

# Pandas

- Open source, BSD-licensed library for high-performance, easy-to-use data structures and data analysis tools for Python
- Manipulates and visualizes data in spreadsheet


# Reading in Data From CSV File

I have the following data saved in the file “Grades\_Short.csv”:

F30									
	A	B	C	D	E	F	G	H	I
1	Name	Previous_Par	Participation	Mini_Exam1	Mini_Exam2	Participation	Mini_Exam3	Final	Grade
2	Jake	32	1	19.5	20	1	10	33	A
3	Joe	32	1	20	16	1	14	32	A
4	Susan	30	1	19	19	1	10.5	33	A-
5	Sol	31	1	22	13	1	13	34	A
6	Chris	30	1	19	17	1	12.5	33.5	A
7	Tarik	31	1	19	19	1	8	24	B
8	Malik	31.5	1	20	21	1	9	36	A
9									
10									

Let's see how we read this data into pandas:

**Before you use pandas you must import it. Anytime you use pandas put this line as the top of your code.**



```
import pandas as pd
```

```
df_grades = pd.read_csv("Grades_Short.csv")
```

# Pandas – read\_csv

```
pd.read_csv?
```

**Signature:** `pd.read_csv(filepath_or_buffer, sep=',', 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, skip_footer=0, doublequote=True, delim_whitespace=False, as_recarray=None, compact_ints=None, use_unsigned=None, low_memory=True, buffer_lines=None, memory_map=False, float_precision=None)`

**Docstring:**

Read CSV (comma-separated) file into DataFrame

Also supports optionally iterating or breaking of the file into chunks.

Additional help can be found in the `online docs for IO Tools`\_ <<http://pandas.pydata.org/pandas-docs/stable/io.html>>`\_.

**Parameters**

-----

**filepath\_or\_buffer** : str, pathlib.Path, py.\_path.local.LocalPath or any object with a read() method (such as a file handle or StringIO)

The string could be a URL. Valid URL schemes include http, ftp, s3, and file. For file URLs, a host is expected. For instance, a local file could be file://localhost/path/to/table.csv

**sep** : str, default ','

Delimiter to use. If sep is None, the C engine cannot automatically detect the separator, but the Python parsing engine can, meaning the latter will be used and automatically detect the separator by Python's builtin sniffer tool, ``csv.Sniffer``. In addition, separators longer than 1 character and different from ``'\s+'`` will be interpreted as regular expressions and will also force the use of the Python parsing engine. Note that regex delimiters are prone to ignoring quoted data. Regex example: ``'\r\t'``

**delimiter** : str, default ``None``

Always specify the input name (order of inputs only matters if you don't)

# Reading in Data From Excel

So, what is df\_grades and how does it store the data?

```
import pandas as pd

df_grades = pd.read_csv("Grades_Short.csv")
df_grades
```

	Name	Previous_Part	Participation1	Mini_Exam1	Mini_Exam2	Participation2	Mini_Exam3	Final	Grade
0	Jake	32.0	1	19.5	20	1	10.0	33.0	A
1	Joe	32.0	1	20.0	16	1	14.0	32.0	A
2	Susan	30.0	1	19.0	19	1	10.5	33.0	A-
3	Sol	31.0	1	22.0	13	1	13.0	34.0	A
4	Chris	30.0	1	19.0	17	1	12.5	33.5	A
5	Tarik	31.0	1	19.0	19	1	8.0	24.0	B
6	Malik	31.5	1	20.0	21	1	9.0	36.0	A

- df\_grades is a pandas **dataframe**.
- The data is stored in a tabular format very similar to excel.

# The head() Method

Using the **head()** method

```
import pandas as pd

df_grades = pd.read_csv("Grades_Short.csv")
df_grades.head(3)
```

	Name	Previous_Part	Participation1	Mini_Exam1	Mini_Exam2	Participation2	Mini_Exam3	Final	Grade
0	Jake	32.0	1	19.5	20	1	10.0	33.0	A
1	Joe	32.0	1	20.0	16	1	14.0	32.0	A
2	Susan	30.0	1	19.0	19	1	10.5	33.0	A-

- If the data is really large you don't want to print out the entire dataframe to your output.
- The **head(n)** method outputs the first n rows of the data frame. If n is not supplied, the default is the first 5 rows.
- Run the head() method after reading in the dataframe to check that everything is read in correctly.
- There is also a **tail(n)** method that returns the last n rows of the dataframe

# Basic Features

```
import pandas as pd
```

```
df_grades = pd.read_csv("Grades_Short.csv")  
df_grades.head(3)
```

	Name	Previous_Part	Participation1	Mini_Exam1	Mini_Exam2	Participation2	Mini_Exam3	Final	Grade
0	Jake	32.0	1	19.5	20	1	10.0	33.0	A
1	Joe	32.0	1	20.0	16	1	14.0	32.0	A
2	Susan	30.0	1	19.0	19	1	10.5	33.0	A-

```
#dimension of df  
df_grades.shape
```

```
(7, 9)
```

```
#How each column is stored  
df_grades.dtypes
```

```
Name           object  
Previous_Part  float64  
Participation1 int64  
Mini_Exam1     float64  
Mini_Exam2     int64  
Participation2 int64  
Mini_Exam3     float64  
Final          float64  
Grade          object  
dtype: object
```

# Basic Features

column names



	Name	Previous_Part	Participation1	Mini_Exam1	Mini_Exam2	Participation2	Mini_Exam3	Final	Grade
0	Jake	32.0	1	19.5	20	1	10.0	33.0	A
1	Joe	32.0	1	20.0	16	1	14.0	32.0	A
2	Susan	30.0	1	19.0	19	1	10.5	33.0	A-

row names = index

```
#Get column names  
df_grades.columns
```

```
Index(['Name', 'Previous_Part', 'Participation1', 'Mini_Exam1', 'Mini_Exam2',  
      'Participation2', 'Mini_Exam3', 'Final', 'Grade'],  
      dtype='object')
```

```
#Get row names  
df_grades.index
```

```
RangeIndex(start=0, stop=7, step=1)
```

---

# Selecting a Single Column

	Name	Previous_Part	Participation1	Mini_Exam1	Mini_Exam2	Participation2	Mini_Exam3	Final	Grade
0	Jake	32.0	1	19.5	20	1	10.0	33.0	A
1	Joe	32.0	1	20.0	16	1	14.0	32.0	A
2	Susan	30.0	1	19.0	19	1	10.5	33.0	A-
3	Sol	31.0	1	22.0	13	1	13.0	34.0	A
4	Chris	30.0	1	19.0	17	1	12.5	33.5	A
5	Tarik	31.0	1	19.0	19	1	8.0	24.0	B
6	Malik	31.5	1	20.0	21	1	9.0	36.0	A

```
#Get Name column  
df_grades[ 'Name' ]
```

```
0    Jake  
1     Joe  
2   Susan  
3     Sol  
4   Chris  
5   Tarik  
6   Malik  
Name: Name, dtype: object
```

- Between square brackets, the column must be given as a string
- Outputs column as a series
  - A series is a one-dimensional dataframe



# Selecting a Single Column

	Name	Previous_Part	Participation1	Mini_Exam1	Mini_Exam2	Participation2	Mini_Exam3	Final	Grade
0	Jake	32.0	1	19.5	20	1	10.0	33.0	A
1	Joe	32.0	1	20.0	16	1	14.0	32.0	A
2	Susan	30.0	1	19.0	19	1	10.5	33.0	A-
3	Sol	31.0	1	22.0	13	1	13.0	34.0	A
4	Chris	30.0	1	19.0	17	1	12.5	33.5	A
5	Tarik	31.0	1	19.0	19	1	8.0	24.0	B
6	Malik	31.5	1	20.0	21	1	9.0	36.0	A

```
#Get Name column  
df_grades.Name
```

```
0    Jake  
1    Joe  
2   Susan  
3    Sol  
4   Chris  
5   Tarik  
6   Malik  
Name: Name, dtype: object
```

- Exactly equivalent way to get Name column
  - + : don't have to type brackets or quotes
  - -: won't generalize to selecting multiple columns,, won't work if column names have spaces, can't create new columns this way

# Selecting Multiple Columns

	Name	Previous_Part	Participation1	Mini_Exam1	Mini_Exam2	Participation2	Mini_Exam3	Final	Grade
0	Jake	32.0	1	19.5	20	1	10.0	33.0	A
1	Joe	32.0	1	20.0	16	1	14.0	32.0	A
2	Susan	30.0	1	19.0	19	1	10.5	33.0	A-
3	Sol	31.0	1	22.0	13	1	13.0	34.0	A
4	Chris	30.0	1	19.0	17	1	12.5	33.5	A
5	Tarik	31.0	1	19.0	19	1	8.0	24.0	B
6	Malik	31.5	1	20.0	21	1	9.0	36.0	A

```
#Select multiple columns  
df_grades[["Name", "Grade"]]
```

	Name	Grade
0	Jake	A
1	Joe	A
2	Susan	A-
3	Sol	A
4	Chris	A
5	Tarik	B
6	Malik	A

- List of strings, which correspond to column names.
- You can select as many column as you want.
- Columns don't have to be contiguous.

# Storing Result

```
#Print the column  
df_grades[ "Name" ]
```

```
0    Jake  
1     Joe  
2   Susan  
3     Sol  
4   Chris  
5   Tarik  
6   Malik  
Name: Name, dtype: object
```

```
#Store the column  
names= df_grades[ "Name" ]  
names
```

```
0    Jake  
1     Joe  
2   Susan  
3     Sol  
4   Chris  
5   Tarik  
6   Malik  
Name: Name, dtype: object
```

The variable name stores a series

Why store a slice?

- We might want/have to do our analysis in steps.
  - Less error prone
  - More readable

# Slicing a Series

Slice/index through the index, which is usually numbers



```
names= df_grades[ "Name" ]  
names
```

```
0    Jake  
1     Joe  
2   Susan  
3     Sol  
4   Chris  
5   Tarik  
6   Malik  
Name: Name, dtype: object
```

Picking out single element

```
names[0]
```

```
'Jake'
```

Contiguous slice

```
names[1:4]
```

```
1     Joe  
2   Susan  
3     Sol  
Name: Name, dtype: object
```

Arbitrary slice

```
names[[1,2,4]]
```

```
1     Joe  
2   Susan  
4   Chris  
Name: Name, dtype: object
```

# Slicing a Data Frame

	Name	Previous_Part	Participation1	Mini_Exam1	Mini_Exam2	Participation2	Mini_Exam3	Final	Grade
0	Jake	32.0	1	19.5	20	1	10.0	33.0	A
1	Joe	32.0	1	20.0	16	1	14.0	32.0	A
2	Susan	30.0	1	19.0	19	1	10.5	33.0	A-
3	Sol	31.0	1	22.0	13	1	13.0	34.0	A
4	Chris	30.0	1	19.0	17	1	12.5	33.5	A
5	Tarik	31.0	1	19.0	19	1	8.0	24.0	B
6	Malik	31.5	1	20.0	21	1	9.0	36.0	A

- Pick a single value out.

Index label  
(number)

Column name  
(string)

```
first_name = df_grades.loc[0, "Name"]  
first_name
```

'Jake'

# Slicing a Data Frame

	Name	Previous_Part	Participation1	Mini_Exam1	Mini_Exam2	Participation2	Mini_Exam3	Final	Grade
0	Jake	32.0	1	19.5	20	1	10.0	33.0	A
1	Joe	32.0	1	20.0	16	1	14.0	32.0	A
2	Susan	30.0	1	19.0	19	1	10.5	33.0	A-
3	Sol	31.0	1	22.0	13	1	13.0	34.0	A
4	Chris	30.0	1	19.0	17	1	12.5	33.5	A
5	Tarik	31.0	1	19.0	19	1	8.0	24.0	B
6	Malik	31.5	1	20.0	21	1	9.0	36.0	A

- Pick out entire row:

```
first_row = df_grades.loc[0,:]
first_row
```

“pick out all  
columns”

```
Name      Jake
Previous_Part      32
Participation1      1
Mini_Exam1      19.5
Mini_Exam2      20
Participation2      1
Mini_Exam3      10
Final      33
Grade      A
Name: 0, dtype: object
```

first\_row is a series

# Slicing a Data Frame

	Name	Previous_Part	Participation1	Mini_Exam1	Mini_Exam2	Participation2	Mini_Exam3	Final	Grade
0	Jake	32.0	1	19.5	20	1	10.0	33.0	A
1	Joe	32.0	1	20.0	16	1	14.0	32.0	A
2	Susan	30.0	1	19.0	19	1	10.5	33.0	A-
3	Sol	31.0	1	22.0	13	1	13.0	34.0	A
4	Chris	30.0	1	19.0	17	1	12.5	33.5	A
5	Tarik	31.0	1	19.0	19	1	8.0	24.0	B
6	Malik	31.5	1	20.0	21	1	9.0	36.0	A

- Pick out contiguous chunk:

Endpoints are inclusive!

```
slice_one = df_grades.loc[0:2, "Name": "Mini_Exam2"]  
slice_one
```

	Name	Previous_Part	Participation1	Mini_Exam1	Mini_Exam2
0	Jake	32.0	1	19.5	20
1	Joe	32.0	1	20.0	16
2	Susan	30.0	1	19.0	19

# Slicing a Data Frame

	Name	Previous_Part	Participation1	Mini_Exam1	Mini_Exam2	Participation2	Mini_Exam3	Final	Grade
0	Jake	32.0	1	19.5	20	1	10.0	33.0	A
1	Joe	32.0	1	20.0	16	1	14.0	32.0	A
2	Susan	30.0	1	19.0	19	1	10.5	33.0	A-
3	Sol	31.0	1	22.0	13	1	13.0	34.0	A
4	Chris	30.0	1	19.0	17	1	12.5	33.5	A
5	Tarik	31.0	1	19.0	19	1	8.0	24.0	B
6	Malik	31.5	1	20.0	21	1	9.0	36.0	A

- Pick out arbitrary chunk:

```
slice_two = df_grades.loc[[0,2,3], ["Name", "Grade"]]  
slice_two
```

	Name	Grade
0	Jake	A
2	Susan	A-
3	Sol	A