

Pandas & Matplotlib



Agenda

- Read data file
- Data selection
- Grouping
- Group function



Read Data File

- import pandas as pd
- pd.read_table("File Path", names=[...], index_col=..., header = ... ,sep=...)

```
1 help(pd.read_table)
```

Help on function read_table in module pandas.io.parsers:

```
read_table(filepath_or_buffer, sep='\t', delimiter=None, header='infer', names=None, index_col=None, usecols=None, squeeze=False, prefix=None, mangle_dupe_cols=True, dtype=None, engine=None, converters=None, true_values=None, false_values=None, skipinitialspace=False, skiprows=None, nrows=None, na_values=None, keep_default_na=True, na_filter=True, verbose=False, skip_blank_lines=True, parse_dates=False, infer_datetime_format=False, keep_date_col=False, date_parser=None, dayfirst=False, iterator=False, chunksize=None, compression='infer', thousands=None, decimal=b'.', lineterminator=None, quotechar='"', quoting=0, escapechar=None, comment=None, encoding=None, dialect=None, tupleize_cols=None, error_bad_lines=True, warn_bad_lines=True, skipfooter=0, doublequote=True, delim_whitespace=False, low_memory=True, memory_map=False, float_precision=None)
```



Read Data File - File Path

- `C:\...\Desktop\lab8\test.ipynb`
 - `C:\...\Desktop\lab8\Dataset\xxx.txt`
 - `C:\...\Desktop\lab8\Dataset\yyy.csv`
 - `C:\...\Desktop\lab8\zzz.csv`
 - `pd.read_table("File Path")`
-
- `"Dataset/xxx.txt"` or `"./Dataset/xxx.txt"`
 - `"Dataset/yyy.csv"` or `"./Dataset/yyy.csv"`
 - `"zzz.csv"` or `"./zzz.csv"`



Read Data File - Header

- The default value of header is “infer”
- Here is the description:

header : int or list of ints, default 'infer' Row number(s) to use as the column names, and the start of the data.

Default behavior is to infer the column names:

if no names are passed the behavior is identical to ``header=0`` and column names are inferred from the first line of the file,
if column names are passed explicitly then the behavior is identical to ``header=None``.

Explicitly pass ``header=0`` to be able to replace existing names. The header can be a list of integers that specify row locations for a multi-index on the columns e.g. [0,1,3]. Intervening rows that are not specified will be skipped (e.g. 2 in this example is skipped). Note that this parameter ignores commented lines and empty lines if ``skip_blank_lines=True``, so header=0 denotes the first line of data rather than the first line of the file.

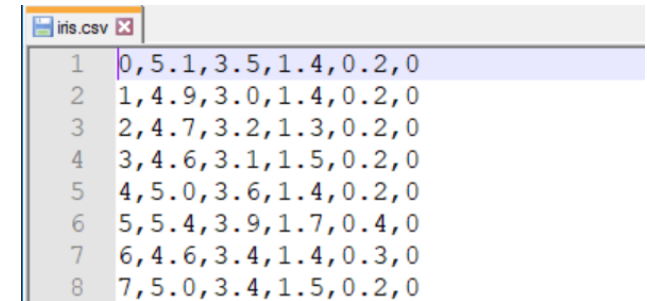
country,year,population
Afghanistan,1952,8425333
Afghanistan,1957,9240934
Afghanistan,1962,10267083
Afghanistan,1967,11537966
Afghanistan,1972,13079460
Afghanistan,1977,14880372
Afghanistan,1982,12881816
Afghanistan,1987,13867957

0,5.1,3.5,1.4,0.2,0
1,4.9,3.0,1.4,0.2,0
2,4.7,3.2,1.3,0.2,0
3,4.6,3.1,1.5,0.2,0
4,5.0,3.6,1.4,0.2,0
5,5.4,3.9,1.7,0.4,0
6,4.6,3.4,1.4,0.3,0
7,5.0,3.4,1.5,0.2,0



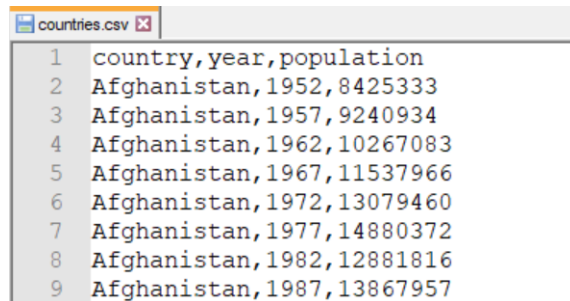
Read Data File - Names

- `pd.read_table("iris.csv", header = None, names=`
 `["col1", "col2 ",...])`



1	0,5.1,3.5,1.4,0.2,0
2	1,4.9,3.0,1.4,0.2,0
3	2,4.7,3.2,1.3,0.2,0
4	3,4.6,3.1,1.5,0.2,0
5	4,5.0,3.6,1.4,0.2,0
6	5,5.4,3.9,1.7,0.4,0
7	6,4.6,3.4,1.4,0.3,0
8	7,5.0,3.4,1.5,0.2,0

What if the dataset includes column title and you want to rename it?



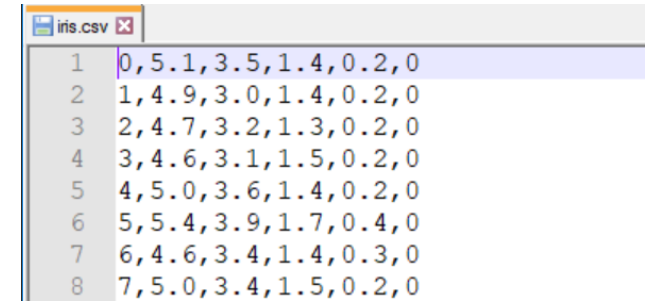
1	country,year,population
2	Afghanistan,1952,8425333
3	Afghanistan,1957,9240934
4	Afghanistan,1962,10267083
5	Afghanistan,1967,11537966
6	Afghanistan,1972,13079460
7	Afghanistan,1977,14880372
8	Afghanistan,1982,12881816
9	Afghanistan,1987,13867957



Read Data File - index_col

- `pd.read_table("iris.csv", header = None, names= ["index", "col2 ",...])`

	index	sepal_length	sepal_width	petal_length	petal_width	target_names
0	0	5.1	3.5	1.4	0.2	0
1	1	4.9	3.0	1.4	0.2	0
2	2	4.7	3.2	1.3	0.2	0
-	-	-	-	-	-	-



1	0, 5.1, 3.5, 1.4, 0.2, 0
2	1, 4.9, 3.0, 1.4, 0.2, 0
3	2, 4.7, 3.2, 1.3, 0.2, 0
4	3, 4.6, 3.1, 1.5, 0.2, 0
5	4, 5.0, 3.6, 1.4, 0.2, 0
6	5, 5.4, 3.9, 1.7, 0.4, 0
7	6, 4.6, 3.4, 1.4, 0.3, 0
8	7, 5.0, 3.4, 1.5, 0.2, 0

- Try this:
- `pd.read_table("iris.csv", header = None, names= ["index", "col2 ",...], index_col = "index")`



Try yourself 1

- Read the provided data file
- Assign the column names:
"index", "sepal_length", "sepal_width", "petal_length", "petal_width", "target_names"



Data Selection

- `df[5:8]`

1	data[5:8]				
	sepal_length	sepal_width	petal_length	petal_width	target_names
index					
5	5.4	3.9	1.7	0.4	0
6	4.6	3.4	1.4	0.3	0
7	5.0	3.4	1.5	0.2	0

- `df[5:8][["sepal_length", "sepal_width"]]`

	sepal_length	sepal_width
index		
5	5.4	3.9
6	4.6	3.4
7	5.0	3.4

Dataframe



Data Selection

- `df[5:8][["sepal_length"]]`

1	<code>data[5:8][["sepal_length"]]</code>	
	sepal_length	
	index	
	5	5.4
	6	4.6
	7	5.0

Dataframe

Dataframe

- `df[5:8]["sepal_length"]`

1	<code>data[5:8]["sepal_length"]</code>
---	--

index

5 5.4

6 4.6

7 5.0

Name: sepal_length, dtype: float64

Series



Data Selection - iloc / loc

- `df.iloc[4:8][["col1", "col2"]]`

```
1 data.iloc[4:8][["petal_length", "petal_width"]]
```

	petal_length	petal_width
index		
4	1.4	0.2
5	1.7	0.4
6	1.4	0.3
7	1.5	0.2

- `df.loc[4:8, ["col1", "col2"]]`

```
1 data.loc[4:8, ["petal_length", "petal_width"]]
```

	petal_length	petal_width
index		
4	1.4	0.2
5	1.7	0.4
6	1.4	0.3
7	1.5	0.2
8	1.4	0.2



Data Selection - Filtering

- `df[df["sepal_length"] > 5]`

```
1 data[ data["sepal_length"] > 5 ]
```

	sepal_length	sepal_width	petal_length	petal_width	target_names
index					
0	5.1	3.5	1.4	0.2	0
5	5.4	3.9	1.7	0.4	0
10	5.4	3.7	1.5	0.2	0
14	5.8	4.0	1.2	0.2	0
15	5.7	4.4	1.5	0.4	0
16	5.4	3.9	1.3	0.4	0
17	5.1	3.5	1.4	0.3	0

```
1 data["sepal_length"] > 5
```

```
index
0      True
1     False
2     False
3     False
4     False
5      True
6     False
7     False
8     False
9     False
10     True
```



Try yourself 2

- 1. Display the dataframe where index 51 to 100.
- 2. Display the data from index 51- 100,
and `sepal_length > 6`

	sepal_length	sepal_width	petal_length	petal_width	target_names
index					
51	6.4	3.2	4.5	1.5	1
52	6.9	3.1	4.9	1.5	1
54	6.5	2.8	4.6	1.5	1
56	6.3	3.3	4.7	1.6	1
58	6.6	2.9	4.6	1.3	1
63	6.1	2.9	4.7	1.4	1
65	6.7	3.1	4.4	1.4	1
68	6.2	2.2	4.5	1.5	1
71	6.1	2.8	4.0	1.3	1
72	6.3	2.5	4.9	1.5	1
73	6.1	2.8	4.7	1.2	1
74	6.4	2.9	4.3	1.3	1
75	6.6	3.0	4.4	1.4	1
76	6.8	2.8	4.8	1.4	1
77	6.7	3.0	5.0	1.7	1
86	6.7	3.1	4.7	1.5	1
87	6.3	2.3	4.4	1.3	1
91	6.1	3.0	4.6	1.4	1
97	6.2	2.9	4.3	1.3	1



Data Selection - Multiple Condition

- The important part is the **round bracket**

```
3 data[(data["sepal_width"]>3.2) & (data["sepal_width"] < 3.5)]
```

	sepal_length	sepal_width	petal_length	petal_width	target_names
index					
6	4.6	3.4	1.4	0.3	0
7	5.0	3.4	1.5	0.2	0
11	4.8	3.4	1.6	0.2	0
20	5.4	3.4	1.7	0.2	0
23	5.1	3.3	1.7	0.5	0
24	4.8	3.4	1.9	0.2	0
26	5.0	3.4	1.6	0.4	0



Grouping

- <https://pandas.pydata.org/pandas-docs/stable/api.html#groupby>

```
1 data.groupby("target_names").mean()
```

	sepal_length	sepal_width	petal_length	petal_width
target_names				
0	5.006	3.418	1.464	0.244
1	5.936	2.770	4.260	1.326
2	6.588	2.974	5.552	2.026



Try yourself 3

- 1. Display the dataframe where:
 "sepal_width" > 3.2; and
 "petal_length" > 5
- 2. Display the dataframe contain only sepal_width where for each group (target_names) Hint: get_group(X)

