

# Python assumed knowledge

1. Python programming to at least the level defined in SQA Computer Programming Level 5 (HY2C 45)
2. How to use a Jupyter notebook to write, edit and run Python code
3. Completion of the **Combining Datasets** lesson

# Practise combining datasets in Python

Version: 1.1



# Learning intentions

We will be learning about **how to combine datasets in Python**, specifically

- how to append rows to a dataset, and
- how to join columns to a dataset

# Background

In the “Combining datasets” lesson we looked at what we mean by appending rows and joining columns.

In this lesson we will use pandas to **practise appending** and **joining data in Python**.

*Appending rows*


*Joining columns*


# Appending data in Python

Here are two datasets containing CO<sub>2</sub> emissions for the UK and USA from 2015 to 2019. We will append the dataset on the right to the dataset on the left, to create a single dataset.

uk\_co2

	country_name	year	co2_amount_kiloton
0	United Kingdom	2015	401080
1	United Kingdom	2016	380810
2	United Kingdom	2017	367000
3	United Kingdom	2018	360730
4	United Kingdom	2019	348920

us\_co2

	country_name	year	co2_amount_kiloton
0	United States	2015	4990710
1	United States	2016	4894500
2	United States	2017	4819370
3	United States	2018	4975310
4	United States	2019	4817720



# Appending data using concat()

The pandas **concat()** function can be used to append one data frame to another. **concat()** is short for 'concatenate' which means to put things together as a connected series.

Step 1. Create a list of data frames to be concatenated.

```
co2_frames = [uk_co2, us_co2]
```

new list



uk\_co2

	country_name	year	co2_amount_kiloton
0	United Kingdom	2015	401080
1	United Kingdom	2016	380810
2	United Kingdom	2017	367000
3	United Kingdom	2018	360730
4	United Kingdom	2019	348920

us\_co2

	country_name	year	co2_amount_kiloton
0	United States	2015	4990710
1	United States	2016	4894500
2	United States	2017	4819370
3	United States	2018	4975310
4	United States	2019	4817720

# Appending data using concat()

Step 2. Create a new data frame by passing **concat()** the list.

This will append **us\_co2** to **uk\_co2**.

```
co2_uk_us = pd.concat(co2_frames)
```

new data frame

List of data frames

Reminder: pandas has been imported using

```
import pandas as pd
```

co2_uk_us			
	country_name	year	co2_amount_kiloton
0	United Kingdom	2015	401080
1	United Kingdom	2016	380810
2	United Kingdom	2017	367000
3	United Kingdom	2018	360730
4	United Kingdom	2019	348920
0	United States	2015	4990710
1	United States	2016	4894500
2	United States	2017	4819370
3	United States	2018	4975310
4	United States	2019	4817720

# Fixing the indexes

Did you spot that the new data frame has the same **index values** that were in the two concatenated data frames?

	country_name	year	co2_amount_kiloton
0	United Kingdom	2015	401080
1	United Kingdom	2016	380810
2	United Kingdom	2017	367000
3	United Kingdom	2018	360730
4	United Kingdom	2019	348920
0	United States	2015	4990710
1	United States	2016	4894500
2	United States	2017	4819370
3	United States	2018	4975310
4	United States	2019	4817720

*Reminder:*

Pandas indexes are a kind of label and are mainly used to **access rows by their label.**

For further information, see the lesson **Practise Reshaping Data in Python.**

This isn't very helpful and may cause difficulties later.

However, we can easily fix it...



# Fixing the indexes

Use the **ignore\_index** parameter of **concat()** to fix this issue.

```
co2_uk_us = pd.concat(co2_frames, ignore_index = True)
```

This creates a new default index.

co2\_uk\_us

	country_name	year	co2_amount_kiloton
0	United Kingdom	2015	401080
1	United Kingdom	2016	380810
2	United Kingdom	2017	367000
3	United Kingdom	2018	360730
4	United Kingdom	2019	348920
5	United States	2015	4990710
6	United States	2016	4894500
7	United States	2017	4819370
8	United States	2018	4975310
9	United States	2019	4817720

# Appending data frames with different column names

In the last example, both datasets being combined had exactly **the same columns**, in the **same order**, and with the same **data types**.

Sometimes, the column names are different and you may need to **rename a column** before using **concat()**. Use the pandas **rename()** DataFrame method for this.

songs1

	album	song
0	Revolver	Taxman
1	Revolver	Eleanor Rigby
2	Revolver	I'm Only Sleeping

songs2

	some_album	song
122	Led Zeppelin IV	Four Sticks
123	Led Zeppelin IV	Going to California
124	Led Zeppelin IV	When the Levee Breaks

```
songs2_renamed = songs2.rename(columns={'some_album': 'album'})
```

songs2\_renamed

	album	song
122	Led Zeppelin IV	Four Sticks
123	Led Zeppelin IV	Going to California
124	Led Zeppelin IV	When the Levee Breaks

# Appending data frames with a different column order

**concat()** can also be used to append data frames where the **column order** is different.

**concat()** will automatically sort the columns.



# Appending data frames with different columns

**concat()** can also be used to append data frames which have **different columns**.

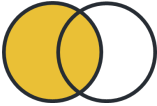
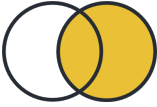
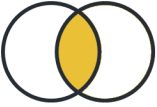
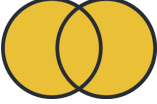


## Next steps

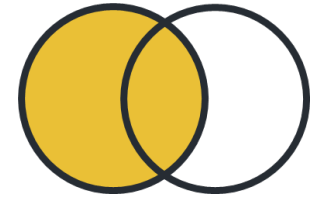
Complete the **Setup** and  
**Appending Rows** section of the  
'Practise Combining Datasets in Python' notebook.

# Joining datasets in Python

The pandas function **merge()** can be used for all the types of joins shown below.

Join Type		What it returns
Left Join		All rows from left dataset, matching rows from right.
Right Join		All rows from right dataset, matching rows from left.
Inner join		Matching rows in left <i>and</i> right datasets.
Outer join		Matching rows in left <i>or</i> right datasets.

# Left join datasets using merge()



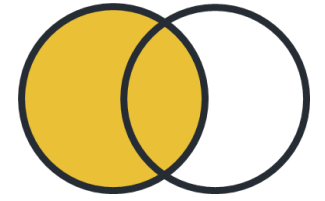
We would like to create a dataset with *all* the rows from the left data frame (**dog\_heights**) and the matching rows from the right data frame (**dog\_good\_with**).

dog_heights		
	name	max_height
0	Border Collie	22
1	Irish Setter	27
2	Labrador Retriever	24
3	Doberman Pinscher	28
4	Yorkshire Terrier	8
5	Beagle	16
6	Great Dane	32
7	Pomeranian	7

dog_good_with			
	name	good_with_children	good_with_other_dogs
0	Border Collie	3	3
1	Irish Setter	5	5
2	Labrador Retriever	5	5
3	Doberman Pinscher	5	3
4	Yorkshire Terrier	5	3
5	Chihuahua	1	3

Both datasets have a **name** column – this will be used as the **key**.

# Left join datasets using merge()



Now that we know:

- the name of the left data frame (**dog\_heights**)
- the name of the right data frame (**dog\_good\_with**)
- the type of join we want to perform (**left join**), and
- the name of the key column (**name**)

...we can create a new data frame using **merge()**...

	name	max_height	good_with_children	good_with_other_dogs
0	Border Collie	22	3	3
1	Irish Setter	27	5	5
2	Labrador Retriever	24	5	5
3	Doberman Pinscher	28	5	3
4	Yorkshire Terrier	8	5	3
5	Beagle	16	NaN	NaN
6	Great Dane	32	NaN	NaN
7	Pomeranian	7	NaN	NaN

```
dogs = pd.merge(dog_heights, dog_good_with, how='left', on='name')
```

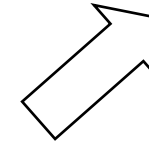
new data frame

left data frame

right data frame

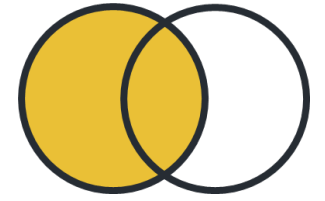
type of join

key





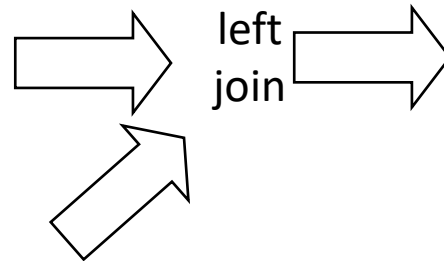
# Left join datasets using merge()



Because some of the rows in the right data frame (**dog\_good\_with**) do not match some of the key values in the left data frame (**dog\_heights**), some of the values are missing (and denoted as **NaN** in pandas).

dog\_heights

	name	max_height
0	Border Collie	22
1	Irish Setter	27
2	Labrador Retriever	24
3	Doberman Pinscher	28
4	Yorkshire Terrier	8
5	Beagle	16
6	Great Dane	32
7	Pomeranian	7



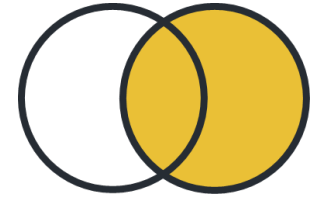
dog\_good\_with

	name	good_with_children	good_with_other_dogs
0	Border Collie	3	3
1	Irish Setter	5	5
2	Labrador Retriever	5	5
3	Doberman Pinscher	5	3
4	Yorkshire Terrier	5	3
5	Chihuahua	1	3

dogs

	name	max_height	good_with_children	good_with_other_dogs
0	Border Collie	22	3	3
1	Irish Setter	27	5	5
2	Labrador Retriever	24	5	5
3	Doberman Pinscher	28	5	3
4	Yorkshire Terrier	8	5	3
5	Beagle	16	NaN	NaN
6	Great Dane	32	NaN	NaN
7	Pomeranian	7	NaN	NaN

# Right join datasets using merge()



We can also use **merge()** to **right join** the same datasets. This time, we want the joined dataset to have *all* the rows from the *right* data frame (**dog\_good\_with**) and the matching rows from the left data frame (**dog\_heights**).

dog\_heights

	name	max_height
0	Border Collie	22
1	Irish Setter	27
2	Labrador Retriever	24
3	Doberman Pinscher	28
4	Yorkshire Terrier	8
5	Beagle	16
6	Great Dane	32
7	Pomeranian	7

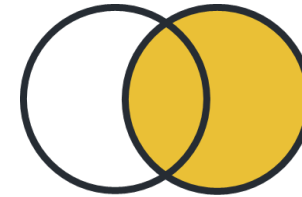
dog\_good\_with

	name	good_with_children	good_with_other_dogs
0	Border Collie	3	3
1	Irish Setter	5	5
2	Labrador Retriever	5	5
3	Doberman Pinscher	5	3
4	Yorkshire Terrier	5	3
5	Chihuahua	1	3

# Your turn...



When we **right join** these datasets, how many rows will the final dataset have?



dog\_heights

	name	max_height
0	Border Collie	22
1	Irish Setter	27
2	Labrador Retriever	24
3	Doberman Pinscher	28
4	Yorkshire Terrier	8
5	Beagle	16
6	Great Dane	32
7	Pomeranian	7

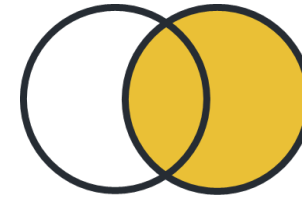
dog\_good\_with

	name	good_with_children	good_with_other_dogs
0	Border Collie	3	3
1	Irish Setter	5	5
2	Labrador Retriever	5	5
3	Doberman Pinscher	5	3
4	Yorkshire Terrier	5	3
5	Chihuahua	1	3

# Your turn...

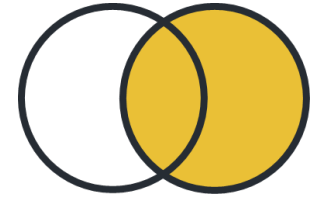


Using a **right** join, there will be 6 rows in the final dataset.  
This is the same number of rows as the **right** dataset.



	name	good_with_children	good_with_other_dogs
0	Border Collie	3	3
1	Irish Setter	5	5
2	Labrador Retriever	5	5
3	Doberman Pinscher	5	3
4	Yorkshire Terrier	5	3
5	Chihuahua	1	3

# Right join datasets using merge()



To use a right join, we set the **how** parameter of **merge()** to 'right'.

As before, we'll use the **name** column as the **key**.

	name	good_with_children	good_with_other_dogs
0	Border Collie	3	3
1	Irish Setter	5	5
2	Labrador Retriever	5	5
3	Doberman Pinscher	5	3
4	Yorkshire Terrier	5	3
5	Chihuahua	1	3

```
right_dogs = pd.merge(dog_heights, dog_good_with, how='right', on='name')
```

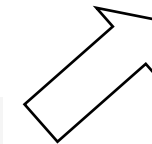
new data frame

left data frame

right data frame

type of join

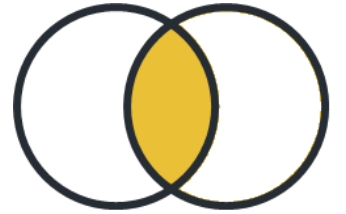
key



## Next steps

Complete the **Left Join** and  
**Right Join** sections of the  
'Practise Combining Datasets in Python' notebook.

# Inner join datasets using merge()



To combine two datasets using an inner join, set the **how** parameter of **merge()** to 'inner'.

We'll continue using the **name** column as the **key**.

	name	max_height	good_with_children	good_with_other_dogs
0	Border Collie	22	3	3
1	Irish Setter	27	5	5
2	Labrador Retriever	24	5	5
3	Doberman Pinscher	28	5	3
4	Yorkshire Terrier	8	5	3

```
inner_dogs = pd.merge(dog_heights, dog_good_with, how='inner', on='name')
```

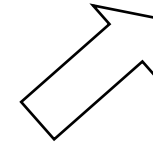
new data frame

left data frame

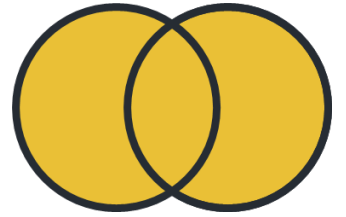
right data frame

type of join

key



# Outer join datasets using merge()



To combine two datasets using an outer join, set the **how** parameter of **merge()** to 'outer'.

As before, we will use the **name** column as the **key**.

	name	max_height	good_with_children	good_with_other_dogs
0	Border Collie	22	3	3
1	Irish Setter	27	5	5
2	Labrador Retriever	24	5	5
3	Doberman Pinscher	28	5	3
4	Yorkshire Terrier	8	5	3
5	Beagle	16	NaN	NaN
6	Great Dane	32	NaN	NaN
7	Pomeranian	7	NaN	NaN
8	Chihuahua	NaN	1	3

```
outer_dogs = pd.merge(dog_heights, dog_good_with, how='outer', on='name')
```

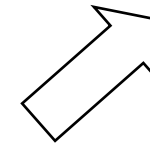
new data frame

left data frame

right data frame

type of join

key





## Next steps

Complete the **Inner Join** and  
**Outer Join** sections of the  
'Practise Combining Datasets in Python' notebook.

# Additional information

Some useful resources:

- pandas **concat()** method (<https://dataed.in/pandasconcat>)
- a clear explanation of the pandas index (<https://dataed.in/pandasindexblog>)
- pandas **rename()** method (<https://dataed.in/pandasrename>)
- pandas **merge()** method (<https://dataed.in/pandasmerge>)
- pandas cheatsheet – see the ‘Combine Data Sets’ section (<https://dataed.in/pandascheatsheet>)

# Learning checklist

I can *use* Python to append rows to a simple dataset.

I can *use* Python to left join simple datasets.

I can *use* Python to right join simple datasets.

I can *use* Python to inner join simple datasets.

I can *use* Python to outer join simple datasets.

# How you can use this lesson



You are free to:

- **Share** – copy and redistribute the material in any medium or format
- **Adapt** – remix, transform and build upon the material

Under the following terms:

- **Attribution** — You must give [appropriate credit](#), provide a link to the license, and [indicate if changes were made](#). You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
- **NonCommercial** — You may not use the material for [commercial purposes](#).
- **ShareAlike** — If you remix, transform, or build upon the material, you must distribute your contributions under the [same license](#) as the original.

© 2023. This work is licensed under a [CC BY-NC-SA 4.0 license](#).

Created by effini in partnership with The Data Lab.



# Alternative format

If you require this document in an alternative format, such as large print or a coloured background, please contact

**hello@effini.com**

or

**4th Floor, The Bayes Centre  
47 Potterrow  
Edinburgh  
EH8 9BT**

