Python102

Python for Data Science Bootcamp

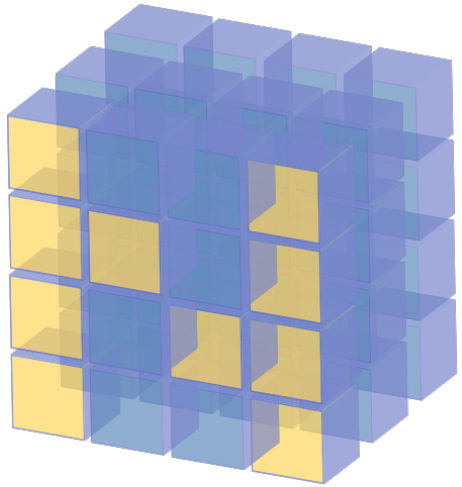
(4.2) Python for Data Analysis

Pandas

AIAT Academy

Python for Data Analysis Outline

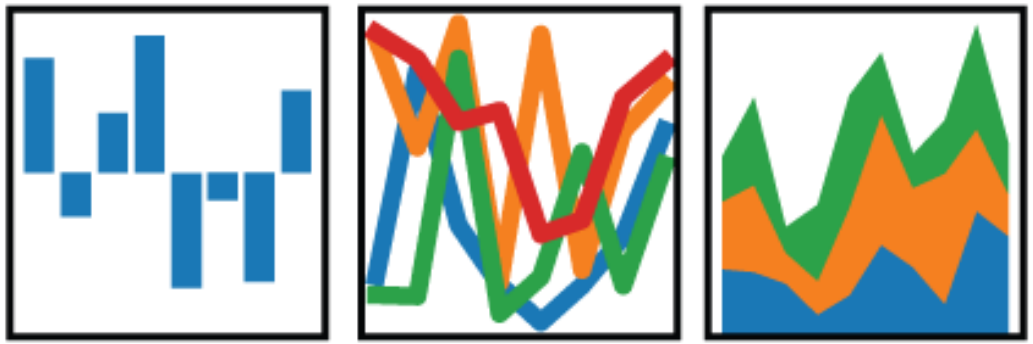
- NumPy
- **Pandas**



NumPy

pandas

$$y_i t = \beta' x_{it} + \mu_i + \epsilon_{it}$$



Pandas

Introduction to Pandas

- Open-source library built on top of NumPy
- Fast analysis and data cleaning and preparation
- High performance and productivity (feature-rich)
- Built-in visualisation features are available
- A wide variety of sources data support

Pandas Installation

- To install Pandas, just going to your terminal or command prompt and typing

```
conda install pandas
```

or

```
pip install pandas
```

Pandas in Python102

AiAT ACADEMY

- Series
- DataFrames
- Missing data
- GroupBy
- Merging, Joining, and Concatenating
- Operations
- Data input/output

AiAT ACADEMY

Series

Pandas Series

```
import numpy as np
import pandas as pd
my_data = [1, 2, 3]
pd.Series(data=my_data)
# 0    1
# 1    2
# 2    3
#dtype: int64
```

```
pandas.Series(data=None, index=None, dtype=None, name=None, copy=False, fastpath=False)
```

Pandas Series

AiAT ACADEMY

```
import numpy as np
import pandas as pd
labels = ['a', 'b', 'c']
my_data = [1, 2, 3]
pd.Series(data=my_data, index=labels)
# a    1
# b    2
# c    3
# dtype: int64
```

```
pandas.Series(data=None, index=None, dtype=None, name=None, copy=False, fastpath=False)
```

Pandas Series

AiAT ACADEMY

```
import numpy as np
import pandas as pd
labels = ['a', 'b', 'c']
my_data = [1, 2, 3]
pd.Series(my_data, labels)
# a    1
# b    2
# c    3
# dtype: int64
```

```
pandas.Series(data=None, index=None, dtype=None, name=None, copy=False, fastpath=False)
```

Pandas Series

AiAT ACADEMY

```
import numpy as np
import pandas as pd
labels = ['a', 'b', 'c']
my_data = [1, 2, 3]
arr = np.array(my_data)
pd.Series(arr, labels)
# a    1
# b    2
# c    3
# dtype: int32
```

```
pandas.Series(data=None, index=None, dtype=None, name=None, copy=False, fastpath=False)
```

Pandas Series

```
import numpy as np
import pandas as pd
d = {'a':1, 'b':2, 'c':3}    # dictionary
pd.Series(d)
# a    1
# b    2
# c    3
# dtype: int64
```

```
pandas.Series(data=None, index=None, dtype=None, name=None, copy=False, fastpath=False)
```

Pandas Series

```
import numpy as np
import pandas as pd


pd.Series(data=[print, 1, [1, 2, 3]])
# 0    <built-in function print>
# 1    1
# 2    [1, 2, 3]
# dtype: object
```

Data parameter can get a list of any variable types

```
pandas.Series(data=None, index=None, dtype=None, name=None, copy=False, fastpath=False)
```

Pandas Series (Indexing)

AiAT ACADEMY

```
import numpy as np
import pandas as pd
serie1 = pd.Series([1, 2, 3], ['Ant', 'Bird', 'Cat'])
# Ant    1
# Bird   2
# Cat    3
# dtype: int64


serie1['Ant']                  # 1
serie1[1]                      # 2
```

```
pandas.Series(data=None, index=None, dtype=None, name=None, copy=False, fastpath=False)
```

Pandas Series (Operations)

AiAT ACADEMY

```
import numpy as np
import pandas as pd
serie1 = pd.Series([1, 2, 3], ['Ant', 'Bird', 'Cat'])
serie2 = pd.Series([1, 2, 3], ['Ant', 'Bird', 'Rat'])
serie1 + series2
# Ant    2
# Bird   4
# Cat    NaN    (Not a Number)
# Rat    NaN    (Not a Number)
# dtype: float64
```

```
pandas.Series(data=None, index=None, dtype=None, name=None, copy=False, fastpath=False)
```

AiAT ACADEMY

DataFrames

Pandas DataFrames

AiAT ACADEMY

```
import numpy as np
import pandas as pd
from numpy.random import randn  # random function
np.random.seed(101)   # set a seed for random function

df = pd.DataFrame(randn(3,3), ['A','B','C'], ['X','Y','Z'])
#           X         Y         Z
# A  -0.993263  0.196800 -1.136645
# B   0.000366  1.025984 -0.156598
# C  -0.031579  0.649826  2.154846
```

```
pandas.DataFrame(data=None, index=None, columns=None, dtype=None, copy=False)
```

Pandas DataFrames (Indexing)

```
df = pd.DataFrame(randn(3,3), ['A','B','C'], ['X','Y','Z'])
#              X         Y         Z
# A    -0.993263  0.196800 -1.136645
# B     0.000366  1.025984 -0.156598
# C    -0.031579  0.649826  2.154846
```

```
df['X']
#              X
# A    -0.993263
# B     0.000366
# C    -0.031579
# Name: X, dtype: float64
```

```
df[['X', 'Y']]
#              X         Y
# A    -0.993263  0.196800
# B     0.000366  1.025984
# C    -0.031579  0.649826
```

```
pandas.DataFrame(data=None, index=None, columns=None, dtype=None, copy=False)
```

Pandas DataFrames (Indexing)

```
df = pd.DataFrame(randn(3,3), ['A','B','C'], ['X','Y','Z'])
#              X         Y         Z
# A    -0.993263  0.196800 -1.136645
# B     0.000366  1.025984 -0.156598
# C    -0.031579  0.649826  2.154846

df['new'] = df['X'] + df['Z']
df
#              X         Y         Z       new
# A    -0.993263  0.196800 -1.136645 -2.129908
# B     0.000366  1.025984 -0.156598 -0.156231
# C    -0.031579  0.649826  2.154846  2.123267
```

```
pandas.DataFrame(data=None, index=None, columns=None, dtype=None, copy=False)
```

Pandas DataFrames (Indexing)

AiAT ACADEMY

```
df
#            X         Y         Z       new
# A  -0.993263  0.196800 -1.136645 -2.129908
# B   0.000366  1.025984 -0.156598 -0.156231
# C  -0.031579  0.649826  2.154846  2.123267

df.reset_index()
#       index         X         Y         Z       new
# 0         A  -0.993263  0.196800 -1.136645 -2.129908
# 1         B   0.000366  1.025984 -0.156598 -0.156231
# 2         C  -0.031579  0.649826  2.154846  2.123267
```

Pandas DataFrames (Indexing)

AiAT ACADEMY

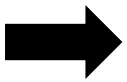
```
cities = ['Bangkok', 'Cuba', 'Tokyo']
df['Cities'] = cities
#           X         Y         Z       new    Cities
# A  -0.993263  0.196800 -1.136645 -2.129908   Bangkok
# B   0.000366  1.025984 -0.156598 -0.156231      Cuba
# C  -0.031579  0.649826  2.154846  2.123267     Tokyo

df.set_index('Cities')
#  Cities           X         Y         Z       new
# Bangkok  -0.993263  0.196800 -1.136645 -2.129908
# Cuba      0.000366  1.025984 -0.156598 -0.156231
# Tokyo    -0.031579  0.649826  2.154846  2.123267
```

Pandas DataFrames (Dropping)

AiAT ACADEMY

```
df
#              X         Y         Z       new
# A    -0.993263  0.196800 -1.136645 -2.129908
# B     0.000366  1.025984 -0.156598 -0.156231
# C    -0.031579  0.649826  2.154846  2.123267
```

```
df.drop('new', axis=1)
# while axis=1 is drop by column,
# axis=0 is drop by index (row)
#              X         Y         Z
# A    -0.993263  0.196800 -1.136645
# B     0.000366  1.025984 -0.156598
# C    -0.031579  0.649826  2.154846
```



```
df
#              X         Y         Z       new
# A    -0.993263  0.196800 -1.136645 -2.129908
# B     0.000366  1.025984 -0.156598 -0.156231
# C    -0.031579  0.649826  2.154846  2.123267
```

`df.drop('new', axis=1)` did not actually drop `df['new']` from `df`

```
DataFrame.drop(labels=None, axis=0, index=None, columns=None, level=None, inplace=False, errors='raise')
```

Pandas DataFrames (Dropping)

```
df
#            X         Y         Z       new
# A   -0.993263  0.196800 -1.136645 -2.129908
# B    0.000366  1.025984 -0.156598 -0.156231
# C   -0.031579  0.649826  2.154846  2.123267

df.drop('new', axis=1, inplace=True)
df
#            X         Y         Z
# A   -0.993263  0.196800 -1.136645
# B    0.000366  1.025984 -0.156598
# C   -0.031579  0.649826  2.154846
```

```
DataFrame.drop(labels=None, axis=0, index=None, columns=None, level=None, inplace=False, errors='raise')
```

Pandas DataFrames (Dropping)

```
df
#              X         Y         Z
# A    -0.993263  0.196800 -1.136645
# B     0.000366  1.025984 -0.156598
# C    -0.031579  0.649826  2.154846

df.drop('C', axis=0, inplace=True)      # permanently drop row (axis=0) 'C'
df
#              X         Y         Z
# A    -0.993263  0.196800 -1.136645
# B     0.000366  1.025984 -0.156598
```

```
DataFrame.drop(labels=None, axis=0, index=None, columns=None, level=None, inplace=False, errors='raise')
```

Pandas DataFrames

```
df
#            X         Y         Z
# A  -0.993263  0.196800 -1.136645
# B   0.000366  1.025984 -0.156598

df.shape                # (3, 4)

df.loc['A']                         df.loc['A', 'X']
# X    2.706850                     # 2.706850
# Y    0.628133
# Z    0.907969
# Name: A, dtype: float64
```

Pandas DataFrames (Boolean)

```
df
#           X         Y         Z
# A  0.740122  0.528813 -0.589001
# B  0.188695 -0.758872 -0.933237
# C  0.955057  0.190794  1.978757
```

```
booldf = df > 0
df
#       X      Y      Z
# A  True   True  False
# B  True  False  False
# C  True   True   True
```

```
df[df > 0]
#           X         Y         Z
# A  0.740122  0.528813       NaN
# B  0.188695       NaN       NaN
# C  0.955057  0.190794  1.978757
```

```
df.['Y'] > 0
# A    True
# B    False
# C    True
# Name: Y, dtype:bool
```

Pandas DataFrames (Boolean)

```
df.['Y'] > 0
# A    True
# B    False
# C    True
# Name: Y, dtype:bool

df[df['Y']>0]        # df['Y']>0: B is False so A and C should be printed
#           X         Y         Z
# A  0.740122  0.528813 -0.589001
# C  0.955057  0.190794  1.978757
```

Pandas DataFrames (Boolean)

```
df
#           X         Y         Z
# A  2.706850  0.628133  0.907969
# B  0.503826  0.651118 -0.319318
# C -0.848077  0.605965 -2.018168
```

(df['Y']>0) & (df['Z']>0) : True and True
Both are more than xero

(df['Y']>0) + (df['Z']>0)

```
df[(df['Y']>0) & (df['Z']>0)]
# Pandas use & instead of 'and' and use | instead of 'or'
#           X         Y         Z
# A  2.706850  0.628133  0.907969
```

Pandas DataFrames (Multiple Index)

AiAT ACADEMY

```
outside = ['G1', 'G1', 'G2', 'G2']
inside = [1, 2, 1, 2]
h_index = list(zip(outside, inside))
# [('G1', 1), ('G1', 2), ('G2', 1), ('G2', 2)]
h_index = pd.MultiIndex.from_tuples(h_index)
# MultiIndex(levels=[['G1', 'G2'], [1, 2]],
            codes=[[0, 0, 1, 1], [0, 1, 0, 1]])
df = pd.DataFrame(randn(4, 2), h_index, ['A', 'B'])
df
#               A         B
# G1 1  0.740122  0.528813
     2 -0.589001  0.188695
# G2 1 -0.758872 -0.933237
     2  0.955057  0.190794
```

Pandas DataFrames (Multiple Index)

AiAT ACADEMY

```
df
#               A         B
# G1 1  0.740122  0.528813
#    2 -0.589001  0.188695
# G2 1 -0.758872 -0.933237
#    2  0.955057  0.190794
```

```
df.loc['G1']
#           A         B
# 1  0.740122  0.528813
# 2 -0.589001  0.188695
```

```
df.loc['G1'].loc[1]
# A    0.740122
# B    0.528813
# Name: 1, dtype: float64
```

Pandas DataFrames (Multiple Index)

AiAT ACADEMY

```
df
#                A         B
# G1 1   0.740122  0.528813
     2  -0.589001  0.188695
# G2 1  -0.758872 -0.933237
     2   0.955057  0.190794

df.index.names = ['Group', 'Num']
#                    A         B
# Group Num
# G1    1     0.740122  0.528813
#       2    -0.589001  0.188695
# G2    1    -0.758872 -0.933237
#       2     0.955057  0.190794
```

Missing Data (NaN)

Pandas Missing Data

```
import numpy as np
import pandas as pd
d = {'A': [1, 2, np.nan], 'B': [5, np.nan, np.nan], 'C'
: [1, 2, 3]}
d = pd.DataFrame(d)
df
#      A    B  C
# 0  1.0  5.0  1
# 1  2.0  NaN  2
# 2  NaN  NaN  3
```

How to deal with **NaN**

Pandas Missing Data (Dropping)

AiAT ACADEMY

```
df
#      A    B  C
# 0  1.0  5.0  1
# 1  2.0  NaN  2
# 2  NaN  NaN  3

df.dropna()          # drops every NaN row and column
#      A    B  C
# 0  1.0  5.0  1
```

```
DataFrame.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)
```

Pandas Missing Data (Dropping)

```
df
#      A    B  C
# 0  1.0  5.0  1
# 1  2.0  NaN  2
# 2  NaN  NaN  3
```

```
df.dropna(axis=1)
# drops NaN column
#    C
# 0  1
# 1  2
# 2  3
```

```
df.dropna(thresh=2)
# drops every NaN row and \\
column that less than thresh
#      A    B  C
# 0  1.0  5.0  1
# 1  2.0  NaN  2
```

Pandas Missing Data (Filling NaN)

```
df
#      A    B  C
# 0  1.0  5.0  1
# 1  2.0  NaN  2
# 2  NaN  NaN  3

df.fillna(value="AIAT")
#      A    B  C
# 0  1.0  5.0  1
# 1  2.0  AIAT 2
# 2  AIAT AIAT 3
```

```
df.fillna(value=df.mean())
# Fill NaN with mean value of \\
each column
#      A    B  C
# 0  1.0  5.0  1
# 1  2.0  5.0  2
# 2  1.5  5.0  3
```

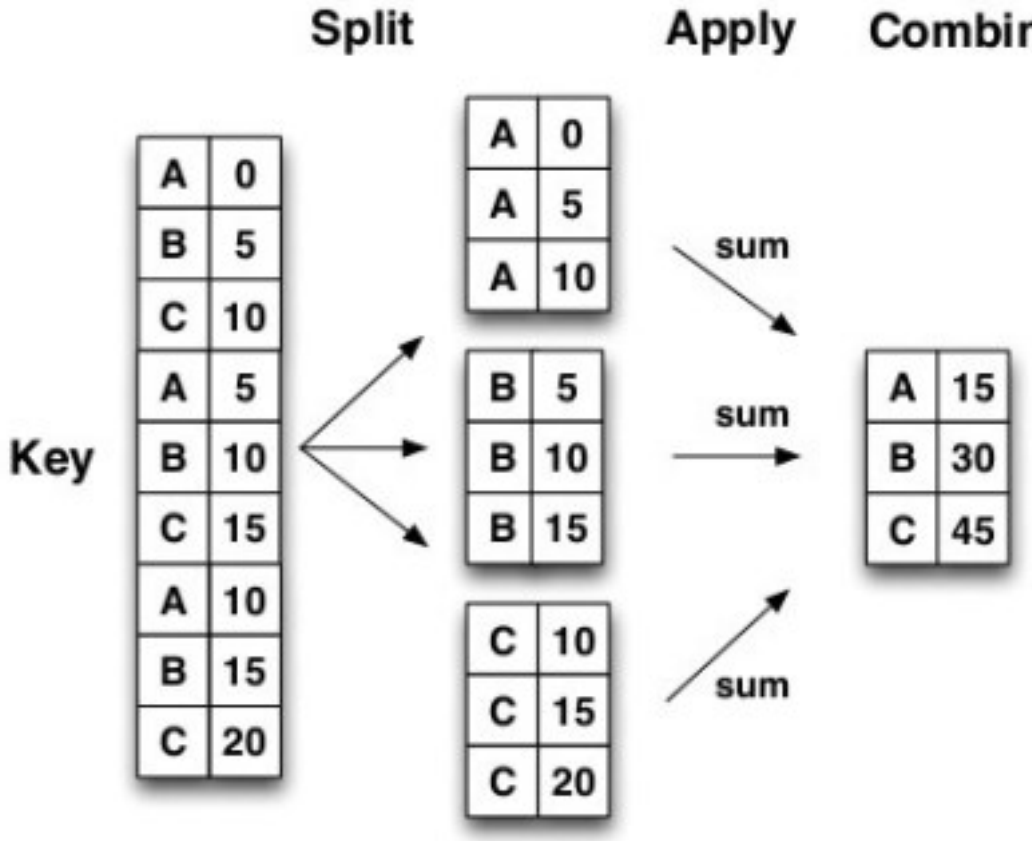
```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None, **kwargs)
```

AiAT ACADEMY

Groupby

Pandas groupby

- groupby allows you to group rows together from the column and perform an aggregate (รวม) function on them



Pandas groupby

```
import numpy as np
import pandas as pd
data = {'Institute': ['AIAT','AIAT','TU','TU','SIIT','SIIT'],
         'Person': ['Fluke','Thank','Got','Boat','Bi','Ben'],
         'Age': [21, 22, 24, 26, 30, 25]}
df = pd.DataFrame(data)
df
#   Institute Person  Age
# 0      AIAT  Fluke   21
# 1      AIAT  Thank   22
# 2        TU    Got   24
# 3        TU   Boat   26
# 4      SIIT     Bi   30
# 5      SIIT    Ben   25
```

Pandas groupby

```
by_ins = df.groupby("Institute")
by_ins.mean()
#              Age
# Institute
# AIAT        21.5
# SIIT        27.5
# TU          25.0
by_ins.std()
#                 Age
# Institute
# AIAT       0.707107
# SIIT       3.535534
# TU         1.414214
```

```
df
#   Institute Person  Age
# 0      AIAT  Fluke   21
# 1      AIAT  Thank   22
# 2        TU    Got   24
# 3        TU   Boat   26
# 4      SIIT     Bi   30
# 5      SIIT    Ben   25
```

Pandas groupby

- `df.groupby("Institute").max()`
- `df.groupby("Institute").min()`
- `df.groupby("Institute").sum()`
- `df.groupby("Institute").describe()`

```
#              Age
#            count  mean       std   min     25%   50%     75%   max
# Institute
# AIAT         2.0  21.5  0.707107  21.0  21.25  21.5  21.75  22.0
# SIIT         2.0  27.5  3.535534  25.0  26.25  27.5  28.75  30.0
# TU           2.0  25.0  1.414214  24.0  24.50  25.0  25.50  26.0
```

- **And more!!**

Concatenating, Merging, and Joining

Pandas Concatenating (Concat)

```
df1 = pd.DataFrame({'A': ['A0', 'A1'], 'B': ['B0', 'B1'],
                    'C': ['C0', 'C1'], 'D': ['D0', 'D1']},
                     index=[0, 1])
#      A   B   C   D
# 0   A0  B0  C0  D0
# 1   A1  B1  C1  D1
df2 = pd.DataFrame({'A': ['A2', 'A3'], 'B': ['B3', 'B3'],
                    'C': ['C2', 'C3'], 'D': ['D2', 'D3']},
                     index=[2, 3])
#      A   B   C   D
# 2   A2  B3  C2  D2
# 3   A3  B3  C3  D3
```

Pandas Concatenating (Concat)

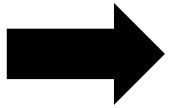
AiAT ACADEMY

df1

```
#      A    B    C    D
# 0   A0   B0   C0   D0
# 1   A1   B1   C1   D1
```

df2

```
#      A    B    C    D
# 2   A2   B3   C2   D2
# 3   A3   B3   C3   D3
```



```
pd.concat([df1, df2])
#      A    B    C    D
# 0   A0   B0   C0   D0
# 1   A1   B1   C1   D1
# 2   A2   B3   C2   D2
# 3   A3   B3   C3   D3
```

Pandas Concatenating (Concat)

```
pd.concat([df1, df2], axis=1)
#       A     B     C     D     A     B     C     D
# 0    A0    B0    C0    D0   NaN   NaN   NaN   NaN
# 1    A1    B1    C1    D1   NaN   NaN   NaN   NaN
# 2   NaN   NaN   NaN   NaN    A2    B3    C2    D2
# 3   NaN   NaN   NaN   NaN    A3    B3    C3    D3
```

Pandas Merging

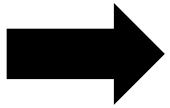
```
left = pd.DataFrame({'key': ['K0', 'K1'],
                     'A': ['A0', 'A1'],
                     'B': ['B0', 'B1']})
#   key   A   B
# 0  K0  A0  B0
# 1  K1  A1  B1
right = pd.DataFrame({'key': ['K0', 'K1'],
                      'C': ['C0', 'C1'],
                      'D': ['D0', 'D1']})
#   key   C   D
# 0  K0  C0  D0
# 1  K1  C1  D1
```

Pandas Merging

AiAT ACADEMY

```
left
#   key   A   B
# 0  K0   A0  B0
# 1  K1   A1  B1
```

```
right
#   key   C   D
# 0  K0   C0  D0
# 1  K1   C1  D1
```



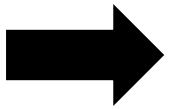
```
pd.merge(left,right,how='inner',on='key')
#   key   A   B   C   D
# 0  K0   A0  B0  C0  D0
# 1  K1   A1  B1  C1  D1
```

Pandas Joining

```
left = pd.DataFrame({'A': ['A0', 'A1'],
                     'B': ['B0', 'B1']},
                      index=['K0', 'K1'])
#       A   B
# K0   A0   B0
# K1   A1   B1
right = pd.DataFrame({'C': ['C0', 'C2'],
                      'D': ['D0', 'D2']},
                       index=['K0', 'K2'])
#       C   D
# K0   C0   D0
# K2   C2   D2
```

Pandas Joining

```
left
#       A    B
# K0   A0   B0
# K1   A1   B1
```



```
right
#       C    D
# K0   C0   D0
# K2   C2   D2
```

```
left.join(right)
#       A    B     C     D
# K0   A0   B0    C0    D0
# K1   A1   B1   NaN   NaN
```

Operations

Pandas Operations

AiAT ACADEMY

```
df = pd.DataFrame({'col1':[1,2,3,4],
                   'col2':[444,555,666,444],
                   'col3':['abc','def','ghi','xyz']})
```

```
df
#    col1  col2 col3
# 0     1   444  abc
# 1     2   555  def
# 2     3   666  ghi
# 3     4   444  xyz
```

Pandas Operations (Unique values)

AiAT ACADEMY

```
df
#    col1  col2 col3
# 0     1   444  abc
# 1     2   555  def
# 2     3   666  ghi
# 3     4   444  xyz


df['col2'].unique()   # get unique values from col2
# array([444, 555, 666])
df['col2'].unique()   # get no.unique values from col2
# 3
```

Pandas Operations (Values)

AiAT ACADEMY

```
df
#    col1  col2 col3
# 0     1   444  abc
# 1     2   555  def
# 2     3   666  ghi
# 3     4   444  xyz


df['col2'].value_count()  # get summary of no. values from col2
# 444    2
# 555    1
# 666    1
# Name: col2, dtype: int64
```

Pandas Operations (Function)

AiAT ACADEMY

```
df
#    col1  col2 col3
# 0     1   444  abc
# 1     2   555  def
# 2     3   666  ghi
# 3     4   444  xyz
```

```
def times2(x):
    return x * 2
```

```
df['col1'].apply(times2)
# 0    2
# 1    4
# 2    6
# 3    8
# Name: col1, dtype: int64
```

Pandas Operations (Sort)

```
df
#    col1  col2 col3
# 0     1   444  abc
# 1     2   555  def
# 2     3   666  ghi
# 3     4   444  xyz
```

```
df.sort_values(by='col2')
#    col1  col2 col3
# 0     1   444  abc
# 3     4   444  xyz
# 1     2   555  def
# 2     3   666  ghi
```

```
DataFrame.sort_values(by, axis=0, ascending=True, inplace=False, kind='quicksort', na_position='last')
```

Data Input/output

Pandas Data Input/output

- CSV
- Excel
- HTML
- SQL

Pandas Data Input/output dependencies

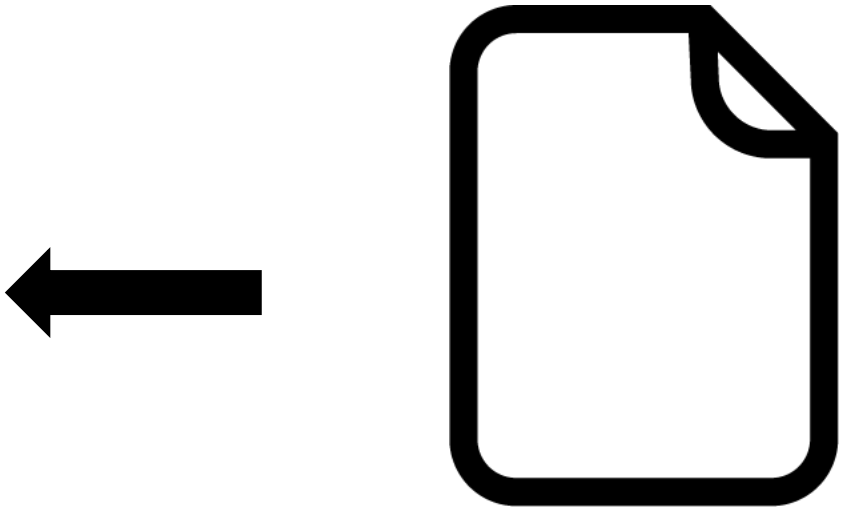
- You need to install required libraries for Pandas data input/output
- You can use pip or conda

```
conda install sqlalchemy
conda install lxml
conda install html5lib
conda install BeautifulSoup4
conda install xlrd
```

```
pip install sqlalchemy
pip install lxml
pip install html5lib
pip install BeautifulSoup4
pip install xlrd
```

Pandas Data Input/output (Read)

```
pd.read_csv('example.csv')
# Read file named 'example.csv' in current directory
# Default: same directory (folder) as .py file


#     a   b   c   d
# 0   0   1   2   3
# 1   4   5   6   7
# 2   8   9  10  11
# 3  12  13  14  15
```



example.csv

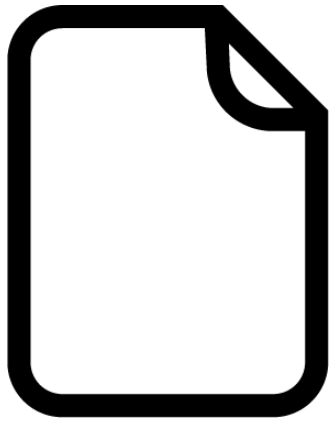
Pandas Data Input/output (Read CSV)

AiAT ACADEMY

```
df = pd.read_csv('example.csv')
df
# Read file named 'example.csv' in current directory
# Default: same directory (folder) as .py file


#     a   b   c   d
# 0   0   1   2   3
# 1   4   5   6   7
# 2   8   9  10  11
# 3  12  13  14  15
```

Pandas Data Input/output (Write CSV output)

```
df = pd.read_csv('example.csv')
#      a   b   c   d
# 0    0   1   2   3
# 1    4   5   6   7
# 2    8   9  10  11
# 3   12  13  14  15
```



my_output

```
df.to_csv('my_output', index=False)
# write dataframe from example.csv to a file named my_output
(excludes index)
```

Pandas Data Input/output (Read/write Excel)

AiAT ACADEMY

```
df = pd.read_excel('Excel_Sample.xlsx', sheet_name='Sheet1')
#    Unnamed: 0   a   b   c   d
# 0           0   0   1   2   3
# 1           1   4   5   6   7
# 2           2   8   9  10  11
# 3           3  12  13  14  15


df.to_excel('excel_sample2.xlsx', sheet_name='new')
# write dataframe from Excel_Sample.xlsx to a file named
excel_sample2.xlsx on a new sheet named new)
```