Python assumed knowledge

1. Data is held in structured data frames
2. Python is a programming language that can be used for data analysis
3. How to open a Jupyter notebook
4. How to use a Jupyter notebook to write, edit and run Python code

Manipulating dataset columns in Python

Learning intentions

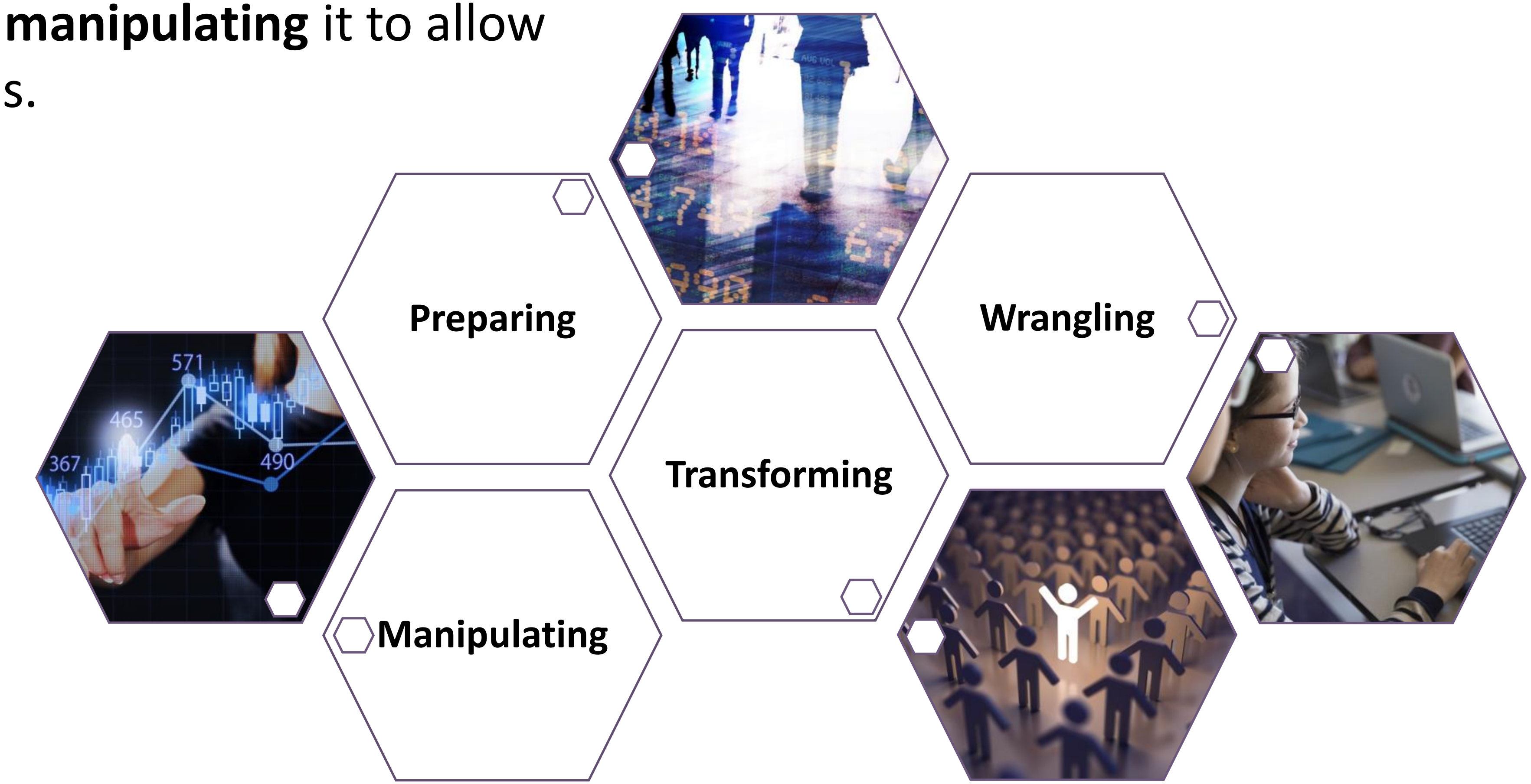
We will be learning how to manipulate data in Python, specifically to

- **select** columns
- **reorder** columns, and
- **reformat** columns

Background

When a data analyst is given a dataset to analyse, **most of their time is spend manipulating** it to allow them to conduct the analysis.

Many words are used to describe this process, but they all mean the same thing.



Preparing

Wrangling

Transforming

Manipulating

Why this is important?

Some benefits of manipulating data are,

Makes it easier for you to understand the dataset you're working with

Helps break the task into **manageable chunks**

Speeds up data processing

It **helps you to focus** on what you are planning to do with the data by choosing which columns you need to keep.

Show me…

Here is an example of a dataset that would need the columns to be manipulated.

There is a column that is **no longer needed**

It would be easier to understand if the **start_time** and **end_time** columns were **next to each other**

end_time	date	old_title	id	start_time
13:06	44467	0	1A	12:30
02:56	29/09/2021	0	2B	21:18
23:53	44,469	0	3C	14:09
16:04	44,470	0	4D	21:49

date column has a **mixture of formats**

Show me…

end_time	date	old_title	id	start_time
13:06	44467	0	1A	12:30
02:56	29/09/2021	0	2B	21:18
23:53	44,469	0	3C	14:09
16:04	44,470	0	4D	21:49

id	date	start_time	end_time
1A	28/9/2021	12:30	13:06
2B	29/09/2021	21:18	02:56
3C	30/9/2021	14:09	23:53
4D	1/10/2021	21:49	16:04

This lesson will show you **how to manipulate columns of data**, which will make any analysis easier to complete and less prone to errors

pandas

We will be using the Python **pandas** package for data manipulation in these lessons.

pandas is the most important and widely-used Python package for data manipulation.

Using pandas:

- makes it easier to work with datasets in Python
- provides us with a powerful set of data manipulation functions
- reduces the amount of code we need to write

Show me…

Pandas has powerful data manipulation functions. To use them, we first of all copy the data from the csv (comma separate values) file into a **pandas data frame**.

dataset (csv file)

```
river, start_point, length_km,
Tay, Ben Lui, 188,
Spey, Loch Spey, 172,
Clyde, Lowther Hill, 170,
Forth, Loch Ard, 47,
Nith, Enoch Hill, 114,
Tweed, Tweeds Well, 156
```

pandas data frame

river	**start_point**	**length_km**
Tay	Ben Lui	188
Spey	Loch Spey	172
Clyde	Lowther Hill	170
Forth	Loch Ard	47
Nith	Enoch Hill	114
Tweed	Tweeds Well	156

Definition

Select

To choose some of the columns from a dataset

Show me…

Here is an example of selecting columns.

address	price_2017	price_2018	price_2019	price_2020
5 Little Gardens	50,000	55,000	60,500	66,550
101 Cherry Tree Lane	65,000	71,500	78,650	86,515
42 Granny Clarks Wynd	100,000	110,000	121,000	133,100

Select **address** and **price_2020** columns

address	price_2020
5 Little Gardens	66,550
101 Cherry Tree Lane	86,515
42 Granny Clarks Wynd	133,100

Example

Select

To choose some of the columns from a dataset

In this example, we will select a *single* column from a dataset.

Select the name of the **loch** from this dataset.

loch	volume	length	max_depth
Loch Ness	7.45	36.2	227
Loch Lomond	2.6	36.0	190
Loch Morar	2.3	18.8	310
Loch Tay	1.6	23.0	150
Loch Awe	1.2	41.0	94

Select a single column in Python

Select the name of the **loch** from the **lochs** data frame.

lochs

loch	**volume**	**length**	**max_depth**
Loch Ness	7.45	36.2	227
Loch Lomond	2.6	36.0	190
Loch Morar	2.3	18.8	310
Loch Tay	1.6	23.0	150
Loch Awe	1.2	41.0	94

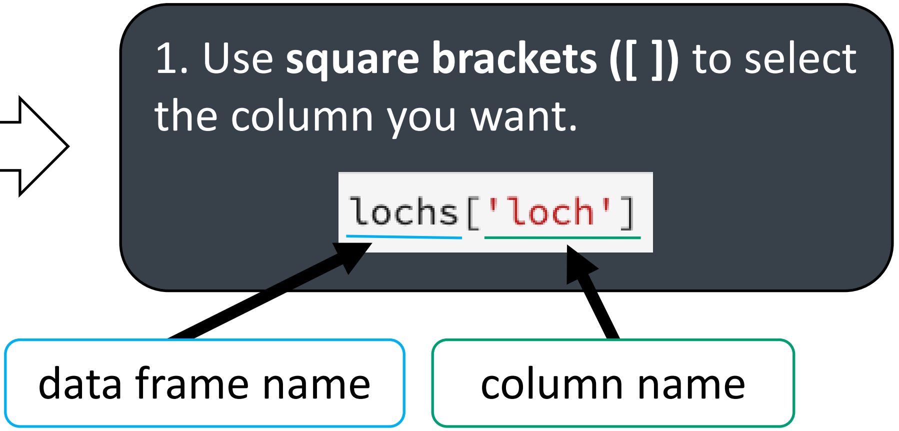
Select **loch**

loch
Loch Ness
Loch Lomond
Loch Morar
Loch Tay
Loch Awe

Select a single column in Python

lochs

loch	volume	length	max_depth
Loch Ness	7.45	36.2	227
Loch Lomond	2.6	36.0	190
Loch Morar	2.3	18.8	310
Loch Tay	1.6	23.0	150
Loch Awe	1.2	41.0	94



1. Use **square brackets ([ ])** to select the column you want.

```
lochs['loch']
```

loch
Loch Ness
Loch Lomond
Loch Morar
Loch Tay
Loch Awe

In pandas, columns are selected using square brackets (i.e. [ ]). In the example above, square brackets are used to select a *single* column but they can also be used to select *multiple* columns. We'll look at this next.

Example

Select

To choose some of the columns from a dataset

Now that we have selected a *single* column, we will select a *more than one* column from a dataset.

Select the name of the **loch** and its **length** from this dataset

loch	volume	length	max_depth
Loch Ness	7.45	36.2	227
Loch Lomond	2.6	36.0	190
Loch Morar	2.3	18.8	310
Loch Tay	1.6	23.0	150
Loch Awe	1.2	41.0	94

Select multiple columns in Python

1. Select the name of the **loch** and its **length** from the **lochs** data frame.
2. Assign these columns to a *new* data frame, **lochs_small**.

lochs

loch	volume	length	max_depth
Loch Ness	7.45	36.2	227
Loch Lomond	2.6	36.0	190
Loch Morar	2.3	18.8	310
Loch Tay	1.6	23.0	150
Loch Awe	1.2	41.0	94

Select **loch** and **length**

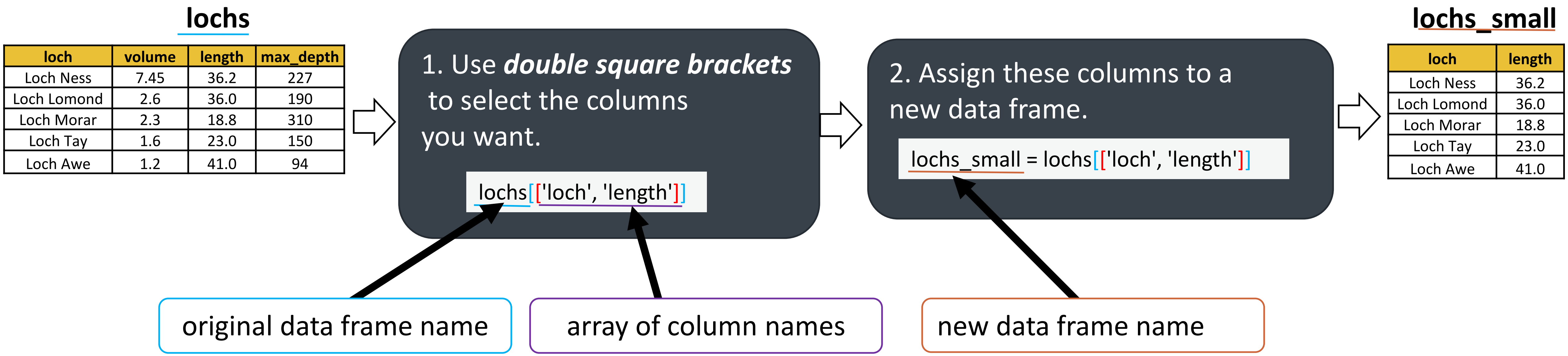
lochs_small

loch	length
Loch Ness	36.2
Loch Lomond	36.0
Loch Morar	18.8
Loch Tay	23.0
Loch Awe	41.0

Creating new data frames when manipulating data is good practice because:

- It means that you don't overwrite the original dataframe (which allows you to return to it later)
- it makes it easier for you to *see* and *reproduce* the steps you have taken in your analysis.

Select multiple columns in Python



lochs

loch	volume	length	max_depth
Loch Ness	7.45	36.2	227
Loch Lomond	2.6	36.0	190
Loch Morar	2.3	18.8	310
Loch Tay	1.6	23.0	150
Loch Awe	1.2	41.0	94

1. Use ***double square brackets*** to select the columns you want.

`lochs[['loch', 'length']]`

2. Assign these columns to a new data frame.

`lochs_small = lochs[['loch', 'length']]`

lochs_small

loch	length
Loch Ness	36.2
Loch Lomond	36.0
Loch Morar	18.8
Loch Tay	23.0
Loch Awe	41.0

Why *double square brackets?*

Python arrays are surrounded by square brackets.

The *inner* brackets ([ ]) in `lochs[['loch', 'length']]` are part of the Python *array* that contains the column names '**loch**' and '**length**' (**i**.e. ['loch', 'length']), and

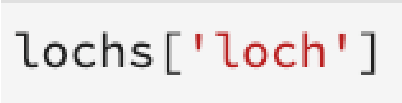
the *outer* brackets ([ ]) in `lochs[['loch', 'length']]` are for *selecting* the variables from the data frame **lochs**.

Select columns in Python

In summary:

Selecting 1 column

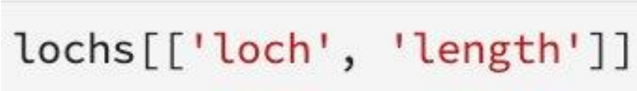
Use **single** square brackets



```
lochs['loch']
```

Selecting more than 1 column

Use **double** square brackets



```
lochs[['loch', 'length']]
```

Next steps

Complete the **Set Up** and **Select Columns** sections of the

‘Manipulating Columns in a Dataset' Jupyter Notebook.

Definition

Reorder

To change the order of columns in a dataset

Show me…

Here is an example of reordering columns.

A	B	C
1	Apples	Red
3	Bananas	Yellow
5	Grapes	Green

Reorder column
B to be 1st

B	A	C
Apples	1	Red
Bananas	3	Yellow
Grapes	5	Green

Example

Reorder

To change the order of columns in a dataset

Reorder this dataset so the columns are in the following order:
location, temperature, dawn, dusk

temperature	dusk	dawn	location
15	22:50	03:30	Edinburgh
21	22:28	05:09	Paris
17	17:21	06:25	Sydney
17	20:55	04:54	New York

Reorder in Python

1. Select the **location**, **temperature**, **dawn** and **dusk** from the **cities** data frame, in this order.
2. Assign these columns to a new data frame, **cities_small**.

cities

temperature	dusk	dawn	location
15	22:50	03:30	Edinburgh
21	22:28	05:09	Paris
17	17:21	06:25	Sydney
17	20:55	04:54	New York

Reorder

cities_new

location	temperature	dawn	dusk
Edinburgh	15	03:30	22:50
Paris	21	05:09	22:28
Sydney	17	06:25	17:21
New York	17	04:54	20:55

Reorder in Python

cities

temperature	dusk	dawn	location
15	22:50	03:30	Edinburgh
21	22:28	05:09	Paris
17	17:21	06:25	Sydney
17	20:55	04:54	New York

1. Use **double brackets** to select the columns you want, in the order you want.

```
cities[['location', 'temperature', 'dawn', 'dusk']]
```

original data frame name

column names in order

2. Assign these columns to a new data frame.

```
cities_new = cities[['location', 'temperature', 'dawn', 'dusk']]
```

new data frame name

cities_new

location	temperature	dawn	dusk
Edinburgh	15	03:30	22:50
Paris	21	05:09	22:28
Sydney	17	06:25	17:21
New York	17	04:54	20:55

Next steps

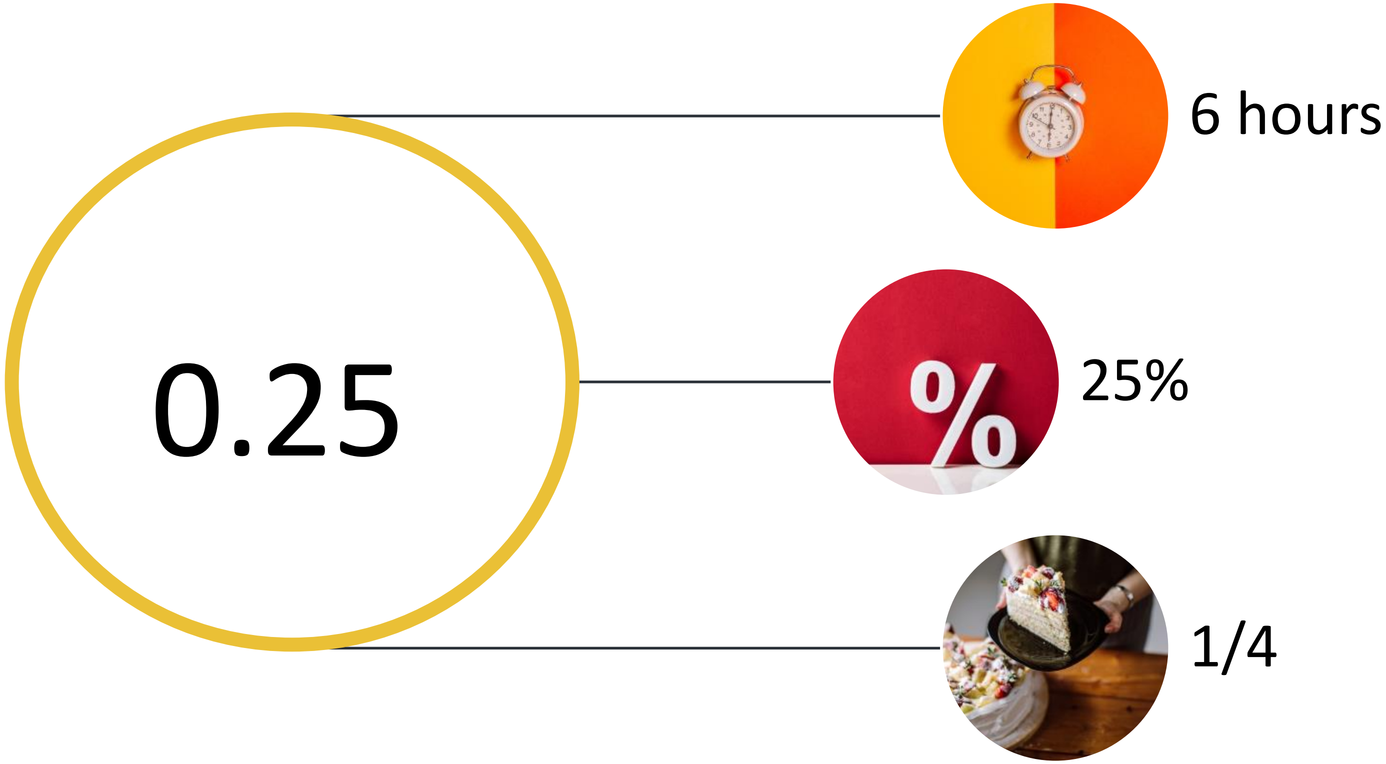
Complete the **Reorder Columns** section of the ‘Manipulating Columns in a Dataset' Jupyter Notebook.

Definition

Reformat

To change how data in a column is displayed

Show me…



A number can be reformatted to appear in different ways.

However, the underlying number stays the same.

Changing the display format can make the data easier to understand.

Show me…

Value
0.25
0.5
0.75
1

Value
25%
50%
75%
100%

Value
£ 0.25
£ 0.50
£ 0.75
£ 1.00

Value
1/4
1/2
3/4
1

All of these columns of data have the same stored values,
but they have been **reformatted to be displayed differently**.

Example

Reformat

To change how data in a column is displayed

Reformat the values in the **price** column so that the values are displayed in £s and have a comma after the *thousands e.g. £3,000.*

used_cars

make	model	price	rating
Audi	A1	12500	62
Ford	Fiesta	12000	48
Ford	Puma	22500	71
Vauxhall	Corsa	11995	74

Reformat in Python

1. Create a **format dictionary** to specify *which* columns to format, and *how* to format them.
2. Apply the format dictionary to the data frame using the **format()** function.

used_cars

make	model	price	rating
Audi	A1	12500	62
Ford	Fiesta	12000	48
Ford	Puma	22500	71
Vauxhall	Corsa	11995	74

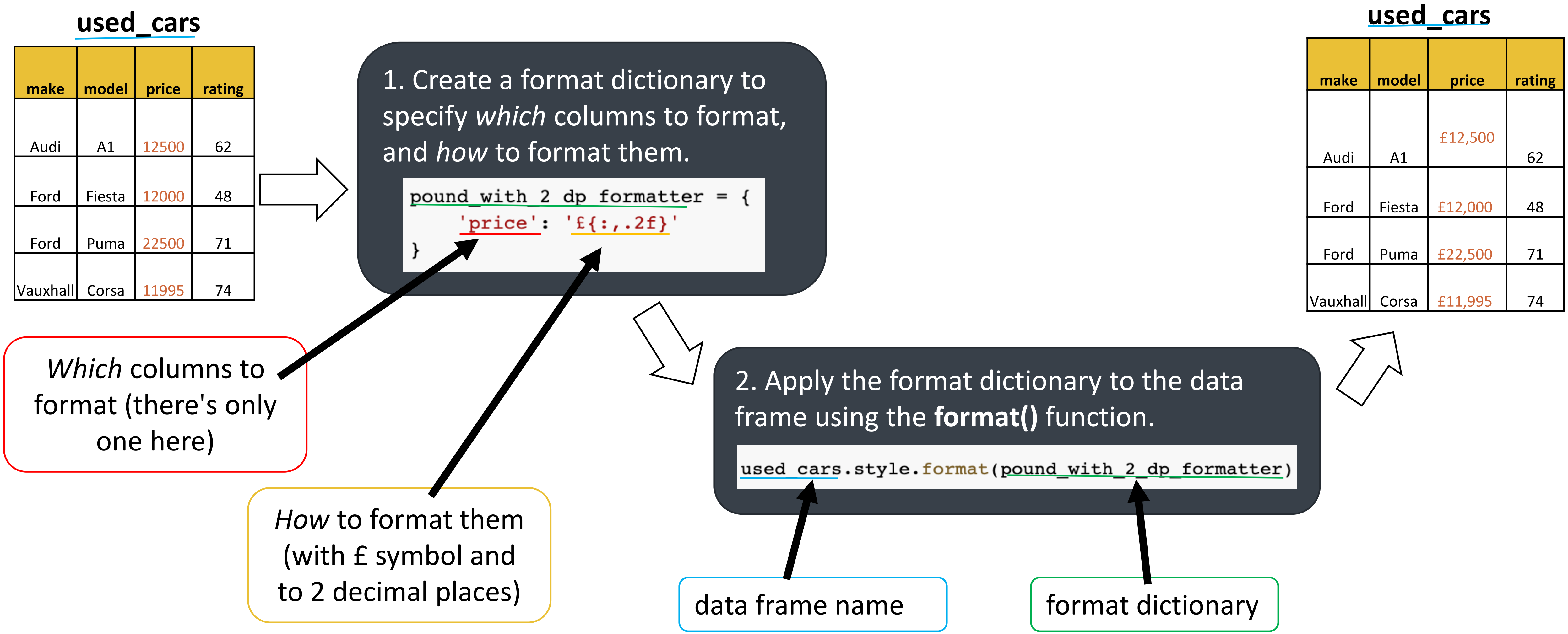
Reformat

used_cars

make	model	price	rating
Audi	A1	£12,500	62
Ford	Fiesta	£12,000	48
Ford	Puma	£22,500	71
Vauxhall	Corsa	£11,995	74

QUESTION: why did we not assign the reformatted data to a ***new*** *data frame?*

Reformat in Python



used_cars

make	model	price	rating
Audi	A1	12500	62
Ford	Fiesta	12000	48
Ford	Puma	22500	71
Vauxhall	Corsa	11995	74

1. Create a format dictionary to specify *which* columns to format, and *how* to format them.

```
pound_with_2_dp_formatter = {
    'price': '£{:,.2f}'
}
```

Which columns to format (there's only one here)

How to format them (with £ symbol and to 2 decimal places)

2. Apply the format dictionary to the data frame using the **format()** function.

```
used_cars.style.format(pound_with_2_dp_formatter)
```

data frame name

format dictionary

used_cars

make	model	price	rating
Audi	A1	£12,500	62
Ford	Fiesta	£12,000	48
Ford	Puma	£22,500	71
Vauxhall	Corsa	£11,995	74

Next steps

Complete the **Reformat Columns** section of the ‘Manipulating Columns in a Dataset' Jupyter Notebook.

Additional information

In this lesson we have covered some of the procedures for manipulating data in Python.

Some useful resources:

- [Official pandas documentation]

Learning checklist

I can *describe* what is means to select and reorder columns in a data frame.

I can *manipulate* data by selecting and reordering columns using Python.

How you can use this lesson

Created by Effini in partnership with Data Education in Schools, The Data Lab and Data Skills for Work, with funding from the Scottish Government.