Topic 4
Reading and writing data on the filesystem

Learning Outcomes

After completing this topic and the recommended reading, you should be able to:

- Write Python programs that can read and write files in CSV and JSON formats.
- Describe different types of data files and evaluate their appropriateness for storing different types of data.
- Process data for purpose.

1. NumPy

- ***NumPy*** is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- https://numpy.org

Installing/Importing NumPy Library

- *conda install numpy*
- *pip install numpy*
- *import numpy as np*

NumPy Arrays (ndarray)

- An ***ndarray*** is a multi-dimensional container (array object) of items/elements (usually numbers) of the same type (homogenous) and size (fixed-sized).
- Examples:
  - *array0d = np.array(42)* *# 0-dimensional array*
  - *array1d = np.array([1, 2, 3, 4, 5])* *# 1-dimensional array*
  - *print(type(array1d))*

```
<class 'numpy.ndarray'>
```

  - *array2d = np.array([[1, 2, 3], [4, 5, 6]])* *# 2-dimensional array*
  - *print(array2d)*

```
[[1 2 3]
 [4 5 6]]
```

- Commonly used arrays
  - *a = np.zeros((2,2))* *# array of all zeros*
  - *print(a)*

```
[[0. 0.]
 [0. 0.]]
```

  - *b = np.ones((1,2))* *# array of all ones*
  - *print(b)*

```
[[1. 1.]]
```

  - *c = np.full((2,2), 7)* *# constant array*
  - *print(c)*

```
[[7 7]
 [7 7]]
```

  - *d = np.eye(2)* *# identity matrix*
  - *print(a)*

```
[[1. 0.]
 [0. 1.]]
```

- Random arrays
  - *np.random.seed(10)*
  - *e = np.random.random((2,2))* *# array with random values*
  - *print(np.round(e, 3))* *# round to 3 d.p.*

```
[[0.771 0.021]
 [0.634 0.749]]
```

- More arrays
  - *rng = np.arrange(10)*
  - *print(rng)*

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

  - *print(np.sqrt(rng))*

```
[0.         1.         1.41421356 1.73205081 2.         2.23606798
 2.44948974 2.64575131 2.82842712 3.        ]
```

NumPy Operations/Functions

- NumPy operations
  - *a = np.array([[1.0, 2.0, 4.0], [-1.0, 2.0, -5.0]])*
  - *print(a)*

```
[[ 1.  2.  4.]
 [-1.  2. -5.]]
```

  - *print(a.shape)*

```
(2, 3)
```

  - *print(a.sum(axis=0))* *# rows*

```
[ 0.  4. -1.]
```

  - *print(a.sum(axis=1))* *# columns*

```
[ 7. -4.]
```

  - *print(a.sum())* *# all*

```
3.0
```

- *b = np.transpose(a)*
- *print(b)*

```
[[ 1. -1.]
 [ 2.  2.]
 [ 4. -5.]]
```

- *print(b.shape)*

```
(3, 2)
```

- *print(np.dot(a,b))* *# a * b*

  print(a @ b)

```
[[ 21. -17.]
 [-17.  30.]]
```

- *print(a[0:2, 1:3])* *# subset*

```
[[ 2.  4.]
 [ 2. -5.]]
```

- *print(np.sum((a<0) & (a>1))* *# and => 0*
  # or "|" => 5

- NumPy universal functions
  - *a1 = np.array([1, 2, 3, 4, 5])*
  - *a2 = np.array([6, 7, 8, 9, 0])*
  - *np.add(a1,a2)* *# result: [7, 9, 11, 13, 5]*
  - *np.subtract(a1,a2)* *# result: [-5, -5, -5, -5, 5]*
  - *np.multiply(a1,a2)* *# result: [6, 14, 24, 36, 0]*

- *np.divide(a1,a2)* *# result: Error*
- *np.power(a1,a2)* *# result: [1, 128, 6561, 262144, 1]*
- *np.sum(a1,a2)* *# result: 45*
- *np.sum([a1,a2],axis=1)* *# sum the rows: [15, 30]*

2. Pandas

- ***Pandas*** is a fast, powerful, flexible and easy to use open source software library written for the Python programming language for data manipulation and analysis.
- It offers data structures and operations for manipulating numerical tables and time series, as series and/or data frame.
- https://pandas.pydata.org

Installing/Importing NumPy Library

- *conda install pandas*
- *pip install pandas*
- *import pandas as pd*

Pandas Series

- One-dimensional labelled array
- Syntax: "*pd.Series(data, index=index)*"
    - *np.random.seed(1)*
    - *s = pd.Series(np.random.randn(5), index = ['a', 'b', 'c', 'd', 'e']*

```
a    1.624345
b   -0.611756
c   -0.528172
d   -1.072969
e    0.865408
dtype: float64
```

Pandas Data Frames

- Two-dimensional labelled data structure

- Syntax: "*pd.DataDrame(data)*"
  - *d = {'one': [1., 2., 3., 4.], 'two': [4., 3., 2., 1.]}*
  - *df = pd.DataFrame(d)*

	one	two
0	1.0	4.0
1	2.0	3.0
2	3.0	2.0
3	4.0	1.0

  - *df.describe()*

	one	two
count	4.000000	4.000000
mean	2.500000	2.500000
std	1.290994	1.290994
min	1.000000	1.000000
25%	1.750000	1.750000
50%	2.500000	2.500000
75%	3.250000	3.250000
max	4.000000	4.000000

  - print(df['one'])

```
0    1.0
1    2.0
2    3.0
3    4.0
Name: one, dtype: float64
```

3. Data Wrangling in Python

- Sometimes referred to as ***data munging*** or ***data carpentry***.
- The process of cleaning, unifying, transforming, and/or mapping messy and complex data sets from one raw form into another format.
- The intention is for making it more appropriate and valuable for easy access and analysis.
- Main data wrangling activities:
    - Discovering patterns in data
        - E.g., identifying correlations and patterns
    - Structuring data
        - E.g., sub-setting, merging, re-ordering, transforming, reshaping
    - Cleaning and validating data
        - E.g., identifying missing or mis-recorded data
    - Enriching data
        - E.g., combining other data sources

Reading Data into Data Frame

- Reading "*airquality.csv*" into data frame, rearrange the columns to have 'Month' and 'Day' in the first two columns.

airquality.csv

```
"Ozone","Solar.R","Wind","Temp","Month","Day"
41,190,7.4,67,5,1
36,118,8,72,5,2
12,149,12.6,74,5,3
18,313,11.5,62,5,4
NA,NA,14.3,56,5,5
28,NA,14.9,66,5,6
23,299,8.6,65,5,7
19,99,13.8,59,5,8
8,19,20.1,61,5,9
NA,194,8.6,69,5,10
```

```
import pandas as pd
import numpy as np

airquality = pd.read_csv("airquality.csv")
airquality = airquality[['Month','Day','Ozone','Solar.R','Temp','Wind']]
airquality.head()
airquality.tail()
```

- The first and last parts of "*airquality*" data frame

	Month	Day	Ozone	Solar.R	Temp	Wind
0	5	1	41.0	190.0	67	7.4
1	5	2	36.0	118.0	72	8.0
2	5	3	12.0	149.0	74	12.6
3	5	4	18.0	313.0	62	11.5
4	5	5	NaN	NaN	56	14.3

	Month	Day	Ozone	Solar.R	Temp	Wind
148	9	26	30.0	193.0	70	6.9
149	9	27	NaN	145.0	77	13.2
150	9	28	14.0	191.0	75	14.3
151	9	29	18.0	131.0	76	8.0
152	9	30	20.0	223.0	68	11.5

Reshaping Data Sets

- Wide to Long: ***pandas.melt()***

- No id variables, default

```
pd.melt(airquality)
```

```
    variable  value
0      Month    5.0
1      Month    5.0
2      Month    5.0
3      Month    5.0
4      Month    5.0
..       ...    ...
913     Wind    6.9
914     Wind   13.2
915     Wind   14.3
916     Wind    8.0
917     Wind   11.5

[918 rows x 2 columns]
```

- Specify id variables, value variables, variable name, and value name.

```
airquality_long = pd.melt(airquality,
                          id_vars = ['Month','Day'],
                          value_vars = ['Ozone','Solar.R','Wind','Temp'],
                          var_name = 'climate_var',
                          value_name = 'climate_value')
airquality_long.head()
airquality_long.tail()
```

```
   Month  Day climate_var  climate_value
0      5    1       Ozone           41.0
1      5    2       Ozone           36.0
2      5    3       Ozone           12.0
3      5    4       Ozone           18.0
4      5    5       Ozone            NaN
```

```
     Month  Day climate_var  climate_value
607      9   26        Temp           70.0
608      9   27        Temp           77.0
609      9   28        Temp           75.0
610      9   29        Temp           76.0
611      9   30        Temp           68.0
```

- Long to Wide: ***pivot_table()***

- Multindex:

```
airquality_wide = airquality_long.pivot_table(values = 'climate_value',
                                              index = ['Month','Day'],
                                              columns = ['climate_var'])
airquality_wide.head()
airquality_wide.tail()
```

climate_var		Ozone	Solar.R	Temp	Wind
Month	Day				
5	1	41.0	190.0	67.0	7.4
	2	36.0	118.0	72.0	8.0
	3	12.0	149.0	74.0	12.6
	4	18.0	313.0	62.0	11.5
	5	NaN	NaN	56.0	14.3

climate_var		Ozone	Solar.R	Temp	Wind
Month	Day				
9	26	30.0	193.0	70.0	6.9
	27	NaN	145.0	77.0	13.2
	28	14.0	191.0	75.0	14.3
	29	18.0	131.0	76.0	8.0
	30	20.0	223.0	68.0	11.5

- Aggregation:

```
airquality_month1 = airquality_long.pivot_table(values = 'climate_value',
                                                index = ['Month'],
                                                columns = ['climate_var'])
airquality_month2 = airquality_long.pivot_table(values = 'climate_value',
                                                index = ['Month'],
                                                columns = ['climate_var'],
                                                aggfunc = np.mean)
airquality_month1.head()
```

climate_var	Ozone	Solar.R	Temp	Wind
Month				
5	23.615385	181.296296	65.548387	11.622581
6	29.444444	190.166667	79.100000	10.266667
7	59.115385	216.483871	83.903226	8.941935
8	59.961538	171.857143	83.967742	8.793548
9	31.448276	167.433333	76.900000	10.180000

```
airquality_month3 = airquality_long.pivot_table(values = 'climate_value',
                                                index = ['Month'],
                                                columns = ['climate_var'],
                                                aggfunc = np.max)
airquality_month3.head()
```

climate_var	Ozone	Solar.R	Temp	Wind
Month				
5	115.0	334.0	81.0	20.1
6	71.0	332.0	93.0	20.7
7	135.0	314.0	92.0	14.9
8	168.0	273.0	97.0	15.5
9	96.0	259.0	93.0	16.6

Stacking Data Sets

- ***stack()***

```
airquality_stack = airquality_wide.stack()
airquality_stack
```

```
Month  Day  climate_var
5      1    Ozone          41.0
            Solar.R       190.0
            Temp           67.0
            Wind            7.4
       2    Ozone          36.0
                           ...
9      29   Wind            8.0
       30   Ozone          20.0
            Solar.R       223.0
            Temp           68.0
            Wind           11.5
Length: 568, dtype: float64
```

- ***unstack()***

```
airquality_stack.unstack()
```

climate_var	Ozone	Solar.R	Temp	Wind
Month Day				
5 1	41.0	190.0	67.0	7.4
2	36.0	118.0	72.0	8.0
3	12.0	149.0	74.0	12.6
4	18.0	313.0	62.0	11.5
5	NaN	NaN	56.0	14.3
...	...	...	...	...
9 26	30.0	193.0	70.0	6.9
27	NaN	145.0	77.0	13.2
28	14.0	191.0	75.0	14.3
29	18.0	131.0	76.0	8.0
30	20.0	223.0	68.0	11.5

Subsetting Data Sets

- Example:

```
airquality

airquality['Ozone']              # subsetting column, returns a Series
airquality[['Ozone']]            # subsetting column, returns a DataFrame

airquality[['Ozone','Temp']]        # subsetting more than a column
airquality.iloc[:,np.array([2,4])]  # same as above

airquality[:5]                   # subsetting rows 0 to 4
airquality.iloc[:5]              # same as above
airquality.loc[:5]               # subsetting rows 0 to 5

airquality.iloc[:5, 2]           # rows 0 to 4 and column 2
airquality.loc[:5, 'Ozone']      # rows 0 to 5 and column 'Ozone'
```

	Month	Day	Ozone	Solar.R	Temp	Wind
0	5	1	41.0	190.0	67	7.4
1	5	2	36.0	118.0	72	8.0
2	5	3	12.0	149.0	74	12.6
3	5	4	18.0	313.0	62	11.5
4	5	5	NaN	NaN	56	14.3
..	...	...	...	...	...	...
148	9	26	30.0	193.0	70	6.9
149	9	27	NaN	145.0	77	13.2
150	9	28	14.0	191.0	75	14.3
151	9	29	18.0	131.0	76	8.0
152	9	30	20.0	223.0	68	11.5

- With condition:

```
airquality.loc[airquality.Day == 3, 'Ozone']
```

```
2      12.0
33      NaN
63     32.0
94     16.0
125    73.0
Name: Ozone, dtype: float64
```

- Multindex:

```
airquality_wide

airquality_wide.iloc[:5]          # subsetting first 5 rows
airquality_wide.loc[:5]           # subsetting month 5
```

```
climate_var  Ozone  Solar.R  Temp  Wind
Month Day
5     1       41.0    190.0  67.0   7.4
      2       36.0    118.0  72.0   8.0
      3       12.0    149.0  74.0  12.6
      4       18.0    313.0  62.0  11.5
      5        NaN      NaN  56.0  14.3
...            ...      ...   ...   ...
9     26      30.0    193.0  70.0   6.9
      27       NaN    145.0  77.0  13.2
      28      14.0    191.0  75.0  14.3
      29      18.0    131.0  76.0   8.0
      30      20.0    223.0  68.0  11.5

[153 rows x 4 columns]
```

- Missing values:

```
airquality.head(10)
```

```
   Month  Day  Ozone  Solar.R  Temp  Wind
0      5    1   41.0    190.0    67   7.4
1      5    2   36.0    118.0    72   8.0
2      5    3   12.0    149.0    74  12.6
3      5    4   18.0    313.0    62  11.5
4      5    5    NaN      NaN    56  14.3
5      5    6   28.0      NaN    66  14.9
6      5    7   23.0    299.0    65   8.6
7      5    8   19.0     99.0    59  13.8
8      5    9    8.0     19.0    61  20.1
9      5   10    NaN    194.0    69   8.6
```

```
airquality_no_na = airquality.dropna()  # remove rows with 'NaN'
airquality_no_na.head(10)
```

	Month	Day	Ozone	Solar.R	Temp	Wind
0	5	1	41.0	190.0	67	7.4
1	5	2	36.0	118.0	72	8.0
2	5	3	12.0	149.0	74	12.6
3	5	4	18.0	313.0	62	11.5
6	5	7	23.0	299.0	65	8.6
7	5	8	19.0	99.0	59	13.8
8	5	9	8.0	19.0	61	20.1
11	5	12	16.0	256.0	69	9.7
12	5	13	11.0	290.0	66	9.2
13	5	14	14.0	274.0	68	10.9

```
airquality_fill_na = airquality.fillna(method='ffill')  # fill missing values
airquality_fill_na.head(10)                             # with previous values
```

	Month	Day	Ozone	Solar.R	Temp	Wind
0	5	1	41.0	190.0	67	7.4
1	5	2	36.0	118.0	72	8.0
2	5	3	12.0	149.0	74	12.6
3	5	4	18.0	313.0	62	11.5
4	5	5	18.0	313.0	56	14.3
5	5	6	28.0	313.0	66	14.9
6	5	7	23.0	299.0	65	8.6
7	5	8	19.0	99.0	59	13.8
8	5	9	8.0	19.0	61	20.1
9	5	10	8.0	194.0	69	8.6

```
airquality_linear = airquality.interpolate(method='linear')
airquality_linear.head(10)                  # fill using interpolation
```

	Month	Day	Ozone	Solar.R	Temp	Wind
0	5	1	41.0	190.000000	67	7.4
1	5	2	36.0	118.000000	72	8.0
2	5	3	12.0	149.000000	74	12.6
3	5	4	18.0	313.000000	62	11.5
4	5	5	23.0	308.333333	56	14.3
5	5	6	28.0	303.666667	66	14.9
6	5	7	23.0	299.000000	65	8.6
7	5	8	19.0	99.000000	59	13.8
8	5	9	8.0	19.000000	61	20.1
9	5	10	7.5	194.000000	69	8.6

Combining Different Data Sets

- Using "*covid_new_cases.csv*" and "*covid_new_deaths.csv*" as data source.

covid_new_cases.csv

```
date,new_cases
2020-01-30,2
2020-01-31,0
2020-02-01,0
2020-02-02,0
2020-02-03,0
2020-02-04,0
2020-02-05,1
2020-02-06,0
2020-02-07,0
2020-02-08,4
```

covid_new_deaths.csv

```
date,new_deaths
2020-02-29,0
2020-03-01,0
2020-03-02,1
2020-03-03,2
2020-03-04,0
2020-03-05,3
2020-03-06,0
2020-03-07,0
2020-03-08,3
2020-03-09,5
```

- Reading "*covid_new_cases.csv*" and "*covid_new_deaths.csv*" into data frames.

```
cases_df = pd.read_csv("covid_new_cases.csv", index_col = 0)
deaths_df = pd.read_csv("covid_new_deaths.csv", index_col = 0)
```

```
            new_cases
date
2020-01-30          2
2020-01-31          0
2020-02-01          0
2020-02-02          0
2020-02-03          0
...               ...
2020-10-25      15654
2020-10-26      26467
2020-10-27      23757
2020-10-28      22887
2020-10-29      19525

[274 rows x 1 columns]
```

```
            new_deaths
date
2020-02-29           0
2020-03-01           0
2020-03-02           1
2020-03-03           2
2020-03-04           0
...                ...
2020-10-25         234
2020-10-26         253
2020-10-27         227
2020-10-28         216
2020-10-29         217

[244 rows x 1 columns]
```

- Left join

```
df_1 = cases_df.join(deaths_df)
```

```
            new_cases  new_deaths
date
2020-01-30          2         NaN
2020-01-31          0         NaN
2020-02-01          0         NaN
2020-02-02          0         NaN
2020-02-03          0         NaN
...               ...         ...
2020-10-25      15654       234.0
2020-10-26      26467       253.0
2020-10-27      23757       227.0
2020-10-28      22887       216.0
2020-10-29      19525       217.0

[274 rows x 2 columns]
```

- Right join

```
df_2 = cases_df.join(deaths_df, how='right')
```

```
            new_cases  new_deaths
date
2020-02-29          5           0
2020-03-01         22           0
2020-03-02         40           1
2020-03-03         56           2
2020-03-04         56           0
...               ...         ...
2020-10-25      15654         234
2020-10-26      26467         253
2020-10-27      23757         227
2020-10-28      22887         216
2020-10-29      19525         217

[244 rows x 2 columns]
```

- Inner join

```
df_3 = cases_df.join(deaths_df, how='inner')
```

```
            new_cases  new_deaths
date
2020-02-29          5           0
2020-03-01         22           0
2020-03-02         40           1
2020-03-03         56           2
2020-03-04         56           0
...               ...         ...
2020-10-25      15654         234
2020-10-26      26467         253
2020-10-27      23757         227
2020-10-28      22887         216
2020-10-29      19525         217

[244 rows x 2 columns]
```

4. Exercises

4.009 Practicing with dataframes

- Refers to “4.009 EDA.html”

4.010 Lists and Arrays

- Refers to “4.010 lists_and_arrays.html”

5. Practice Quiz

- Work on *Practice Quiz 04* posted on Canvas.

Useful Resources

-
    - http://