

(MSc) Data Mining

Topic 01 : Module Overview

Part 13 : Top X pandas commands

Dr Bernard Butler and Dr Kieran Murphy

Department of Computing and Mathematics, SETU Waterford.
(bernard.butler@setu.ie; kmurphy@wit.ie)

Spring Semester, 2025

Outline

- Reading data formats
- Computing descriptive statistics
- Processing data by filtering and grouping

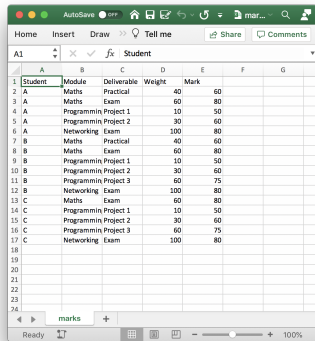
Part I

Introduction

Minimal Dataset

To better understand the various pandas operations we are going to use a tiny* dataset based on (fictional) student results. (marks.csv)

In Excel ...



| Student | Module | Deliverable | Weight | Mark |
|---------|-------------|-------------|--------|------|
| A | Maths | Practical | 40 | 60 |
| A | Maths | Exam | 60 | 80 |
| A | Programming | Project 1 | 10 | 50 |
| A | Programming | Project 2 | 30 | 60 |
| A | Networking | Exam | 100 | 80 |
| B | Maths | Practical | 40 | 60 |
| B | Maths | Exam | 60 | 80 |
| B | Programming | Project 1 | 10 | 50 |
| B | Programming | Project 2 | 30 | 60 |
| B | Programming | Project 3 | 60 | 75 |
| B | Networking | Exam | 100 | 80 |
| C | Maths | Exam | 60 | 80 |
| C | Programming | Project 1 | 10 | 50 |
| C | Programming | Project 2 | 30 | 60 |
| C | Programming | Project 3 | 60 | 75 |
| C | Networking | Exam | 100 | 80 |

...or database schema ...

Students

Name ... Other fields

Modules

Name ... Other fields

Deliverables

Name Weight ... Other fields

Grades

Student Module Deliverable Mark

...like to know ...

- Student performance — weighted mark on each module, missing deliverables etc.
- Module performance — number of attempts and average mark.
- Deliverable performance — number of attempts and average mark, predictor of overall module grade, etc.

*Dataset is small enough that you can verify operation results by hand.

Terminology

```
df.head(1000)
```

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

- A **DataFrame** is a table of data values.
 - `df = pd.read_csv("Marks.csv")`
- A **Series** is a list of data values — typically columns in a dataframe. We can access an individual column using
 - `df.Deliverable` (dot notation)
 - `df["Deliverable"]` (dict notation)
 - `df.iloc[:,2]` (numpy, index notation)
- The **index** is a special column whose values can be used to access rows — rather using row number.
 - The default index is equal to the row number.

Terminology

```
df.head(1000)
```

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

- A **DataFrame** is a table of data values.
 - `df = pd.read_csv("Marks.csv")`
- A **Series** is a list of data values — typically columns in a dataframe. We can access an individual column using
 - `df.Deliverable` (dot notation)
 - `df["Deliverable"]` (dict notation)
 - `df.iloc[:,2]` (numpy, index notation)
- The **index** is a special column whose values can be used to access rows — rather using row number.
 - The default index is equal to the row number.

Terminology

```
df.head(1000)
```

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

- A **DataFrame** is a table of data values.
 - `df = pd.read_csv("Marks.csv")`
- A **Series** is a list of data values — typically columns in a dataframe. We can access an individual column using
 - `df.Deliverable` (dot notation)
 - `df["Deliverable"]` (dict notation)
 - `df.iloc[:,2]` (numpy, index notation)
- The **index** is a special column whose values can be used to access rows — rather using row number.
 - The default index is equal to the row number.

Part II

Input and Output

Setup

Minimal

We begin every data mining project with importing the three core data science packages:

```
1 import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('seaborn-darkgrid')
```

numpy — fast array operations
pandas — data manipulation
matplotlib — visualisation

- We give modules nicknames (`np`, `pd`, ...) to simplify their later use, and we access properties/functions of a package using the dot notation (`np.max`, `pd.DataFrame`, ...).

Extra

```
2 import seaborn as sns
import statsmodels.api as sm

pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
```

seaborn — statistical visualisation
statsmodels — statistical data exploration
pandas options to show all columns for wider datasets

Setup

Minimal

We begin every data mining project with importing the three core data science packages:

```
3 import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('seaborn-darkgrid')
```

numpy — fast array operations
pandas — data manipulation
matplotlib — visualisation

- We give modules nicknames (`np`, `pd`, ...) to simplify their later use, and we access properties/functions of a package using the dot notation (`np.max`, `pd.DataFrame`, ...).

Extra

```
4 import seaborn as sns
import statsmodels.api as sm

pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
```

seaborn — statistical visualisation
statsmodels — statistical data exploration
pandas options to show all columns for wider datasets

Reading data from a CSV file

Pandas supports a huge variety of input/output formats so best approach is to focus on what is needed to process the given data and verify input. Our marks dataset is in CSV format so we start with

```
5 import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('seaborn-darkgrid')
```

and input using

```
6 df = pd.read_csv('Marks.csv', sep=',')
print(df.shape)
df.head()
```

Reading data from a CSV file

Pandas supports a huge variety of input/output formats so best approach is to focus on what is needed to process the given data and verify input. Our marks dataset is in CSV format so we start with

```
7 import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('seaborn-darkgrid')
```

and input using

```
8 df = pd.read_csv('Marks.csv', sep=',')
print(df.shape)
df.head()
```

Reading data from a CSV file

Pandas supports a huge variety of input/output formats so best approach is to focus on what is needed to process the given data and verify input. Our marks dataset is in CSV format so we start with

```
9 • import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('seaborn-darkgrid')
```

and input using

```
10 • df = pd.read_csv('Marks.csv', sep=',')
print(df.shape)
df.head()
```

(16, 5)

(16, 5)

```
[2]:
```

| | Student | Module | Deliverable | Weight | Mark |
|---|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |

Always verify input by checking dataset dimensions and looking at some rows!!!

Datatypes

Pandas data types:

- **object** — used for text or mixed numeric and non-numeric values.
- **int64** — integer values,
 - Does not support missing values, so an int column containing at least one missing value will automatically be converted to float.
- **float64** — floating point numbers.
- **bool** — **True/False** values
- **datetime64** — date and time values
- **category** — Finite (typically small) list of text values

```
Student      object
Module       object
Deliverable  object
Weight       int64
Mark         int64
dtype: object
```

```
11 df.dtypes
```

Regularly verifying datatypes is vital[†] :

- Operations differ based on datatype, eg, '+' concatenate strings but adds numerical values.
- Datatype can change based on results, eg, int converts to float due to missing values.

[†]Google "Detecting Excel's gene auto-conversions."

Datatypes

Pandas data types:

- **object** — used for text or mixed numeric and non-numeric values.
- **int64** — integer values,
 - Does not support missing values, so an int column containing at least one missing value will automatically be converted to float.
- **float64** — floating point numbers.
- **bool** — **True/False** values
- **datetime64** — date and time values
- **category** — Finite (typically small) list of text values

| | |
|-------------|---------------|
| Student | object |
| Module | object |
| Deliverable | object |
| Weight | int64 |
| Mark | int64 |
| dtype: | object |

12 ●

`df.dtypes`

Regularly verifying datatypes is vital[†] :

- Operations differ based on datatype, eg, '+' concatenate strings but adds numerical values.
- Datatype can change based on results, eg, int converts to float due to missing values.

[†]Google "Detecting Excel's gene auto-conversions."

Datatypes

Pandas data types:

- **object** — used for text or mixed numeric and non-numeric values.
- **int64** — integer values,
 - Does not support missing values, so an int column containing at least one missing value will automatically be converted to float.
- **float64** — floating point numbers.
- **bool** — **True/False** values
- **datetime64** — date and time values
- **category** — Finite (typically small) list of text values

| | |
|-------------|---------------|
| Student | object |
| Module | object |
| Deliverable | object |
| Weight | int64 |
| Mark | int64 |
| dtype: | object |

13

df.dtypes

Regularly verifying datatypes is vital[†] :

- Operations differ based on datatype, eg, '+' concatenate strings but adds numerical values.
- Datatype can change based on results, eg, int converts to float due to missing values.

[†]Google "Detecting Excel's gene auto-conversions."

Datatypes — Converting

We will deal with modifying and creating new columns later, but while we are on datatypes, we will look at changing datatype ...

Using the Series function `astype`

```
14 df["Weight"] = df["Weight"].astype('float')  
df["Weight"].dtype
```

`dtype('float64')`

- New datatype is required argument — 'int', 'float', 'str', 'object', 'category', etc.
- Simple, but fragile if data conversion is possible.

or using pandas function `to_numeric`

```
15 df["Weight"] = pd.to_numeric(df["Weight"])  
df["Weight"].dtype
```

`dtype('float64')`

- More powerful, can specify what to do in cases where the conversion fails etc
- Have functions `to_numeric`, `to_datetime`, and `to_timedelta`.

Datatypes — Converting

We will deal with modifying and creating new columns later, but while we are on datatypes, we will look at changing datatype ...

Using the Series function `astype`

```
16 df["Weight"] = df["Weight"].astype('float')  
df["Weight"].dtype
```

`dtype('float64')`

- New datatype is required argument — `'int'`, `'float'`, `'str'`, `'object'`, `'category'`, etc.
- Simple, but fragile if data conversion is possible.

or using pandas function `to_numeric`

```
17 df["Weight"] = pd.to_numeric(df["Weight"])  
df["Weight"].dtype
```

`dtype('float64')`

- More powerful, can specify what to do in cases where the conversion fails etc
- Have functions `to_numeric`, `to_datetime`, and `to_timedelta`.

Datatypes — Converting

We will deal with modifying and creating new columns later, but while we are on datatypes, we will look at changing datatype ...

Using the Series function `astype`

```
18 df["Weight"] = df["Weight"].astype('float')  
df["Weight"].dtype
```

`dtype('float64')`

- New datatype is required argument — `'int'`, `'float'`, `'str'`, `'object'`, `'category'`, etc.
- Simple, but fragile if data conversion is possible.

or using pandas function `to_numeric`

```
19 df["Weight"] = pd.to_numeric(df["Weight"])  
df["Weight"].dtype
```

`dtype('float64')`

- More powerful, can specify what to do in cases where the conversion fails etc
- Have functions `to_numeric`, `to_datetime`, and `to_timedelta`.

Missing Values

Identifying and dealing with missing values is critical step in data preparation. What should you do? delete rows containing missing values? or impute then?

Here we will just look at identifying missing values.

20 ● `df.isna()`

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|--------|-------------|--------|-------|
| 0 | False | False | False | False | False |
| 1 | False | False | False | False | False |
| 2 | False | False | False | False | False |
| 3 | False | False | False | False | False |
| 4 | False | False | False | False | False |
| 5 | False | False | False | False | False |
| 6 | False | False | False | False | False |
| 7 | False | False | False | False | False |
| 8 | False | False | False | False | False |
| 9 | False | False | False | False | False |
| 10 | False | False | False | False | False |
| 11 | False | False | False | False | False |
| 12 | False | False | False | False | False |
| 13 | False | False | False | False | False |
| 14 | False | False | False | False | False |
| 15 | False | False | False | False | False |

21 ● `df.isna().sum()`

| | |
|--------------|---|
| Student | 0 |
| Module | 0 |
| Deliverable | 0 |
| Weight | 0 |
| Mark | 0 |
| dtype: int64 | |

22 ● `df.isna().sum().sum()`

0

- Use dataframe function `fillna` to replace missing values.
- Recall **False** and **True** map to 0 and 1 respectively.
- Use `df.isna().sum(axis=1)` to sum along rows.

Output

Saving dataframe to CSV is straightforward (I rarely include the (default) index when saving datasets).

```
23 df.to_csv('marks_2.csv', index=False)
```

- CSV has become the default file format in Data Mining application especially for 'informal' datasets.
 - ✓ human readable, easy to generate / parse (if correct).
 - ✗ Can be highly redundant, slow to input/output.
 - ✗ No meta information.
- Other formats are better for speed and resulting file size and for saving meta data not supported by CSV (such as columns datatypes, category information, etc).

towards
data science

: The Best Format to Save Pandas Data

Part III

Filtering

Selecting individual rows/columns results in a series

Columns can be accessed using dot, dict and numpy index notation.

```
df.head(1000)
```

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

26 ● `df.iloc[:,0]`

25 ● `df['Student']`

24 ● `df.Student`

```
Student      A
Module      Maths
Deliverable  Practical
Weight      40
Mark        60
Name: 0, dtype: object
```

27 ● `df.iloc[0]`



```
0      A
1      A
2      A
3      A
4      A
5      B
6      B
7      B
8      B
9      B
10     B
11     C
12     C
13     C
14     C
15     C
Name: Student, dtype: object
```

← Access row using numpy index

Selecting individual rows/columns results in a series

Columns can be accessed using dot, dict and numpy index notation.

```
df.head(1000)
```

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

df.iloc[:,0]

df['Student']

df.Student

```
Student      A
Module      Maths
Deliverable  Practical
Weight      40
Mark        60
Name: 0, dtype: object
```

df.iloc[0]

Access row using numpy index

```
0    A
1    A
2    A
3    A
4    A
5    B
6    B
7    B
8    B
9    B
10   B
11   C
12   C
13   C
14   C
15   C
Name: Student, dtype: object
```

Selecting individual rows/columns results in a series

Columns can be accessed using dot, dict and numpy index notation.

```
df.head(1000)
```

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

df.iloc[:,0]

df['Student']

df.Student

```
Student      A
Module      Maths
Deliverable  Practical
Weight      40
Mark        60
Name: 0, dtype: object
```

df.iloc[0]

Access row using numpy index

```
0    A
1    A
2    A
3    A
4    A
5    B
6    B
7    B
8    B
9    B
10   B
11   C
12   C
13   C
14   C
15   C
Name: Student, dtype: object
```


Head and Tail

Commands `head` and `tail` return the first and last n rows (default $n = 5$) of a dataframe/series.

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

36 ● `df.head()`

37 ● `df.head(1)`

38 ● `df.tail()`

39 ● `df.tail()`

Head and Tail

Commands `head` and `tail` return the first and last n rows (default $n = 5$) of a dataframe/series.

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

40 ● `df.head()`

| | Student | Module | Deliverable | Weight | Mark |
|---|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |

42 ● `df.tail()`

41 ● `df.head(1)`

| | Student | Module | Deliverable | Weight | Mark |
|---|---------|--------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |

43 ● `df.tail()`

Head and Tail

Commands `head` and `tail` return the first and last n rows (default $n = 5$) of a dataframe/series.

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

44 `df.head()`

| | Student | Module | Deliverable | Weight | Mark |
|---|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |

46 `df.tail()`

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

45 `df.head(1)`

| | Student | Module | Deliverable | Weight | Mark |
|---|---------|--------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |

47 `df.tail()`

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|------------|-------------|--------|------|
| 15 | C | Networking | Lab Work | 100 | 80 |

Query — on a single-column criteria

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

48

```
df.query("Student=='A'")
```

49

```
df.query("Mark>=70")
```

Query — on a single-column criteria

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

50

```
df.query("Student=='A'")
```

| | Student | Module | Deliverable | Weight | Mark |
|---|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |

51

```
df.query("Mark>=70")
```

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

Query — on a single-column criteria

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

52

`df.query("Student=='A'")`

| Student | | Deliverable | | Weight | Mark |
|---------|---|-------------|-----------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

53

`df.query("Mark>=70")`

| Student | Module | Deliverable | Weight | Mark |
|---------|--------|-------------|-----------|------|
| 0 | A | Maths | Practical | 40 |
| 1 | A | Maths | Exam | 60 |
| 2 | A | Programming | Project 1 | 10 |
| 3 | A | Programming | Project 2 | 30 |
| 4 | A | Networking | Lab Work | 100 |
| 5 | B | Maths | Practical | 40 |
| 6 | B | Maths | Exam | 60 |
| 7 | B | Programming | Project 1 | 10 |
| 8 | B | Programming | Project 2 | 30 |
| 9 | B | Programming | Project 3 | 60 |
| 10 | B | Networking | Project | 100 |
| 11 | C | Maths | Exam | 60 |
| 12 | C | Programming | Project 1 | 10 |
| 13 | C | Programming | Project 2 | 30 |
| 14 | C | Programming | Project 3 | 60 |
| 15 | C | Networking | Lab Work | 100 |

| | Student | Module | Deliverable | Weight | Mark |
|---|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 1 | A | Maths | Exam | 60 | 80 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 6 | B | Maths | Exam | 60 | 80 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

Query — on multiple columns (using python logical operators)

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

54

```
df.query("Mark<70 and Module=='Maths'")
```

55

```
df.query("Mark<70 or Module=='Maths'")
```

Query — on multiple columns (using python logical operators)

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

56

```
df.query("Mark<70 and Module=='Maths'")
```

57

```
df.query("Mark<70 or Module=='Maths'")
```


Query — on multiple columns (using python logical operators)

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

| Student | Module | Deliverable | Weight | Mark |
|---------|-------------|-------------|--------|------|
| 0 A | Maths | Practical | 40 | 60 |
| 1 A | Maths | Exam | 60 | 80 |
| 2 A | Programming | Project 1 | 10 | 50 |
| 3 A | Programming | Project 2 | 30 | 60 |
| 4 A | Networking | Lab Work | 100 | 80 |
| 5 B | Maths | Practical | 40 | 60 |
| 6 B | Maths | Exam | 60 | 80 |
| 7 B | Programming | Project 1 | 10 | 50 |
| 8 B | Programming | Project 2 | 30 | 60 |
| 9 B | Programming | Project 3 | 60 | 75 |
| 10 B | Networking | Project | 100 | 80 |
| 11 C | Maths | Exam | 60 | 80 |
| 12 C | Programming | Project 1 | 10 | 50 |
| 13 C | Programming | Project 2 | 30 | 60 |
| 14 C | Programming | Project 3 | 60 | 75 |
| 15 C | Networking | Lab Work | 100 | 80 |

| Student | Module | Deliverable | Weight | Mark |
|---------|-------------|-------------|--------|------|
| 0 A | Maths | Practical | 40 | 60 |
| 1 A | Maths | Exam | 60 | 80 |
| 2 A | Programming | Project 1 | 10 | 50 |
| 3 A | Programming | Project 2 | 30 | 60 |
| 4 A | Networking | Lab Work | 100 | 80 |
| 5 B | Maths | Practical | 40 | 60 |
| 6 B | Maths | Exam | 60 | 80 |
| 7 B | Programming | Project 1 | 10 | 50 |
| 8 B | Programming | Project 2 | 30 | 60 |
| 9 B | Programming | Project 3 | 60 | 75 |
| 10 B | Networking | Project | 100 | 80 |
| 11 C | Maths | Exam | 60 | 80 |
| 12 C | Programming | Project 1 | 10 | 50 |
| 13 C | Programming | Project 2 | 30 | 60 |
| 14 C | Programming | Project 3 | 60 | 75 |
| 15 C | Networking | Lab Work | 100 | 80 |

58

```
df.query("Mark<70 and Module=='Maths'")
```

| | Student | Module | Deliverable | Weight | Mark |
|---|---------|--------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 5 | B | Maths | Practical | 40 | 60 |

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |

59

```
df.query("Mark<70 or Module=='Maths'")
```

Query — on multiple columns (using pandas logical operators)

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

| Student Module | | Deliverable Weight Mark | | |
|----------------|---|-------------------------|-----------|--------|
| 0 | A | Maths | Practical | 40 60 |
| 1 | A | Maths | Exam | 60 80 |
| 2 | A | Programming | Project 1 | 10 50 |
| 3 | A | Programming | Project 2 | 30 60 |
| 4 | A | Networking | Lab Work | 100 80 |
| 5 | B | Maths | Practical | 40 60 |
| 6 | B | Maths | Exam | 60 80 |
| 7 | B | Programming | Project 1 | 10 50 |
| 8 | B | Programming | Project 2 | 30 60 |
| 9 | B | Programming | Project 3 | 60 75 |
| 10 | B | Networking | Project | 100 80 |
| 11 | C | Maths | Exam | 60 80 |
| 12 | C | Programming | Project 1 | 10 50 |
| 13 | C | Programming | Project 2 | 30 60 |
| 14 | C | Programming | Project 3 | 60 75 |
| 15 | C | Networking | Lab Work | 100 80 |

| Student Module | | Deliverable Weight | | Mark |
|----------------|---|--------------------|-----------|------|
| 0 | A | Maths | Practical | 40 |
| 1 | A | Maths | Exam | 60 |
| 2 | A | Programming | Project 1 | 10 |
| 3 | A | Programming | Project 2 | 30 |
| 4 | A | Networking | Lab Work | 100 |
| 5 | B | Maths | Practical | 40 |
| 6 | B | Maths | Exam | 60 |
| 7 | B | Programming | Project 1 | 10 |
| 8 | B | Programming | Project 2 | 30 |
| 9 | B | Programming | Project 3 | 60 |
| 10 | B | Networking | Project | 100 |
| 11 | C | Maths | Exam | 60 |
| 12 | C | Programming | Project 1 | 10 |
| 13 | C | Programming | Project 2 | 30 |
| 14 | C | Programming | Project 3 | 60 |
| 15 | C | Networking | Lab Work | 100 |

60

```
df.query("(Mark<70) & (Module=='Maths'))")
```

| | Student | Module | Deliverable | Weight | Mark |
|---|---------|--------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 5 | B | Maths | Practical | 40 | 60 |

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |

61

```
df.query("(Mark<70) | (Module=='Maths'))")
```

Filtering using `loc`

Note the square (not round) brackets — think of `loc` as array indexing not a function call.

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

```
df.loc[ROW_SELECTION, COL_SELECTION]
```

where row and columns selection can be

- Single values: row number or column name
- An integer list for rows or list of column names
- A boolean list for logical indexing of rows
- A colon to indicate every row/column

62

```
df.loc[df.Module=="Maths", ["Student", "Mark"]]
```

Filtering using `loc`

Note the square (not round) brackets — think of `loc` as array indexing not a function call.

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

`df.loc[ROW_SELECTION, COL_SELECTION]`

where row and columns selection can be

- Single values: row number or column name
- An integer list for rows or list of column names
- A boolean list for logical indexing of rows
- A colon to indicate every row/column

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

63 `df.loc[df.Module=="Maths", ["Student", "Mark"]]`

Filtering using loc

Note the square (not round) brackets — think of loc as array indexing not a function call.

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

`df.loc[ROW_SELECTION, COL_SELECTION]`

where row and columns selection can be

- Single values: row number or column name
- An integer list for rows or list of column names
- A boolean list for logical indexing of rows
- A colon to indicate every row/column

64

```
df.loc[df.Module=="Maths", ["Student", "Mark"]]
```

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

Student Mark

| | | |
|----|---|----|
| 0 | A | 60 |
| 1 | A | 80 |
| 5 | B | 60 |
| 6 | B | 80 |
| 11 | C | 80 |

More complicated example

I prefer to define row selection criteria, and the column list and order, separately to the `loc` statement.

| | Student | Module | Deliverable | Weight | Mark |
|---|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |

```
criteria = ((df.Mark<50) & (df.Module=='Maths')) | ((df.Mark<70) & (df.Module!='Maths'))
columns = ['Module', 'Student', 'Mark']

df.loc[criteria, columns]
```

More complicated example

I prefer to define row selection criteria, and the column list and order, separately to the `loc` statement.

| | Student Module | | Deliverable Weight | | Mark |
|---|----------------|-------------|--------------------|-----|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |

| | Student Module | | Deliverable Weight | | Mark |
|----|----------------|-------------|--------------------|-----|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

```
criteria = ((df.Mark<50) & (df.Module=='Maths')) | ((df.Mark<70) & (df.Module!='Maths'))
columns = ['Module', 'Student', 'Mark']
```

```
df.loc[criteria, columns]
```

More complicated example

I prefer to define row selection criteria, and the column list and order, separately to the `loc` statement.

| | Student Module | | Deliverable Weight | | Mark |
|---|----------------|-------------|--------------------|-----|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |

| | Student Module | | Deliverable Weight | | Mark |
|----|----------------|-------------|--------------------|-----|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

| | Student Module | | Deliverable Weight | | Mark |
|----|----------------|-------------|--------------------|-----|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |

```
criteria = ((df.Mark<50) & (df.Module=='Maths')) | ((df.Mark<70) & (df.Module!='Maths'))
columns = ['Module', 'Student', 'Mark']
```

```
df.loc[criteria, columns]
```


Sampling

The `sample` function selects a random subset of the dataframe rows.

- Either specify the number of rows (as an integer) or fraction of the data (as a float).
- Can set the seed using `random_state` parameter for reproducible samples.

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 14 | C | Programming | Project 3 | 60 | 75 |
| 15 | C | Networking | Lab Work | 100 | 80 |

`df.sample(n=3)`

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 11 | C | Maths | Exam | 60 | 80 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 8 | B | Programming | Project 2 | 30 | 60 |

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 13 | C | Programming | Project 2 | 30 | 60 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 12 | C | Programming | Project 1 | 10 | 50 |
| 5 | B | Maths | Practical | 40 | 60 |
| 2 | A | Programming | Project 1 | 10 | 50 |

Part IV

Sorting

Sorting

A pandas dataframe has two sorting operations:

- `sort_index()` orders rows based on current index.
- `sort_values(COLUMNS)` orders rows based on single column or list of columns.

Two important modifications:

- By default, the sort order is in ascending. Set parameter `ascending=False` to reverse this.
- By default, a new dataframe is returned with desired sort order, set parameter `inplace=True` to update current dataframe instead (then no output is generated).

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 10 | B | Networking | Project | 100 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 13 | C | Programming | Project 2 | 30 | 60 |

`df.sort_values(['Module', 'Deliverable'])`

Sorting

A pandas dataframe has two sorting operations:

- `sort_index()` orders rows based on current index.
- `sort_values(COLUMNS)` orders rows based on single column or list of columns.

Two important modifications:

- By default, the sort order is in ascending. Set parameter `ascending=False` to reverse this.
- By default, a new dataframe is returned with desired sort order, set parameter `inplace=True` to update current dataframe instead (then no output is generated).

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-----------------------|-------------|--------|------|
| 0 | A | Maths | Practical | 40 | 60 |
| 1 | A | Maths | Exam | 60 | 80 |
| 2 | A | Programming Project 1 | 10 | 50 | |
| 3 | A | Programming Project 2 | 30 | 60 | |
| 4 | A | Networking Lab Work | 100 | 80 | |
| 5 | B | Maths | Practical | 40 | 60 |
| 6 | B | Maths | Exam | 60 | 80 |
| 7 | B | Programming Project 1 | 10 | 50 | |
| 8 | B | Programming Project 2 | 30 | 60 | |
| 9 | B | Programming Project 3 | 60 | 75 | |
| 10 | B | Networking Project | 100 | 80 | |
| 11 | C | Maths | Exam | 60 | 80 |
| 12 | C | Programming Project 1 | 10 | 50 | |
| 13 | C | Programming Project 2 | 30 | 60 | |

`df.sort_values(['Module', 'Deliverable'])`

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 1 | A | Maths | Exam | 60 | 80 |
| 6 | B | Maths | Exam | 60 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| | A | Maths | Practical | 40 | 60 |
| | B | Maths | Practical | 40 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 15 | C | Networking | Lab Work | 100 | 80 |
| 10 | B | Networking | Project | 100 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 13 | C | Programming | Project 2 | 30 | 60 |

Part V

Defining New Columns

Defining new columns — row-wise operation

We want to compute the weighted mark for each module for each student. Two steps:

- Create column, `W_Mark`, to store the weighted mark for each deliverable. This is a row by row calculation — only need data in current row to compute the result.
- Create column, `M_Mark`, to store the module mark for each student. This is a group calculation — need all rows for that student and module to compute the result.

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 1 | A | Maths | Exam | 60 | 80 |
| 6 | B | Maths | Exam | 60 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 0 | A | Maths | Practical | 40 | 60 |
| 5 | B | Maths | Practical | 40 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 15 | C | Networking | Lab Work | 100 | 80 |
| 10 | B | Networking | Project | 100 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 14 | C | Programming | Project 3 | 60 | 75 |

```
df['W_Mark'] = df.Weight * df.Mark // 100
```

Need to use dict notation (not dot notation) when defining a new column.

Defining new columns — row-wise operation

We want to compute the weighted mark for each module for each student. Two steps:

- Create column, W_Mark, to store the weighted mark for each deliverable. This is a row by row calculation — only need data in current row to compute the result.
- Create column, M_Mark, to store the module mark for each student. This is a group calculation — need all rows for that student and module to compute the result.

| | Student | Module | Deliverable | Weight | Mark |
|----|---------|-------------|-------------|--------|------|
| 1 | A | Maths | Exam | 60 | 80 |
| 6 | B | Maths | Exam | 60 | 80 |
| 11 | C | Maths | Exam | 60 | 80 |
| 0 | A | Maths | Practical | 40 | 60 |
| 5 | B | Maths | Practical | 40 | 60 |
| 4 | A | Networking | Lab Work | 100 | 80 |
| 15 | C | Networking | Lab Work | 100 | 80 |
| 10 | B | Networking | Project | 100 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 |
| 7 | B | Programming | Project 1 | 10 | 50 |
| 12 | C | Programming | Project 1 | 10 | 50 |
| 3 | A | Programming | Project 2 | 30 | 60 |
| 8 | B | Programming | Project 2 | 30 | 60 |
| 13 | C | Programming | Project 2 | 30 | 60 |
| 9 | B | Programming | Project 3 | 60 | 75 |
| 14 | C | Programming | Project 3 | 60 | 75 |

`df['W_Mark'] = df.Weight * df.Mark // 100`

Need to use dict notation (not dot notation) when defining a new column.

| | Student | Module | Deliverable | Weight | Mark | W_Mark |
|----|---------|-------------|-------------|--------|------|--------|
| 1 | A | Maths | Exam | 60 | 80 | 48 |
| 6 | B | Maths | Exam | 60 | 80 | 48 |
| 11 | C | Maths | Exam | 60 | 80 | 48 |
| 0 | A | Maths | Practical | 40 | 60 | 24 |
| 5 | B | Maths | Practical | 40 | 60 | 24 |
| 4 | A | Networking | Lab Work | 100 | 80 | 80 |
| 15 | C | Networking | Lab Work | 100 | 80 | 80 |
| 10 | B | Networking | Project | 100 | 80 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 | 5 |
| 7 | B | Programming | Project 1 | 10 | 50 | 5 |
| 12 | C | Programming | Project 1 | 10 | 50 | 5 |
| 3 | A | Programming | Project 2 | 30 | 60 | 18 |
| 8 | B | Programming | Project 2 | 30 | 60 | 18 |
| 13 | C | Programming | Project 2 | 30 | 60 | 18 |
| 9 | B | Programming | Project 3 | 60 | 75 | 45 |
| 14 | C | Programming | Project 3 | 60 | 75 | 45 |

Defining new columns — group aggregate result

I

- Create column, W_Mark, to store the weighted mark for each deliverable. This is a row by row calculation — only need data in current row to compute the result.
- Create column, M_Mark, to store the module mark for each student. This is a group calculation — need all rows for that student and module to compute the result.

73 ● columns to group on output cols aggregate
`df.groupby(['Student', 'Module'])['W_Mark'].sum()`

| | Student | Module | Deliverable | Weight | Mark | W_Mark |
|----|---------|-------------|-------------|--------|------|--------|
| 1 | A | Maths | Exam | 60 | 80 | 48 |
| 6 | B | Maths | Exam | 60 | 80 | 48 |
| 11 | C | Maths | Exam | 60 | 80 | 48 |
| 0 | A | Maths | Practical | 40 | 60 | 24 |
| 5 | B | Maths | Practical | 40 | 60 | 24 |
| 4 | A | Networking | Lab Work | 100 | 80 | 80 |
| 15 | C | Networking | Lab Work | 100 | 80 | 80 |
| 10 | B | Networking | Project | 100 | 80 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 | 5 |
| 7 | B | Programming | Project 1 | 10 | 50 | 5 |
| 12 | C | Programming | Project 1 | 10 | 50 | 5 |
| 3 | A | Programming | Project 2 | 30 | 60 | 18 |
| 8 | B | Programming | Project 2 | 30 | 60 | 18 |
| 13 | C | Programming | Project 2 | 30 | 60 | 18 |
| 9 | B | Programming | Project 3 | 60 | 75 | 45 |
| 14 | C | Programming | Project 3 | 60 | 75 | 45 |

Result has multi-level index, need to use `reset_index` to revert to default index

Defining new columns — group aggregate result

I

- Create column, W_Mark, to store the weighted mark for each deliverable. This is a row by row calculation — only need data in current row to compute the result.
- Create column, M_Mark, to store the module mark for each student. This is a group calculation — need all rows for that student and module to compute the result.

74 ● columns to group on output cols aggregate
`df.groupby(['Student', 'Module'])['W_Mark'].sum()`

| | Student | Module | Deliverable | Weight | Mark | W_Mark |
|----|---------|-------------|-------------|--------|------|--------|
| 1 | A | Maths | Exam | 60 | 80 | 48 |
| 6 | B | Maths | Exam | 60 | 80 | 48 |
| 11 | C | Maths | Exam | 60 | 80 | 48 |
| 0 | A | Maths | Practical | 40 | 60 | 24 |
| 5 | B | Maths | Practical | 40 | 60 | 24 |
| 4 | A | Networking | Lab Work | 100 | 80 | 80 |
| 15 | C | Networking | Lab Work | 100 | 80 | 80 |
| 10 | B | Networking | Project | 100 | 80 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 | 5 |
| 7 | B | Programming | Project 1 | 10 | 50 | 5 |
| 12 | C | Programming | Project 1 | 10 | 50 | 5 |
| 3 | A | Programming | Project 2 | 30 | 60 | 18 |
| 8 | B | Programming | Project 2 | 30 | 60 | 18 |
| 13 | C | Programming | Project 2 | 30 | 60 | 18 |
| 9 | B | Programming | Project 3 | 60 | 75 | 45 |
| 14 | C | Programming | Project 3 | 60 | 75 | 45 |

Result has multi-level index, need to use `reset_index` to revert to default index

| | | W_Mark |
|----------------|-------------|--------|
| Student Module | | |
| A | Maths | 72 |
| | Networking | 80 |
| | Programming | 23 |
| B | Maths | 72 |
| | Networking | 80 |
| | Programming | 68 |
| C | Maths | 48 |
| | Networking | 80 |
| | Programming | 68 |

Defining new columns — group aggregate result

II

75

```
df.groupby(['Student', 'Module'])[['W_Mark']].sum().reset_index()
```

| | Student | Module | Deliverable | Weight | Mark | W_Mark |
|----|---------|-------------|-------------|--------|------|--------|
| 1 | A | Maths | Exam | 60 | 80 | 48 |
| 6 | B | Maths | Exam | 60 | 80 | 48 |
| 11 | C | Maths | Exam | 60 | 80 | 48 |
| 0 | A | Maths | Practical | 40 | 60 | 24 |
| 5 | B | Maths | Practical | 40 | 60 | 24 |
| 4 | A | Networking | Lab Work | 100 | 80 | 80 |
| 15 | C | Networking | Lab Work | 100 | 80 | 80 |
| 10 | B | Networking | Project | 100 | 80 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 | 5 |
| 7 | B | Programming | Project 1 | 10 | 50 | 5 |
| 12 | C | Programming | Project 1 | 10 | 50 | 5 |
| 3 | A | Programming | Project 2 | 30 | 60 | 18 |
| 8 | B | Programming | Project 2 | 30 | 60 | 18 |
| 13 | C | Programming | Project 2 | 30 | 60 | 18 |
| 9 | B | Programming | Project 3 | 60 | 75 | 45 |
| 14 | C | Programming | Project 3 | 60 | 75 | 45 |

Defining new columns — group aggregate result

II

76

```
df.groupby(['Student', 'Module'])[['W_Mark']].sum().reset_index()
```

| | Student | Module | Deliverable | Weight | Mark | W_Mark |
|----|---------|-------------|-------------|--------|------|--------|
| 1 | A | Maths | Exam | 60 | 80 | 48 |
| 6 | B | Maths | Exam | 60 | 80 | 48 |
| 11 | C | Maths | Exam | 60 | 80 | 48 |
| 0 | A | Maths | Practical | 40 | 60 | 24 |
| 5 | B | Maths | Practical | 40 | 60 | 24 |
| 4 | A | Networking | Lab Work | 100 | 80 | 80 |
| 15 | C | Networking | Lab Work | 100 | 80 | 80 |
| 10 | B | Networking | Project | 100 | 80 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 | 5 |
| 7 | B | Programming | Project 1 | 10 | 50 | 5 |
| 12 | C | Programming | Project 1 | 10 | 50 | 5 |
| 3 | A | Programming | Project 2 | 30 | 60 | 18 |
| 8 | B | Programming | Project 2 | 30 | 60 | 18 |
| 13 | C | Programming | Project 2 | 30 | 60 | 18 |
| 9 | B | Programming | Project 3 | 60 | 75 | 45 |
| 14 | C | Programming | Project 3 | 60 | 75 | 45 |

This is the required result and we can save this to a new dataframe. However, we often want to put this into our original dataframe as an extra column. Only problem we have is different rows so can't just assign to a new column — need to use transform function.

| | Student | Module | W_Mark |
|---|---------|-------------|--------|
| 0 | A | Maths | 72 |
| 1 | A | Networking | 80 |
| 2 | A | Programming | 23 |
| 3 | B | Maths | 72 |
| 4 | B | Networking | 80 |
| 5 | B | Programming | 68 |
| 6 | C | Maths | 48 |
| 7 | C | Networking | 80 |
| 8 | C | Programming | 68 |

Defining new columns — group aggregate result

III

77

columns to group on

output cols

aggregate

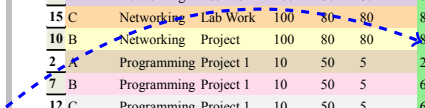
```
df['M_Mark'] = df.groupby(['Student', 'Module'])['W_Mark'].transform(sum)
```

| | Student | Module | Deliverable | Weight | Mark | W_Mark |
|----|---------|-------------|-------------|--------|------|--------|
| 1 | A | Maths | Exam | 60 | 80 | 48 |
| 6 | B | Maths | Exam | 60 | 80 | 48 |
| 11 | C | Maths | Exam | 60 | 80 | 48 |
| 0 | A | Maths | Practical | 40 | 60 | 24 |
| 5 | B | Maths | Practical | 40 | 60 | 24 |
| 4 | A | Networking | Lab Work | 100 | 80 | 80 |
| 15 | C | Networking | Lab Work | 100 | 80 | 80 |
| 10 | B | Networking | Project | 100 | 80 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 | 5 |
| 7 | B | Programming | Project 1 | 10 | 50 | 5 |
| 12 | C | Programming | Project 1 | 10 | 50 | 5 |
| 3 | A | Programming | Project 2 | 30 | 60 | 18 |
| 8 | B | Programming | Project 2 | 30 | 60 | 18 |
| 13 | C | Programming | Project 2 | 30 | 60 | 18 |
| 9 | B | Programming | Project 3 | 60 | 75 | 45 |
| 14 | C | Programming | Project 3 | 60 | 75 | 45 |



| | Student | Module | W_Mark |
|---|---------|-------------|--------|
| 0 | A | Maths | 72 |
| 1 | A | Networking | 80 |
| 2 | A | Programming | 23 |
| 3 | B | Maths | 72 |
| 4 | B | Networking | 80 |
| 5 | B | Programming | 68 |
| 6 | C | Maths | 48 |
| 7 | C | Networking | 80 |
| 8 | C | Programming | 68 |

| | Student | Module | Deliverable | Weight | Mark | W_Mark | M_Mark |
|----|---------|-------------|-------------|--------|------|--------|--------|
| 1 | A | Maths | Exam | 60 | 80 | 48 | 72 |
| 6 | B | Maths | Exam | 60 | 80 | 48 | 72 |
| 11 | C | Maths | Exam | 60 | 80 | 48 | 48 |
| 0 | A | Maths | Practical | 40 | 60 | 24 | 72 |
| 5 | B | Maths | Practical | 40 | 60 | 24 | 72 |
| 4 | A | Networking | Lab Work | 100 | 80 | 80 | 80 |
| 15 | C | Networking | Lab Work | 100 | 80 | 80 | 80 |
| 10 | B | Networking | Project | 100 | 80 | 80 | 80 |
| 2 | A | Programming | Project 1 | 10 | 50 | 5 | 23 |
| 7 | B | Programming | Project 1 | 10 | 50 | 5 | 68 |
| 12 | C | Programming | Project 1 | 10 | 50 | 5 | 68 |
| 3 | A | Programming | Project 2 | 30 | 60 | 18 | 23 |
| 8 | B | Programming | Project 2 | 30 | 60 | 18 | 68 |
| 13 | C | Programming | Project 2 | 30 | 60 | 18 | 68 |
| 9 | B | Programming | Project 3 | 60 | 75 | 45 | 68 |
| 14 | C | Programming | Project 3 | 60 | 75 | 45 | 68 |



The transform broadcasts the result for each group over every row in that group.

Part VI

Review Exercises

Review Exercises

Generate the following reports:

- 1 Number of deliverables by each student.
- 2 List and rank deliverables by grade.
- 3 Top 2 deliverables (by grade).
- 4 Top 2 module (by average grade).
- 5 Top 2 modules (by minimum grade).
- 6 Modules (by minimum grade).

(value_counts, or groupby and count)

(sort_values, rank)

Harder exercises (new functions)

- 1 List which students missed which deliverables.

(pivot, melt)