83]:
```
iris.mean()
```

```
C:\Users\aadar\AppData\Local\Temp\ipykernel_11968\935066809.py:1: FutureWarning:
Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None')
is deprecated; in a future version this will raise TypeError.  Select only valid
columns before calling the reduction.
  iris.mean()
```

[83]:
```
sepal_length    5.84
sepal_width     3.06
petal_length    3.76
petal_width     1.20
dtype: float64
```

[84]:
```
iris.median()
```

```
C:\Users\aadar\AppData\Local\Temp\ipykernel_11968\1297003277.py:1:
FutureWarning: Dropping of nuisance columns in DataFrame reductions (with
'numeric_only=None') is deprecated; in a future version this will raise
TypeError.  Select only valid columns before calling the reduction.
  iris.median()
```

[84]:
```
sepal_length    5.80
sepal_width     3.00
petal_length    4.35
petal_width     1.30
```

```
dtype: float64
```

```
[85]: iris.mode()
```

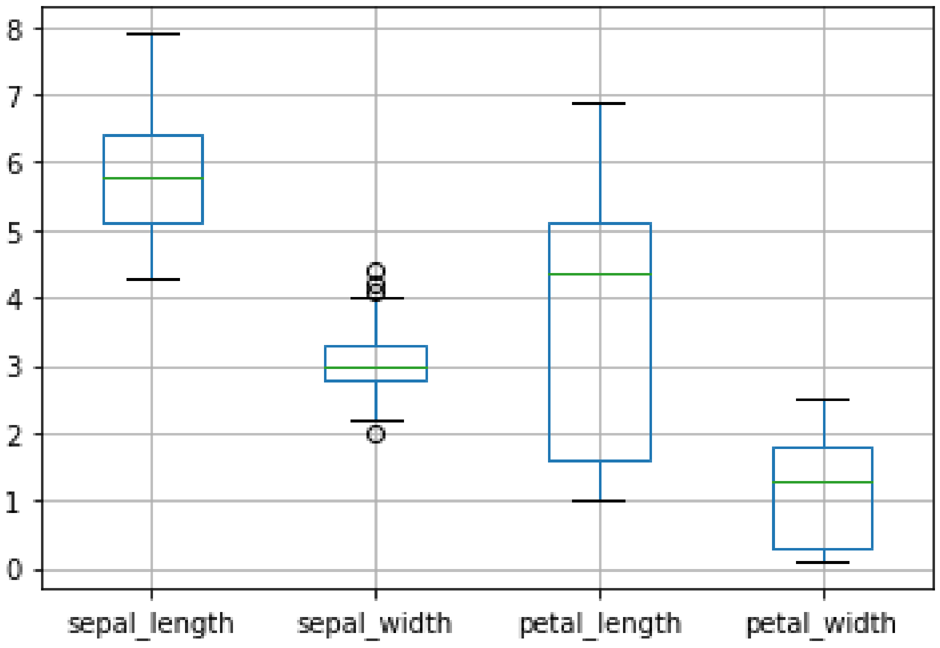
```
[85]:    sepal_length  sepal_width  petal_length  petal_width     species colors
      0          5.00         3.00          1.40         0.20      setosa   blue
      1           NaN          NaN          1.50          NaN  versicolor  green
      2           NaN          NaN           NaN          NaN   virginica    red
```

```
[86]: iris.std()
```

```
C:\Users\aadar\AppData\Local\Temp\ipykernel_11968\3849825860.py:1:
FutureWarning: Dropping of nuisance columns in DataFrame reductions (with
'numeric_only=None') is deprecated; in a future version this will raise
TypeError.  Select only valid columns before calling the reduction.
  iris.std()
```

```
[86]: sepal_length   0.83
      sepal_width    0.44
      petal_length   1.77
      petal_width    0.76
      dtype: float64
```

```
[87]: iris.boxplot();
```



```
[88]: iris.describe()
```

```
[88]:        sepal_length  sepal_width  petal_length  petal_width
      count        150.00       150.00        150.00       150.00
      mean           5.84         3.06          3.76         1.20
      std            0.83         0.44          1.77         0.76
      min            4.30         2.00          1.00         0.10
      25%            5.10         2.80          1.60         0.30
      50%            5.80         3.00          4.35         1.30
      75%            6.40         3.30          5.10         1.80
      max            7.90         4.40          6.90         2.50
```

```
[89]: iris.corr()
```

```
[89]:               sepal_length  sepal_width  petal_length  petal_width
      sepal_length          1.00        -0.12          0.87         0.82
      sepal_width          -0.12         1.00         -0.43        -0.37
      petal_length          0.87        -0.43          1.00         0.96
      petal_width           0.82        -0.37          0.96         1.00
```

```
[90]: iris.corr().style.background_gradient(cmap='RdYlGn', axis=None)
```

```
[90]: <pandas.io.formats.style.Styler at 0x2a02f402880>
```

5 Chapter 5

```
[91]: # !pip install pandas_profiling
      from pandas_profiling import ProfileReport
```

```
[92]: iris = pd.read_csv('./iris.csv')
```

```
[93]: profile = ProfileReport(iris,title="Iris Data Profile")
```

```
[94]: profile.to_notebook_iframe()

      #profile.to_widgets() if using Jupyter
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]

Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]

Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]

<IPython.core.display.HTML object>
```

```
[95]: # profile.to_file("iris-profile.html")
```

```
[96]: import pandas as pd
      import geopandas
```

```
PROJ: proj_create_from_database: SQLite error on SELECT name, type,
coordinate_system_auth_name, coordinate_system_code, datum_auth_name,
datum_code, area_of_use_auth_name, area_of_use_code, text_definition, deprecated
FROM geodetic_crs WHERE auth_name = ? AND code = ?: no such column:
area_of_use_auth_name
```

[97]:

```
peaks = pd.DataFrame(
    {'Peak Name': ['Green Mtn.', 'So. Boulder Peak', 'Bear Peak', 'Flagstaff␣
 ↪Mtn.', 'Mt. Sanitas'],
     'Latitude': [39.9821, 39.9539, 39.9603, 40.0017, 40.0360968],
     'Longitude': [-105.3016, -105.2992, -105.2952, -105.3075, -105.3061024]})
```

[98]:

```
gdf = geopandas.GeoDataFrame(
    peaks, geometry=geopandas.points_from_xy(peaks.Longitude, peaks.Latitude))
gdf
```

[98]:

```
          Peak Name  Latitude  Longitude                    geometry
0        Green Mtn.     39.98    -105.30  POINT (-105.30160 39.98210)
1  So. Boulder Peak     39.95    -105.30  POINT (-105.29920 39.95390)
2         Bear Peak     39.96    -105.30  POINT (-105.29520 39.96030)
3    Flagstaff Mtn.     40.00    -105.31  POINT (-105.30750 40.00170)
4       Mt. Sanitas     40.04    -105.31  POINT (-105.30610 40.03610)
```

[99]:

```
token ='your token'
```

[100]:

```
import plotly.express as px
px.set_mapbox_access_token(token)
gdf['size'] = 65

fig = px.scatter_mapbox(gdf,
                        lat=gdf.geometry.y,
                        lon=gdf.geometry.x,
                        color="Peak Name",
                        hover_name="Peak Name",
                        mapbox_style='outdoors',
                        size='size',
                        zoom=10)


fig.show()
```

[101]:

```
# !pip install Dask
```

[102]:

```
import dask.dataframe as dd
df = dd.read_csv('iris.csv')
df.head()
```

[102]:

```
   sepal_length  sepal_width  petal_length  petal_width species
0          5.10         3.50          1.40         0.20  setosa
1          4.90         3.00          1.40         0.20  setosa
2          4.70         3.20          1.30         0.20  setosa
```

```
3           4.60         3.10          1.50         0.20  setosa
4           5.00         3.60          1.40         0.20  setosa
```

[103]:

```
# !pip install databricks
```

[104]:

```
# !pip install pyspark
```

[105]:

```
import pandas as pd
import numpy as np
import databricks.koalas as ks
from pyspark.sql import SparkSession
```

```
WARNING:root:Found pyspark version "3.2.1" installed. The pyspark version 3.2
and above has a built-in "pandas APIs on Spark" module ported from Koalas. Try
`import pyspark.pandas as ps` instead.
WARNING:root:'PYARROW_IGNORE_TIMEZONE' environment variable was not set. It is
required to set this environment variable to '1' in both driver and executor
sides if you use pyarrow>=2.0.0. Koalas will set it for you but it does not work
if there is a Spark context already launched.
```

[106]:

```
pdf = pd.DataFrame(np.random.randn(6, 4),  columns=list('ABCD'))
pdf.head()
```

[106]:

```
      A     B     C     D
0  0.28  0.31  0.87 -0.17
1  0.12  1.25 -0.34  1.33
2  0.85  0.25 -0.63 -0.88
3  0.00  1.56  0.09  0.77
4 -0.50 -0.42 -0.06 -1.40
```

[ ]:

```
kdf = ks.from_pandas(pdf)
kdf.head()
```