Pandas

- FAQ
- Common mistakes
- Reading documentation + Google Search + Fixing bugs
- Code walkthrough of common operations
- Cheat Sheet
- Exercise problems: https://www.machinelearningplus.com/python/101-pandas-exercises-python/ (https://www.machinelearningplus.com/python/101-pandas-exercises-python/)
- We will see many more examples in the next Code-Walkthrough sessions when we use Pandas in other chapters.

Pre-requisites: LIVE Sessions in Python programming + Course videos

In [127]:

```
# now we will see in pandas from nyc_weather.csv in the same folder
import pandas as pd
df = pd.read_csv("./nyc_weather.csv")
df
```

Out[127]:

	EST	Temperature	DewPoint	Humidity	Sea Level PressureIn	VisibilityMiles	WindSpeedMPH
0	1/1/2016	38	23	52	30.03	10	8.0
1	1/2/2016	36	18	46	30.02	10	7.0
2	1/3/2016	40	21	47	29.86	10	8.0
3	1/4/2016	25	9	44	30.05	10	9.0
4	1/5/2016	20	-3	41	30.57	10	5.0
5	1/6/2016	33	4	35	30.50	10	4.0
6	1/7/2016	39	11	33	30.28	10	2.0
7	1/8/2016	39	29	64	30.20	10	4.0
8	1/9/2016	44	38	77	30.16	9	8.0
9	1/10/2016	50	46	71	29.59	4	NaN
10	1/11/2016	33	8	37	29.92	10	NaN
11	1/12/2016	35	15	53	29.85	10	6.0
12	1/13/2016	26	4	42	29.94	10	10.0
13	1/14/2016	30	12	47	29.95	10	5.0
14	1/15/2016	43	31	62	29.82	9	5.0
15	1/16/2016	47	37	70	29.52	8	7.0
16	1/17/2016	36	23	66	29.78	8	6.0
17	1/18/2016	25	6	53	29.83	9	12.0
18	1/19/2016	22	3	42	30.03	10	11.0
19	1/20/2016	32	15	49	30.13	10	6.0
20	1/21/2016	31	11	45	30.15	10	6.0
21	1/22/2016	26	6	41	30.21	9	NaN
22	1/23/2016	26	21	78	29.77	1	16.0
23	1/24/2016	28	11	53	29.92	8	6.0
24	1/25/2016	34	18	54	30.25	10	3.0
25	1/26/2016	43	29	56	30.03	10	7.0
26	1/27/2016	41	22	45	30.03	10	7.0
27	1/28/2016	37	20	51	29.90	10	5.0
28	1/29/2016	36	21	50	29.58	10	8.0
29	1/30/2016	34	16	46	30.01	10	7.0
30	1/31/2016	46	28	52	29.90	10	5.0

In [2]:

```
# Common-mistake

# Read the documentation:https://pandas.pydata.org/pandas-docs/version/0.21.1/generated/pandas.read_csv.html
# Load data from Google Drive Link
import pandas as pd
df = pd.read_csv("https://drive.google.com/file/d/1KxwFsL6IF7OD_XN28kjxl0-amnELIhZ8/view?usp=sharing")
df
```

```
---------------------------------------------------------------------
-------
ParserError                                 Traceback (most recent cal
l last)
<ipython-input-2-9fe4f2b140df> in <module>
      3 # Load data from Google Drive Link
      4 import pandas as pd
----> 5 df = pd.read_csv("https://drive.google.com/file/d/1KxwFsL6IF
7OD_XN28kjxl0-amnELIhZ8/view?usp=sharing")
      6 df

/usr/local/lib/python3.7/site-packages/pandas/io/parsers.py in parse
r_f(filepath_or_buffer, sep, delimiter, header, names, index_col, us
ecols, squeeze, prefix, mangle_dupe_cols, dtype, engine, converters,
true_values, false_values, skipinitialspace, skiprows, skipfooter, n
rows, na_values, keep_default_na, na_filter, verbose, skip_blank_lin
es, parse_dates, infer_datetime_format, keep_date_col, date_parser,
 dayfirst, iterator, chunksize, compression, thousands, decimal, lin
eterminator, quotechar, quoting, doublequote, escapechar, comment, e
ncoding, dialect, tupleize_cols, error_bad_lines, warn_bad_lines, de
lim_whitespace, low_memory, memory_map, float_precision)
    700                     skip_blank_lines=skip_blank_lines)
    701
--> 702         return _read(filepath_or_buffer, kwds)
    703
    704     parser_f.__name__ = name

/usr/local/lib/python3.7/site-packages/pandas/io/parsers.py in _read
(filepath_or_buffer, kwds)
    433
    434     try:
--> 435         data = parser.read(nrows)
    436     finally:
    437         parser.close()

/usr/local/lib/python3.7/site-packages/pandas/io/parsers.py in read
(self, nrows)
   1137     def read(self, nrows=None):
   1138         nrows = _validate_integer('nrows', nrows)
-> 1139         ret = self._engine.read(nrows)
   1140
   1141         # May alter columns / col_dict

/usr/local/lib/python3.7/site-packages/pandas/io/parsers.py in read
(self, nrows)
   1993     def read(self, nrows=None):
   1994         try:
-> 1995             data = self._reader.read(nrows)
   1996         except StopIteration:
   1997             if self._first_chunk:

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader.read()

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._read_lo
w_memory()

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._read_ro
ws()

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._tokeniz
e_rows()
```

```
pandas/_libs/parsers.pyx in pandas._libs.parsers.raise_parser_error
()

ParserError: Error tokenizing data. C error: Expected 283 fields in
 line 131, saw 409
```

In [ ]:

```
```

In [24]:

```
# Q. Raw-strings for filenames.
import pandas as pd
df = pd.read_csv(r'./nyc_weather.csv')


raw_s = r'Hi\nHello'
print(raw_s)

s = 'Hi\nHello'
print(s)
```

```
Hi\nHello
Hi
Hello
```

Dealing with file-paths across various OS.

- Refer: https://medium.com/@ageitgey/python-3-quick-tip-the-easy-way-to-deal-with-file-paths-on-windows-mac-and-linux-11a072b58d5f (https://medium.com/@ageitgey/python-3-quick-tip-the-easy-way-to-deal-with-file-paths-on-windows-mac-and-linux-11a072b58d5f)

Windows filenames: C:\some_folder\some_file.txt

Most other operating systems: /some_folder/some_file.txt

In [136]:

```
# Python 3 has pathlib to simplify things
from pathlib import Path

data_folder = Path(".") # Just use forward slashes for all OS.

file_to_open = data_folder / "nyc_weather.csv"
df = pd.read_csv(file_to_open)
```

In [26]:

```
df
```

Out[26]:

	EST	Temperature	DewPoint	Humidity	Sea Level PressureIn	VisibilityMiles	WindSpeedMPH
0	1/1/2016	38	23	52	30.03	10	8.0
1	1/2/2016	36	18	46	30.02	10	7.0
2	1/3/2016	40	21	47	29.86	10	8.0
3	1/4/2016	25	9	44	30.05	10	9.0
4	1/5/2016	20	-3	41	30.57	10	5.0
5	1/6/2016	33	4	35	30.50	10	4.0
6	1/7/2016	39	11	33	30.28	10	2.0
7	1/8/2016	39	29	64	30.20	10	4.0
8	1/9/2016	44	38	77	30.16	9	8.0
9	1/10/2016	50	46	71	29.59	4	NaN
10	1/11/2016	33	8	37	29.92	10	NaN
11	1/12/2016	35	15	53	29.85	10	6.0
12	1/13/2016	26	4	42	29.94	10	10.0
13	1/14/2016	30	12	47	29.95	10	5.0
14	1/15/2016	43	31	62	29.82	9	5.0
15	1/16/2016	47	37	70	29.52	8	7.0
16	1/17/2016	36	23	66	29.78	8	6.0
17	1/18/2016	25	6	53	29.83	9	12.0
18	1/19/2016	22	3	42	30.03	10	11.0
19	1/20/2016	32	15	49	30.13	10	6.0
20	1/21/2016	31	11	45	30.15	10	6.0
21	1/22/2016	26	6	41	30.21	9	NaN
22	1/23/2016	26	21	78	29.77	1	16.0
23	1/24/2016	28	11	53	29.92	8	6.0
24	1/25/2016	34	18	54	30.25	10	3.0
25	1/26/2016	43	29	56	30.03	10	7.0
26	1/27/2016	41	22	45	30.03	10	7.0
27	1/28/2016	37	20	51	29.90	10	5.0
28	1/29/2016	36	21	50	29.58	10	8.0
29	1/30/2016	34	16	46	30.01	10	7.0
30	1/31/2016	46	28	52	29.90	10	5.0

In [5]:

```
df
```

Out[5]:

	EST	Temperature	DewPoint	Humidity	Sea Level PressureIn	VisibilityMiles	WindSpeedMPH
0	1/1/2016	38	23	52	30.03	10	8.0
1	1/2/2016	36	18	46	30.02	10	7.0
2	1/3/2016	40	21	47	29.86	10	8.0
3	1/4/2016	25	9	44	30.05	10	9.0
4	1/5/2016	20	-3	41	30.57	10	5.0
5	1/6/2016	33	4	35	30.50	10	4.0
6	1/7/2016	39	11	33	30.28	10	2.0
7	1/8/2016	39	29	64	30.20	10	4.0
8	1/9/2016	44	38	77	30.16	9	8.0
9	1/10/2016	50	46	71	29.59	4	NaN
10	1/11/2016	33	8	37	29.92	10	NaN
11	1/12/2016	35	15	53	29.85	10	6.0
12	1/13/2016	26	4	42	29.94	10	10.0
13	1/14/2016	30	12	47	29.95	10	5.0
14	1/15/2016	43	31	62	29.82	9	5.0
15	1/16/2016	47	37	70	29.52	8	7.0
16	1/17/2016	36	23	66	29.78	8	6.0
17	1/18/2016	25	6	53	29.83	9	12.0
18	1/19/2016	22	3	42	30.03	10	11.0
19	1/20/2016	32	15	49	30.13	10	6.0
20	1/21/2016	31	11	45	30.15	10	6.0
21	1/22/2016	26	6	41	30.21	9	NaN
22	1/23/2016	26	21	78	29.77	1	16.0
23	1/24/2016	28	11	53	29.92	8	6.0
24	1/25/2016	34	18	54	30.25	10	3.0
25	1/26/2016	43	29	56	30.03	10	7.0
26	1/27/2016	41	22	45	30.03	10	7.0
27	1/28/2016	37	20	51	29.90	10	5.0
28	1/29/2016	36	21	50	29.58	10	8.0
29	1/30/2016	34	16	46	30.01	10	7.0
30	1/31/2016	46	28	52	29.90	10	5.0

In [128]:

```
#Q. what's the difference betn df['EST'] & df[['EST']]


print(df['EST'])
```

```
0      1/1/2016
1      1/2/2016
2      1/3/2016
3      1/4/2016
4      1/5/2016
5      1/6/2016
6      1/7/2016
7      1/8/2016
8      1/9/2016
9     1/10/2016
10    1/11/2016
11    1/12/2016
12    1/13/2016
13    1/14/2016
14    1/15/2016
15    1/16/2016
16    1/17/2016
17    1/18/2016
18    1/19/2016
19    1/20/2016
20    1/21/2016
21    1/22/2016
22    1/23/2016
23    1/24/2016
24    1/25/2016
25    1/26/2016
26    1/27/2016
27    1/28/2016
28    1/29/2016
29    1/30/2016
30    1/31/2016
Name: EST, dtype: object
```

In [8]:

```
print(df[['EST']])
```

```
          EST
0    1/1/2016
1    1/2/2016
2    1/3/2016
3    1/4/2016
4    1/5/2016
5    1/6/2016
6    1/7/2016
7    1/8/2016
8    1/9/2016
9   1/10/2016
10  1/11/2016
11  1/12/2016
12  1/13/2016
13  1/14/2016
14  1/15/2016
15  1/16/2016
16  1/17/2016
17  1/18/2016
18  1/19/2016
19  1/20/2016
20  1/21/2016
21  1/22/2016
22  1/23/2016
23  1/24/2016
24  1/25/2016
25  1/26/2016
26  1/27/2016
27  1/28/2016
28  1/29/2016
29  1/30/2016
30  1/31/2016
```

In [129]:

```
print(type(df['EST'])) # Series is a vector

print(type(df[['EST']])) # DF is a sequence of series objects, DF:Matrix

#Refer: https://stackoverflow.com/a/26240208
#"So, the Series is the data structure for a single column of a DataFrame,
# not only conceptually, but literally, i.e.
# the data in a DataFrame is actually stored in memory as a collection of Serie
s."
```

```
<class 'pandas.core.series.Series'>
<class 'pandas.core.frame.DataFrame'>
```

In [15]:

```
print(df[['EST', 'Temperature']])
```

```
          EST  Temperature
0    1/1/2016           38
1    1/2/2016           36
2    1/3/2016           40
3    1/4/2016           25
4    1/5/2016           20
5    1/6/2016           33
6    1/7/2016           39
7    1/8/2016           39
8    1/9/2016           44
9   1/10/2016           50
10  1/11/2016           33
11  1/12/2016           35
12  1/13/2016           26
13  1/14/2016           30
14  1/15/2016           43
15  1/16/2016           47
16  1/17/2016           36
17  1/18/2016           25
18  1/19/2016           22
19  1/20/2016           32
20  1/21/2016           31
21  1/22/2016           26
22  1/23/2016           26
23  1/24/2016           28
24  1/25/2016           34
25  1/26/2016           43
26  1/27/2016           41
27  1/28/2016           37
28  1/29/2016           36
29  1/30/2016           34
30  1/31/2016           46
```

In [16]:

```
print(df['EST', 'Temperature'])
```

```
---------------------------------------------------------------------
-------
KeyError                                  Traceback (most recent cal
l last)
/usr/local/lib/python3.7/site-packages/pandas/core/indexes/base.py i
n get_loc(self, key, method, tolerance)
   2656             try:
-> 2657                 return self._engine.get_loc(key)
   2658             except KeyError:

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.Py
ObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.Py
ObjectHashTable.get_item()

KeyError: ('EST', 'Temperature')

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent cal
l last)
<ipython-input-16-cab765355b80> in <module>
----> 1 print(df['EST', 'Temperature'])

/usr/local/lib/python3.7/site-packages/pandas/core/frame.py in __get
item__(self, key)
   2925             if self.columns.nlevels > 1:
   2926                 return self._getitem_multilevel(key)
-> 2927             indexer = self.columns.get_loc(key)
   2928             if is_integer(indexer):
   2929                 indexer = [indexer]

/usr/local/lib/python3.7/site-packages/pandas/core/indexes/base.py i
n get_loc(self, key, method, tolerance)
   2657                 return self._engine.get_loc(key)
   2658             except KeyError:
-> 2659                 return self._engine.get_loc(self._maybe_cast
_indexer(key))
   2660         indexer = self.get_indexer([key], method=method, tol
erance=tolerance)
   2661         if indexer.ndim > 1 or indexer.size > 1:

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.Py
ObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.Py
ObjectHashTable.get_item()

KeyError: ('EST', 'Temperature')
```

In [20]:

```
# loc: label based accessing
# https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.loc.html
df.loc[:, ['EST', 'Temperature']] # list of column names
```

Out[20]:

	EST	Temperature
0	1/1/2016	38
1	1/2/2016	36
2	1/3/2016	40
3	1/4/2016	25
4	1/5/2016	20
5	1/6/2016	33
6	1/7/2016	39
7	1/8/2016	39
8	1/9/2016	44
9	1/10/2016	50
10	1/11/2016	33
11	1/12/2016	35
12	1/13/2016	26
13	1/14/2016	30
14	1/15/2016	43
15	1/16/2016	47
16	1/17/2016	36
17	1/18/2016	25
18	1/19/2016	22
19	1/20/2016	32
20	1/21/2016	31
21	1/22/2016	26
22	1/23/2016	26
23	1/24/2016	28
24	1/25/2016	34
25	1/26/2016	43
26	1/27/2016	41
27	1/28/2016	37
28	1/29/2016	36
29	1/30/2016	34
30	1/31/2016	46

In [21]:

```
#Q. want to know both the date and temperature, when it was raining

df[df['Events']=='Rain'][['EST','Temperature']]

# same as
df1 = df[df['Events']=='Rain'] # unneccsarily creating more variables and taking up more space.
df1[['EST','Temperature']]
```

Out[21]:

	EST	Temperature
8	1/9/2016	44
9	1/10/2016	50
15	1/16/2016	47
26	1/27/2016	41

In [132]:

```
print(df['Events']=='Rain')
```

```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8      True
9      True
10    False
11    False
12    False
13    False
14    False
15     True
16    False
17    False
18    False
19    False
20    False
21    False
22    False
23    False
24    False
25    False
26     True
27    False
28    False
29    False
30    False
Name: Events, dtype: bool
```

In [134]:

```
print(type(df[df['Events']=='Rain']))
```

```
<class 'pandas.core.frame.DataFrame'>
```

In [22]:

```
# another way suugested by  a student
df[['EST','Temperature']][df['Events']=='Rain']
```

Out[22]:

	EST	Temperature
8	1/9/2016	44
9	1/10/2016	50
15	1/16/2016	47
26	1/27/2016	41

In [27]:

```
#Q. Why does this not work for me?

df[["EST"]][df[["Events"]]=="Rain"]
```

Out[27]:

	EST
0	NaN
1	NaN
2	NaN
3	NaN
4	NaN
5	NaN
6	NaN
7	NaN
8	NaN
9	NaN
10	NaN
11	NaN
12	NaN
13	NaN
14	NaN
15	NaN
16	NaN
17	NaN
18	NaN
19	NaN
20	NaN
21	NaN
22	NaN
23	NaN
24	NaN
25	NaN
26	NaN
27	NaN
28	NaN
29	NaN
30	NaN

In [152]:

```
t1 = df[["Events"]]=="Rain"
t2 = df[["EST"]]
print(type(t1))
print(type(t2))

#print(t2[pd.Series(t1)])
print(type(t1.iloc[:,0])) # convert DF to Series [https://stackoverflow.com/questions/33246771/convert-pandas-data-frame-to-series]

print(t2[t1.iloc[:,0]])
```

```
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.series.Series'>
          EST
8    1/9/2016
9   1/10/2016
15  1/16/2016
26  1/27/2016
```

In [30]:

```
# Q. All columns with Events==Rain
df[df['Events'] == 'Rain']
```

Out[30]:

	EST	Temperature	DewPoint	Humidity	Sea Level PressureIn	VisibilityMiles	WindSpeedMPH
8	1/9/2016	44	38	77	30.16	9	8.0
9	1/10/2016	50	46	71	29.59	4	NaN
15	1/16/2016	47	37	70	29.52	8	7.0
26	1/27/2016	41	22	45	30.03	10	7.0

In [41]:

```
#Q. How does this work?

df['EST'][df.Events == 'Rain'] # indexing using boolean Series
```

Out[41]:

```
8      1/9/2016
9     1/10/2016
15    1/16/2016
26    1/27/2016
Name: EST, dtype: object
```

In [45]:

```
print(df.Events == 'Rain')
print(type(df.Events == 'Rain'))
```

```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8      True
9      True
10    False
11    False
12    False
13    False
14    False
15     True
16    False
17    False
18    False
19    False
20    False
21    False
22    False
23    False
24    False
25    False
26     True
27    False
28    False
29    False
30    False
Name: Events, dtype: bool
<class 'pandas.core.series.Series'>
```

In [46]:

```
print(type(df['EST']))
```

```
<class 'pandas.core.series.Series'>
```

In [49]:

```
#Q. What if individual values in a CSV have a comma? [Good boundary case]

#create a DF from list: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html

df_test= pd.DataFrame([[11,12,13,14],[21,22,23,24],[31,32,33,34]]) # constructor
print(df_test)
```

```
    0   1   2   3
0  11  12  13  14
1  21  22  23  24
2  31  32  33  34
```

In [51]:

```
df_test= pd.DataFrame([[11,12,13,14],[21,22,23,24],[31, 32, 33, 3,400]]) # const
ructor
print(df_test)
```

```
    0   1   2   3      4
0  11  12  13  14    NaN
1  21  22  23  24    NaN
2  31  32  33   3  400.0
```

In [52]:

```
df_test= pd.DataFrame([[11,12,13,14],[21,22,23,24],[31, 32, 33, '3,400']]) # con
structor
print(df_test)
```

```
    0   1   2      3
0  11  12  13     14
1  21  22  23     24
2  31  32  33  3,400
```

In [55]:

```
print(df_test.iloc[2,3]) # indexed location
print(type(df_test.iloc[2,3]))
```

```
3,400
<class 'str'>
```

In [59]:

```
#Q. Max temperature day: Alternative solutions

df['EST'][df['Temperature'] == df['Temperature'].max()]
```

Out[59]:

```
9    1/10/2016
Name: EST, dtype: object
```

In [63]:

```
df.EST[df.Temperature == df.Temperature.max()]
```

Out[63]:

```
9    1/10/2016
Name: EST, dtype: object
```

In [65]:

```
# Q. indexing errors
df[2:5]
```

Out[65]:

	EST	Temperature	DewPoint	Humidity	Sea Level PressureIn	VisibilityMiles	WindSpeedMPH	Pr
2	1/3/2016	40	21	47	29.86	10	8.0	
3	1/4/2016	25	9	44	30.05	10	9.0	
4	1/5/2016	20	-3	41	30.57	10	5.0	

In [160]:

```
print( df[2:5, 0:2] )
```

```
---------------------------------------------------------------------------
-------
TypeError                                 Traceback (most recent cal
l last)
<ipython-input-160-242e8437ef76> in <module>
----> 1 print( df[2:5, 0:2])

/usr/local/lib/python3.7/site-packages/pandas/core/frame.py in __get
item__(self, key)
   2925             if self.columns.nlevels > 1:
   2926                 return self._getitem_multilevel(key)
-> 2927             indexer = self.columns.get_loc(key)
   2928             if is_integer(indexer):
   2929                 indexer = [indexer]

/usr/local/lib/python3.7/site-packages/pandas/core/indexes/base.py i
n get_loc(self, key, method, tolerance)
   2655                                 'backfill or nearest lookup
s')
   2656             try:
-> 2657                 return self._engine.get_loc(key)
   2658             except KeyError:
   2659                 return self._engine.get_loc(self._maybe_cast
_indexer(key))

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

TypeError: '(slice(2, 5, None), slice(0, 2, None))' is an invalid ke
y
```

In [69]:

```
df.iloc [2:5, 0:2]
```

Out[69]:

	EST	Temperature
2	1/3/2016	40
3	1/4/2016	25
4	1/5/2016	20

In [72]:

```
df.iloc[0,0]
```

Out[72]:

```
'1/1/2016'
```

In [163]:

```
# Q. loc & iloc +ve and -ve indexing

df = pd.DataFrame([1,2,3,4,5,6,7,8,9,19], index=[49,48,47,46,45, 1, 4, 5, 3, 6])
df
```

Out[163]:

	0
49	1
48	2
47	3
46	4
45	5
1	6
4	7
5	8
3	9
6	19

In [164]:

```
print(df.loc[:3])
```

```
    0
49  1
48  2
47  3
46  4
45  5
1   6
4   7
5   8
3   9
```

In [165]:

```
print(df.loc[:45])
```

```
    0
49  1
48  2
47  3
46  4
45  5
```

In [166]:

```
print(df.iloc[:45])
```

```
     0
49   1
48   2
47   3
46   4
45   5
1    6
4    7
5    8
3    9
6   19
```

In [78]:

```
print(df.iloc[:3])
```

```
    0
49  1
48  2
47  3
```

In [167]:

```
print(df.iloc[:-3])
```

```
    0
49  1
48  2
47  3
46  4
45  5
1   6
4   7
```

In [80]:

```
print(df.loc[:-3])
```

```
---------------------------------------------------------------------
-------
ValueError                                Traceback (most recent cal
l last)
/usr/local/lib/python3.7/site-packages/pandas/core/indexes/base.py i
n get_slice_bound(self, label, side, kind)
   4804             try:
-> 4805                 return self._searchsorted_monotonic(label, s
ide)
   4806             except ValueError:

/usr/local/lib/python3.7/site-packages/pandas/core/indexes/base.py i
n _searchsorted_monotonic(self, label, side)
   4764
-> 4765         raise ValueError('index must be monotonic increasing
or decreasing')
   4766

ValueError: index must be monotonic increasing or decreasing

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent cal
l last)
<ipython-input-80-8925f1ad1ad4> in <module>
----> 1 print(df.loc[:-3])

/usr/local/lib/python3.7/site-packages/pandas/core/indexing.py in __
getitem__(self, key)
   1498
   1499             maybe_callable = com.apply_if_callable(key, self
.obj)
-> 1500             return self._getitem_axis(maybe_callable, axis=a
xis)
   1501
   1502     def _is_scalar_access(self, key):

/usr/local/lib/python3.7/site-packages/pandas/core/indexing.py in _g
etitem_axis(self, key, axis)
   1865         if isinstance(key, slice):
   1866             self._validate_key(key, axis)
-> 1867             return self._get_slice_axis(key, axis=axis)
   1868         elif com.is_bool_indexer(key):
   1869             return self._getbool_axis(key, axis=axis)

/usr/local/lib/python3.7/site-packages/pandas/core/indexing.py in _g
et_slice_axis(self, slice_obj, axis)
   1531         labels = obj._get_axis(axis)
   1532         indexer = labels.slice_indexer(slice_obj.start, slic
e_obj.stop,
-> 1533                                        slice_obj.step, kind=
self.name)
   1534
   1535         if isinstance(indexer, slice):

/usr/local/lib/python3.7/site-packages/pandas/core/indexes/base.py i
n slice_indexer(self, start, end, step, kind)
   4671         """
   4672         start_slice, end_slice = self.slice_locs(start, end,
step=step,
-> 4673                                                  kind=kind)
```

```
   4674
   4675         # return a slice

/usr/local/lib/python3.7/site-packages/pandas/core/indexes/base.py i
n slice_locs(self, start, end, step, kind)
   4876         end_slice = None
   4877         if end is not None:
-> 4878             end_slice = self.get_slice_bound(end, 'right', k
ind)
   4879         if end_slice is None:
   4880             end_slice = len(self)

/usr/local/lib/python3.7/site-packages/pandas/core/indexes/base.py i
n get_slice_bound(self, label, side, kind)
   4806             except ValueError:
   4807                 # raise the original KeyError
-> 4808                 raise err
   4809
   4810         if isinstance(slc, np.ndarray):

/usr/local/lib/python3.7/site-packages/pandas/core/indexes/base.py i
n get_slice_bound(self, label, side, kind)
   4800         # we need to look up the label
   4801         try:
-> 4802             slc = self._get_loc_only_exact_matches(label)
   4803         except KeyError as err:
   4804             try:

/usr/local/lib/python3.7/site-packages/pandas/core/indexes/base.py i
n _get_loc_only_exact_matches(self, key)
   4770         get_slice_bound.
   4771         """
-> 4772         return self.get_loc(key)
   4773
   4774     def get_slice_bound(self, label, side, kind):

/usr/local/lib/python3.7/site-packages/pandas/core/indexes/base.py i
n get_loc(self, key, method, tolerance)
   2657                 return self._engine.get_loc(key)
   2658             except KeyError:
-> 2659                 return self._engine.get_loc(self._maybe_cast
_indexer(key))
   2660         indexer = self.get_indexer([key], method=method, tol
erance=tolerance)
   2661         if indexer.ndim > 1 or indexer.size > 1:

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.In
t64HashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.In
t64HashTable.get_item()

KeyError: -3
```

In [168]:

```
print(df[2:5][1:3])
```

```
    0
46  4
45  5
```

In [176]:

```
print(type(df[2:5]))
print(df[2:5])
```

```
<class 'pandas.core.frame.DataFrame'>
    0
47  3
46  4
45  5
```

In [182]:

```
print(df[2:5].iloc[2])
```

```
0    5
Name: 45, dtype: int64
```

In [184]:

```
df = pd.DataFrame([1,2,3,4,5,6,7,8,9,19], index=[49,48,47,46,45, -3, 4, 5, 3, 6
])
print(df.loc[:-3])
```

```
     0
 49  1
 48  2
 47  3
 46  4
 45  5
-3   6
```

Cheat-sheet: Google drive link in course videos

Practice problems: https://www.machinelearningplus.com/python/101-pandas-exercises-python/ (https://www.machinelearningplus.com/python/101-pandas-exercises-python/)

- L1, L2 and L3
- For NumPy: https://www.machinelearningplus.com/python/101-numpy-exercises-python/ (https://www.machinelearningplus.com/python/101-numpy-exercises-python/)

In [81]:

```
#Difficulty Level: L1

#Combine ser1 and ser2 to form a dataframe.

import numpy as np
ser1 = pd.Series(list('abcedfghijklmnopqrstuvwxyz'))
ser2 = pd.Series(np.arange(26))
```

In [83]:

```
# Input
import numpy as np
ser1 = pd.Series(list('abcedfghijklmnopqrstuvwxyz'))
ser2 = pd.Series(np.arange(26))

# Solution 1
df = pd.concat([ser1, ser2], axis=1)
print(df)
print("**********************************************************")

# Solution 2
df = pd.DataFrame({'col1': ser1, 'col2': ser2})
print(df)
```

	0	1
0	a	0
1	b	1
2	c	2
3	e	3
4	d	4
5	f	5
6	g	6
7	h	7
8	i	8
9	j	9
10	k	10
11	l	11
12	m	12
13	n	13
14	o	14
15	p	15
16	q	16
17	r	17
18	s	18
19	t	19
20	u	20
21	v	21
22	w	22
23	x	23
24	y	24
25	z	25

	col1	col2
0	a	0
1	b	1
2	c	2
3	e	3
4	d	4
5	f	5
6	g	6
7	h	7
8	i	8
9	j	9
10	k	10
11	l	11
12	m	12
13	n	13
14	o	14
15	p	15
16	q	16
17	r	17
18	s	18
19	t	19
20	u	20
21	v	21
22	w	22
23	x	23
24	y	24
25	z	25

In [84]:

```
#Difficulty Level: L2

#From ser1 remove items present in ser2.


ser1 = pd.Series([1, 2, 3, 4, 5])
ser2 = pd.Series([4, 5, 6, 7, 8])
```

In [85]:

```
# Google search: "pandas series set difference" ----> https://medium.com/@rodwa
n.bakkar/pandas-set-difference-fde7f4381b53

ser1[ ~ ser1.isin(ser2) ] # https://pandas.pydata.org/pandas-docs/stable/referen
ce/api/pandas.Series.isin.html
```

Out[85]:

```
0    1
1    2
2    3
dtype: int64
```

In [89]:

```
#Difficiulty Level: L3

#Extract the valid emails from the series emails. The regex pattern for valid em
ails is provided as reference.

emails = pd.Series(['buying books at amazom.com', 'rameses@egypt.com', 'matt@t.c
o', 'narendra@modi.com'])
pattern ='[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,4}'
```

In [92]:

```
# Google "Pandas regex" ---> https://kanoki.org/2019/11/12/how-to-use-regex-in-p
andas/

import re

emails.str.findall(pattern, flags=re.IGNORECASE)
```

Out[92]:

```
0                      []
1    [rameses@egypt.com]
2            [matt@t.co]
3    [narendra@modi.com]
dtype: object
```

In [103]:

```
# Difficiulty Level: L3

# Get the positions of peaks (values surrounded by smaller values on both sides)
in ser.

ser = pd.Series([2, 10, 5, 4, 9, 10, 2, 7, 3])
# can apply the same logic we used earlier on NumPy arrays and lists earlier usi
ng loops
# I perfer easy to read code if the time-complexity doesnot change much.
```

In [104]:

```
# https://docs.scipy.org/doc/numpy-1.10.0/reference/generated/numpy.diff.html
# array-like: series also works
np.diff(ser)
```

Out[104]:

```
array([ 8, -5, -1,  5,  1, -8,  5, -4])
```

In [105]:

```
np.sign(np.diff(ser))
```

Out[105]:

```
array([ 1, -1, -1,  1,  1, -1,  1, -1])
```

In [106]:

```
np.diff(np.sign(np.diff(ser)))
```

Out[106]:

```
array([-2,  0,  2,  0, -2,  2, -2])
```

In [101]:

```
peak_locs = np.where(dd == -2)[0] + 1
```

In [102]:

```
print(peak_locs)
```

```
[1 5 7]
```

In [117]:

```
# Difficulty Level: L2

#From ser, keep the top 2 most frequent
#items as it is and replace everything else as 'Other'.

np.random.RandomState(100)
ser = pd.Series(np.random.randint(1, 5, [12]))
print(ser)
```

```
0     3
1     3
2     2
3     1
4     2
5     4
6     4
7     4
8     4
9     2
10    1
11    1
dtype: int64
```

In [118]:

```
print(ser.value_counts())
print(type(ser.value_counts()))
```

```
4    4
2    3
1    3
3    2
dtype: int64
<class 'pandas.core.series.Series'>
```

In [122]:

```
print(ser.value_counts().index)
#https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.index.html
```

```
Int64Index([4, 2, 1, 3], dtype='int64')
```

In [124]:

```
print(ser.value_counts().index[:2])
```

```
Int64Index([4, 2], dtype='int64')
```

In [125]:

```
ser[~ser.isin(ser.value_counts().index[:2])] = 'Other'
```

In [126]:

```
print(ser)
```

```
0     Other
1     Other
2         2
3     Other
4         2
5         4
6         4
7         4
8         4
9         2
10    Other
11    Other
dtype: object
```

Next Session: EDA [Seaborn]

We are planning to do one or more sessions on every chapter/concept in our course with focus on code-walkthroughs + code-FAQ + common-coding mistakes.

We will add all these sessions to the course stucture in appropriate places so that you can revisit them whenever you want.

In [ ]:

```

```