Pandas: Indexing, slicing and filtering DataFrames

- We have seen how to create Pandas Series and DataFrames, and how to read flat files in to Python using Pandas.
- Now we will learn more about indexing, slicing and filtering in Pandas.
- Indexing is assigning row names to make the dataset more readable and accessible.
- Slicing involves using indexes and column names to access certain 'slices' of the data.
- Filtering extracts values or rows that satisfy certain conditions.

Indexing

Index objects

- Indexes are sequences of labels.
- A Series is a 1D array with indexes.
- A DataFrame is a 2D array with Series as columns.
- Indexes are immutable. They can only be changed by overwriting all of the indexes at once.
- Indexes must all have the same data type (like numpy arrays).
- A DataFrame can have more than one set of indexes. This is called hierarchical indexing.

Assigning index names

In [193...
```
import pandas as pd
import os
```

In [194...
```
directory = "C:/Users/cepedazk/Jupyter Notebook/Datasets/"
os.chdir(directory)
```

In [195...
```
nct = pd.ExcelFile("2015 Pass Fail Rate by Centre.xlsx")
print(nct.sheet_names)       # Prints sheet names
```

```
['2015', 'Sheet1']
```

In [196...
```
df_2015 = nct.parse('2015') # Get sheet with name "2015"
```

In [197...
```
display(df_2015)
```

	Centre 2015	PASS (Initial Test)	FAIL (Initial Test)	PASS (Re-test)	FAIL (Re-test)	Total Passes
0	Abbeyfeale	8642	7782	7259	783	15901
1	Arklow	14582	14777	12733	1351	27315
2	Athlone	9044	9121	8720	788	17764
3	Ballina	8648	9352	8638	662	17286
4	Ballinasloe	7332	6767	6396	535	13728
5	Cahir	15884	15639	14309	1262	30193
6	Cahirciveen	1930	1853	1702	76	3632
7	Carlow	13711	16395	14544	1568	28255
8	Carndonagh	3746	3845	3693	375	7439
9	Carrick-on-Shannon	6038	7016	6430	573	12468
10	Castlerea	7048	8229	7529	725	14577
11	Cavan	7083	9212	8820	1031	15903
12	Charleville	8620	9302	8518	792	17138
13	Clifden	1734	2975	2725	310	4459
14	Cork-Blarney	22123	24592	21348	2198	43471
15	Cork-Little Island	40254	35872	34612	3151	74866
16	Deansgrange	54974	44857	43563	4081	98537
17	Derrybeg	2625	3378	3242	270	5867
18	Donegal	5971	5679	5326	606	11297
19	Drogheda	19157	17852	16003	1646	35160
20	Dundalk	10875	12687	11465	1341	22340
21	Ennis	15448	17534	16230	1693	31678
22	Enniscorthy	17397	19221	17929	1721	35326
23	Fonthill	42483	51264	43791	4963	86274
24	Galway	27818	33163	30514	3043	58332
25	Greenhills	36201	38130	36347	3854	72548
26	Kells	15878	18118	15756	1722	31634
27	Kilkenny	16138	12684	12185	798	28323
28	Killarney	11252	8641	8345	551	19597
29	Letterkenny	10388	12883	11833	1231	22221
30	Limerick	25462	26796	26039	2837	51501
31	Longford	6379	8718	7763	980	14142
32	Macroom	9094	7950	7339	577	16433
33	Monaghan	5600	9069	8245	990	13845
34	Mullingar	9527	11395	10300	997	19827
35	Naas	23516	27333	23115	2319	46631
36	Nenagh	13076	13998	12622	1152	25698
37	Northpoint 1	31748	36334	33943	4114	65691
38	Northpoint 2	43951	48898	43822	4523	87773
39	Portlaoise	14511	11563	10465	802	24976
40	Skibbereen	8043	8829	8250	582	16293
41	Sligo	9650	11275	10443	876	20093

	Centre 2015	PASS (Initial Test)	FAIL (Initial Test)	PASS (Re-test)	FAIL (Re-test)	Total Passes
42	Tralee	13911	10466	9793	807	23704
43	Tullamore	11989	10740	9850	832	21839
44	Waterford	21418	19759	17686	1760	39104
45	Westport	11216	11379	10615	814	21831
46	Youghal	8199	8543	7570	649	15769
47	Total	730314	761865	698365	69311	1428679

- To see the last rows of a DataFrame, use `.tail()`
- To see the first rows of a DataFrame, use `.head()`

In [198…
```
display(df_2015.tail())
```

	Centre 2015	PASS (Initial Test)	FAIL (Initial Test)	PASS (Re-test)	FAIL (Re-test)	Total Passes
43	Tullamore	11989	10740	9850	832	21839
44	Waterford	21418	19759	17686	1760	39104
45	Westport	11216	11379	10615	814	21831
46	Youghal	8199	8543	7570	649	15769
47	Total	730314	761865	698365	69311	1428679

- Use `.set_index()` to set a column as an index of the DataFrame.

In [199…
```
df_2015 = df_2015.set_index("Centre 2015")
display(df_2015.head())
```

Centre 2015	PASS (Initial Test)	FAIL (Initial Test)	PASS (Re-test)	FAIL (Re-test)	Total Passes
Abbeyfeale	8642	7782	7259	783	15901
Arklow	14582	14777	12733	1351	27315
Athlone	9044	9121	8720	788	17764
Ballina	8648	9352	8638	662	17286
Ballinasloe	7332	6767	6396	535	13728

- We could also have done either of the following to automatically assign `"Centre 2015"` as the indexes.

In [200…
```
nct = pd.ExcelFile('2015 Pass Fail Rate by Centre.xlsx')
# print(nct.sheet_names)       # Prints sheet names
df_2015 = nct.parse('2015', index_col = 0) # use index_col to establish first column as index of dat
display(df_2015.head())
```

Centre 2015	PASS (Initial Test)	FAIL (Initial Test)	PASS (Re-test)	FAIL (Re-test)	Total Passes
Abbeyfeale	8642	7782	7259	783	15901
Arklow	14582	14777	12733	1351	27315
Athlone	9044	9121	8720	788	17764
Ballina	8648	9352	8638	662	17286
Ballinasloe	7332	6767	6396	535	13728

In [201…
```
nct = pd.ExcelFile('2015 Pass Fail Rate by Centre.xlsx')
# print(nct.sheet_names)       # Prints sheet names
```

```
df_2015 = nct.parse('2015', index_col="Centre 2015") # you can also use the name of the column inste
display(df_2015.head())
```

Centre 2015	PASS (Initial Test)	FAIL (Initial Test)	PASS (Re-test)	FAIL (Re-test)	Total Passes
Abbeyfeale	8642	7782	7259	783	15901
Arklow	14582	14777	12733	1351	27315
Athlone	9044	9121	8720	788	17764
Ballina	8648	9352	8638	662	17286
Ballinasloe	7332	6767	6396	535	13728

Extracting rows and columns

To return a Series: `df_2015["PASS (Initial Test)"]` use one set of square brakets `[ ]`, which returns column "PASS (Initial Test)"

To return a DataFrame use double square brakets `[[ ]]`: `df_2015[["PASS (Initial Test)"]]` which returns the column "PASS (Initial Test)" but as DataFrame type

Note that many but not all operations are shared between Series and DataFrames.

In [202…

```
print(df_2015["FAIL (Re-test)"])
```

```
Centre 2015
Abbeyfeale              783
Arklow                 1351
Athlone                 788
Ballina                 662
Ballinasloe             535
Cahir                  1262
Cahirciveen              76
Carlow                 1568
Carndonagh              375
Carrick-on-Shannon      573
Castlerea               725
Cavan                  1031
Charleville             792
Clifden                 310
Cork-Blarney           2198
Cork-Little Island     3151
Deansgrange            4081
Derrybeg                270
Donegal                 606
Drogheda               1646
Dundalk                1341
Ennis                  1693
Enniscorthy            1721
Fonthill               4963
Galway                 3043
Greenhills             3854
Kells                  1722
Kilkenny                798
Killarney               551
Letterkenny            1231
Limerick               2837
Longford                980
Macroom                 577
Monaghan                990
Mullingar               997
Naas                   2319
Nenagh                 1152
Northpoint 1           4114
Northpoint 2           4523
Portlaoise              802
Skibbereen              582
Sligo                   876
Tralee                  807
Tullamore               832
Waterford              1760
Westport                814
Youghal                 649
Total                69311
Name: FAIL (Re-test), dtype: int64
```

In [203…
```
print(type(df_2015["PASS (Initial Test)"]))
```

```
<class 'pandas.core.series.Series'>
```

In [204…
```
display(type(df_2015[["PASS (Initial Test)"]]))
```

```
pandas.core.frame.DataFrame
```

In [205…
```
display(df_2015[["PASS (Initial Test)"]])
```

Centre 2015	PASS (Initial Test)
Abbeyfeale	8642
Arklow	14582
Athlone	9044
Ballina	8648
Ballinasloe	7332
Cahir	15884
Cahirciveen	1930
Carlow	13711
Carndonagh	3746
Carrick-on-Shannon	6038
Castlerea	7048
Cavan	7083
Charleville	8620
Clifden	1734
Cork-Blarney	22123
Cork-Little Island	40254
Deansgrange	54974
Derrybeg	2625
Donegal	5971
Drogheda	19157
Dundalk	10875
Ennis	15448
Enniscorthy	17397
Fonthill	42483
Galway	27818
Greenhills	36201
Kells	15878
Kilkenny	16138
Killarney	11252
Letterkenny	10388
Limerick	25462
Longford	6379
Macroom	9094
Monaghan	5600
Mullingar	9527
Naas	23516
Nenagh	13076
Northpoint 1	31748
Northpoint 2	43951
Portlaoise	14511
Skibbereen	8043

	PASS (Initial Test)
Centre 2015	
Sligo	9650
Tralee	13911
Tullamore	11989
Waterford	21418
Westport	11216
Youghal	8199
Total	730314

Why would the following command not work?

```
df_2015["PASS (Initial Test)", "FAIL (Initial Test)"]
```

In [206…
```
# df_2015["PASS (Initial Test)", "FAIL (Initial Test)"] # error
```

In [207…
```
# the correct one is
df_2015[["PASS (Initial Test)", "FAIL (Initial Test)", "PASS (Re-test)"]]
```

Out[207]:

Centre 2015	PASS (Initial Test)	FAIL (Initial Test)	PASS (Re-test)
Abbeyfeale	8642	7782	7259
Arklow	14582	14777	12733
Athlone	9044	9121	8720
Ballina	8648	9352	8638
Ballinasloe	7332	6767	6396
Cahir	15884	15639	14309
Cahirciveen	1930	1853	1702
Carlow	13711	16395	14544
Carndonagh	3746	3845	3693
Carrick-on-Shannon	6038	7016	6430
Castlerea	7048	8229	7529
Cavan	7083	9212	8820
Charleville	8620	9302	8518
Clifden	1734	2975	2725
Cork-Blarney	22123	24592	21348
Cork-Little Island	40254	35872	34612
Deansgrange	54974	44857	43563
Derrybeg	2625	3378	3242
Donegal	5971	5679	5326
Drogheda	19157	17852	16003
Dundalk	10875	12687	11465
Ennis	15448	17534	16230
Enniscorthy	17397	19221	17929
Fonthill	42483	51264	43791
Galway	27818	33163	30514
Greenhills	36201	38130	36347
Kells	15878	18118	15756
Kilkenny	16138	12684	12185
Killarney	11252	8641	8345
Letterkenny	10388	12883	11833
Limerick	25462	26796	26039
Longford	6379	8718	7763
Macroom	9094	7950	7339
Monaghan	5600	9069	8245
Mullingar	9527	11395	10300
Naas	23516	27333	23115
Nenagh	13076	13998	12622
Northpoint 1	31748	36334	33943
Northpoint 2	43951	48898	43822
Portlaoise	14511	11563	10465
Skibbereen	8043	8829	8250

Centre 2015	PASS (Initial Test)	FAIL (Initial Test)	PASS (Re-test)
Sligo	9650	11275	10443
Tralee	13911	10466	9793
Tullamore	11989	10740	9850
Waterford	21418	19759	17686
Westport	11216	11379	10615
Youghal	8199	8543	7570
Total	730314	761865	698365

Slicing DataFrames

In [208…
```
# Extract rows 1 to 4 (not including 4).
#  Remember zero-indexing.
df_2015[1:4]
```

Out[208]:

Centre 2015	PASS (Initial Test)	FAIL (Initial Test)	PASS (Re-test)	FAIL (Re-test)	Total Passes
Arklow	14582	14777	12733	1351	27315
Athlone	9044	9121	8720	788	17764
Ballina	8648	9352	8638	662	17286

In [209…
```
# Extract rows 1 to 4 (not including 4) and columns 1 to 3. This gives an error so is not run.

# df_2015[1:4,1:3]
```

Use of `.loc` and `.iloc`

- ... can extract a value from a Pandas DataFrame in a number of ways:

  `df['col']['row']`

  `df.col['row']`

  `df.loc['row', 'col']`

  `df.iloc[row_no, col_no]`

Use double square brackets to make sure the result is a DataFrame: `df.loc[ [ 'row', 'col' ] ]`

`df[ [ 'col1', 'col4', 'col5' ] ]` to select the columns called `'col1'`, `'col4'`, `'col5'` in a DataFrame. Notice we give a list of column names.

- loc is label-based.
- iloc is integer position-based.
- Remember that there is zero-based indexing!

`.loc` to select rows and columns by label(s)

`df.loc['row', 'col']`

In [210…
```
display(df_2015.head())
```

Centre 2015	PASS (Initial Test)	FAIL (Initial Test)	PASS (Re-test)	FAIL (Re-test)	Total Passes
Abbeyfeale	8642	7782	7259	783	15901
Arklow	14582	14777	12733	1351	27315
Athlone	9044	9121	8720	788	17764
Ballina	8648	9352	8638	662	17286
Ballinasloe	7332	6767	6396	535	13728

In [211…
```
# Ballina indexed row as a Series
print(type(df_2015.loc["Ballina"]))
display(df_2015.loc["Ballina"])
```

```
<class 'pandas.core.series.Series'>
PASS (Initial Test)     8648
FAIL (Initial Test)     9352
PASS (Re-test)          8638
FAIL (Re-test)           662
Total Passes           17286
Name: Ballina, dtype: int64
```

In [212…
```
# Ballina indexed row as a DataFrame
print(type(df_2015.loc[["Athlone"]]))
display(df_2015.loc[["Athlone"]])
```

```
<class 'pandas.core.frame.DataFrame'>
```

Centre 2015	PASS (Initial Test)	FAIL (Initial Test)	PASS (Re-test)	FAIL (Re-test)	Total Passes
Athlone	9044	9121	8720	788	17764

In [213…
```
# This gives an error. Why?
# df_2015.loc["Ballina", "Dundalk"]
```

In [214…
```
# Ballina and Dundalk indexed rows as a DataFrame
#  To extract more than one column, always use double set of square brackets
df_2015.loc[["Ballina", "Dundalk", "Abbeyfeale"]]
```

Out[214]:

Centre 2015	PASS (Initial Test)	FAIL (Initial Test)	PASS (Re-test)	FAIL (Re-test)	Total Passes
Ballina	8648	9352	8638	662	17286
Dundalk	10875	12687	11465	1341	22340
Abbeyfeale	8642	7782	7259	783	15901

In [215…
```
# Extract column using loc
#  with colon (:) selecting all rows
df_2015.loc[:, ["PASS (Initial Test)"]]
```

Out[215]:

Centre 2015	PASS (Initial Test)
Abbeyfeale	8642
Arklow	14582
Athlone	9044
Ballina	8648
Ballinasloe	7332
Cahir	15884
Cahirciveen	1930
Carlow	13711
Carndonagh	3746
Carrick-on-Shannon	6038
Castlerea	7048
Cavan	7083
Charleville	8620
Clifden	1734
Cork-Blarney	22123
Cork-Little Island	40254
Deansgrange	54974
Derrybeg	2625
Donegal	5971
Drogheda	19157
Dundalk	10875
Ennis	15448
Enniscorthy	17397
Fonthill	42483
Galway	27818
Greenhills	36201
Kells	15878
Kilkenny	16138
Killarney	11252
Letterkenny	10388
Limerick	25462
Longford	6379
Macroom	9094
Monaghan	5600
Mullingar	9527
Naas	23516
Nenagh	13076
Northpoint 1	31748
Northpoint 2	43951
Portlaoise	14511
Skibbereen	8043

	PASS (Initial Test)
Centre 2015	
Sligo	9650
Tralee	13911
Tullamore	11989
Waterford	21418
Westport	11216
Youghal	8199
Total	730314

In [216…

```
# Extract multiple columns using loc
#  and use colon (:) to select all rows
df_2015.loc[:, ["PASS (Initial Test)", "FAIL (Initial Test)"]]
```

Out[216]:

Centre 2015	PASS (Initial Test)	FAIL (Initial Test)
Abbeyfeale	8642	7782
Arklow	14582	14777
Athlone	9044	9121
Ballina	8648	9352
Ballinasloe	7332	6767
Cahir	15884	15639
Cahirciveen	1930	1853
Carlow	13711	16395
Carndonagh	3746	3845
Carrick-on-Shannon	6038	7016
Castlerea	7048	8229
Cavan	7083	9212
Charleville	8620	9302
Clifden	1734	2975
Cork-Blarney	22123	24592
Cork-Little Island	40254	35872
Deansgrange	54974	44857
Derrybeg	2625	3378
Donegal	5971	5679
Drogheda	19157	17852
Dundalk	10875	12687
Ennis	15448	17534
Enniscorthy	17397	19221
Fonthill	42483	51264
Galway	27818	33163
Greenhills	36201	38130
Kells	15878	18118
Kilkenny	16138	12684
Killarney	11252	8641
Letterkenny	10388	12883
Limerick	25462	26796
Longford	6379	8718
Macroom	9094	7950
Monaghan	5600	9069
Mullingar	9527	11395
Naas	23516	27333
Nenagh	13076	13998
Northpoint 1	31748	36334
Northpoint 2	43951	48898
Portlaoise	14511	11563
Skibbereen	8043	8829

Centre 2015	PASS (Initial Test)	FAIL (Initial Test)
Sligo	9650	11275
Tralee	13911	10466
Tullamore	11989	10740
Waterford	21418	19759
Westport	11216	11379
Youghal	8199	8543
Total	730314	761865

In [217…
```
# Extract rows and columns
print(df_2015.loc["Dundalk", ["PASS (Initial Test)", "FAIL (Initial Test)"]])
```

```
PASS (Initial Test)    10875
FAIL (Initial Test)    12687
Name: Dundalk, dtype: int64
```

`.iloc` to extract rows and columns by index

`df.iloc[row_no, col_no]`

In [218…
```
display(df_2015.head())
```

Centre 2015	PASS (Initial Test)	FAIL (Initial Test)	PASS (Re-test)	FAIL (Re-test)	Total Passes
Abbeyfeale	8642	7782	7259	783	15901
Arklow	14582	14777	12733	1351	27315
Athlone	9044	9121	8720	788	17764
Ballina	8648	9352	8638	662	17286
Ballinasloe	7332	6767	6396	535	13728

In [219…
```
# Extract second row as a Series
print(type(df_2015.iloc[1]))
display(df_2015.iloc[1])
```

```
<class 'pandas.core.series.Series'>
PASS (Initial Test)    14582
FAIL (Initial Test)    14777
PASS (Re-test)         12733
FAIL (Re-test)          1351
Total Passes           27315
Name: Arklow, dtype: int64
```

In [220…
```
# Extract second row as a DataFrame
print(type(df_2015.iloc[[1]]))
display(df_2015.iloc[[1]])
```

```
<class 'pandas.core.frame.DataFrame'>
```

Centre 2015	PASS (Initial Test)	FAIL (Initial Test)	PASS (Re-test)	FAIL (Re-test)	Total Passes
Arklow	14582	14777	12733	1351	27315

In [221…
```
# Extract row indexes 1, 2 and 3
df_2015.iloc[[1,3,2,7,5]]
```

Out[221]:

Centre 2015	PASS (Initial Test)	FAIL (Initial Test)	PASS (Re-test)	FAIL (Re-test)	Total Passes
Arklow	14582	14777	12733	1351	27315
Ballina	8648	9352	8638	662	17286
Athlone	9044	9121	8720	788	17764
Carlow	13711	16395	14544	1568	28255
Cahir	15884	15639	14309	1262	30193

In [222…

```
# Extract row indexes 1 to 6
df_2015.iloc[1:6] # similar to df_2015.iloc[[1,3,2,7,5]]

# Note, you do not need to use double square brackets, as you are sending a range
#  rather than a list.
```

Out[222]:

Centre 2015	PASS (Initial Test)	FAIL (Initial Test)	PASS (Re-test)	FAIL (Re-test)	Total Passes
Arklow	14582	14777	12733	1351	27315
Athlone	9044	9121	8720	788	17764
Ballina	8648	9352	8638	662	17286
Ballinasloe	7332	6767	6396	535	13728
Cahir	15884	15639	14309	1262	30193

In [223…

```
# Extract row indexes 1 and 2
#  Same as Numpy numbers, you can use colon symbol (:) to indicate
#   range of rows and columns
df_2015.iloc[1:3, :]
```

Out[223]:

Centre 2015	PASS (Initial Test)	FAIL (Initial Test)	PASS (Re-test)	FAIL (Re-test)	Total Passes
Arklow	14582	14777	12733	1351	27315
Athlone	9044	9121	8720	788	17764

In [224…

```
# Get all rows, columns 0 and 3
#  When using a list of indexes, instead of a range (:)
#   watch out, here to indicate specific columns, use a list and the indexes
df_2015.iloc[:, [0,3]]

# this is not the same as df_2015.iloc[:, 0:3] why?
```

Out[224]:

Centre 2015	PASS (Initial Test)	FAIL (Re-test)
Abbeyfeale	8642	783
Arklow	14582	1351
Athlone	9044	788
Ballina	8648	662
Ballinasloe	7332	535
Cahir	15884	1262
Cahirciveen	1930	76
Carlow	13711	1568
Carndonagh	3746	375
Carrick-on-Shannon	6038	573
Castlerea	7048	725
Cavan	7083	1031
Charleville	8620	792
Clifden	1734	310
Cork-Blarney	22123	2198
Cork-Little Island	40254	3151
Deansgrange	54974	4081
Derrybeg	2625	270
Donegal	5971	606
Drogheda	19157	1646
Dundalk	10875	1341
Ennis	15448	1693
Enniscorthy	17397	1721
Fonthill	42483	4963
Galway	27818	3043
Greenhills	36201	3854
Kells	15878	1722
Kilkenny	16138	798
Killarney	11252	551
Letterkenny	10388	1231
Limerick	25462	2837
Longford	6379	980
Macroom	9094	577
Monaghan	5600	990
Mullingar	9527	997
Naas	23516	2319
Nenagh	13076	1152
Northpoint 1	31748	4114
Northpoint 2	43951	4523
Portlaoise	14511	802
Skibbereen	8043	582

Centre 2015	PASS (Initial Test)	FAIL (Re-test)
Sligo	9650	876
Tralee	13911	807
Tullamore	11989	832
Waterford	21418	1760
Westport	11216	814
Youghal	8199	649
Total	730314	69311

Slicing DataFrames

We will use the Premier League dataset to show some slicing of a DataFrame. Each row represents a match in the 2018/19 season.

The columns we will use are:

1. HomeTeam: The team that played at home in the match
2. AwayTeam: The team that played away from home in the match
3. FTHG: Full Time Home Goals
4. FTAG: Full Time Away Goals
5. HTHG: Half Time Home Goals
6. HTAG: Half Time Away Goals
7. Referee: The referee name

In [225…
```
import os
import pandas as pd

print(os.getcwd())
directory = "C:/Users/cepedazk/Jupyter Notebook/Datasets/"
os.chdir(directory)
```

```
C:\Users\cepedazk\Jupyter Notebook\Datasets
```

In [226…
```
pl = pd.read_csv("premier_league_1819.csv")
print(pl.columns)  # To see the columns in the dataset.
```

```
Index(['Date', 'HomeTeam', 'AwayTeam', 'FTHG', 'FTAG', 'FTR', 'HTHG', 'HTAG',
       'HTR', 'Referee', 'HS', 'AS', 'HST', 'AST', 'HF', 'AF', 'HC', 'AC',
       'HY', 'AY', 'HR', 'AR'],
      dtype='object')
```

In [227…
```
pl
```

Out[227]:

	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	Referee	...	HST	AST	HF	AF
0	10/08/2018	Man United	Leicester	2	1	H	1	0	H	A Marriner	...	6	4	11	8
1	11/08/2018	Bournemouth	Cardiff	2	0	H	1	0	H	K Friend	...	4	1	11	9
2	11/08/2018	Fulham	Crystal Palace	0	2	A	0	1	A	M Dean	...	6	9	9	11
3	11/08/2018	Huddersfield	Chelsea	0	3	A	0	2	A	C Kavanagh	...	1	4	9	8
4	11/08/2018	Newcastle	Tottenham	1	2	A	1	2	A	M Atkinson	...	2	5	11	12
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
375	12/05/2019	Liverpool	Wolves	2	0	H	1	0	H	M Atkinson	...	5	2	3	11
376	12/05/2019	Man United	Cardiff	0	2	A	0	1	A	J Moss	...	10	4	9	6
377	12/05/2019	Southampton	Huddersfield	1	1	D	1	0	H	L Probert	...	3	3	8	6
378	12/05/2019	Tottenham	Everton	2	2	D	1	0	H	A Marriner	...	3	9	10	13
379	12/05/2019	Watford	West Ham	1	4	A	0	2	A	C Kavanagh	...	8	9	10	10

380 rows × 22 columns

In [228…
```
pl['HTHG']
```

Out[228]:
```
0      1
1      1
2      0
3      0
4      1
      ..
375    1
376    0
377    1
378    1
379    0
Name: HTHG, Length: 380, dtype: int64
```

In [229…
```
pl['HTHG'][1:5]
```

Out[229]:
```
1    1
2    0
3    0
4    1
Name: HTHG, dtype: int64
```

- Use of `.loc()` in DataFrame

- To get all rows, and some columns (all columns from 'FTHG' to 'HTAG' inclusive).
- How does this slicing using column names differ from slicing using column numbers?

In [230…
```
pl.loc[:, 'FTHG':'HTAG']
```

Out[230]:

	FTHG	FTAG	FTR	HTHG	HTAG
0	2	1	H	1	0
1	2	0	H	1	0
2	0	2	A	0	1
3	0	3	A	0	2
4	1	2	A	1	2
...	...	...	...	...	...
375	2	0	H	1	0
376	0	2	A	0	1
377	1	1	D	1	0
378	2	2	D	1	0
379	1	4	A	0	2

380 rows × 5 columns

- The rows or columns you want to select will not always be consecutive. To select certain rows or columns, use lists:

In [231…
```
pl.loc[ :, ['FTHG', 'Referee', 'HTHG' ] ]
```

Out[231]:

	FTHG	Referee	HTHG
0	2	A Marriner	1
1	2	K Friend	1
2	0	M Dean	0
3	0	C Kavanagh	0
4	1	M Atkinson	1
...	...	...	...
375	2	M Atkinson	1
376	0	J Moss	0
377	1	L Probert	1
378	2	A Marriner	1
379	1	C Kavanagh	0

380 rows × 3 columns

Use of `.iloc[]` in DataFrame

In [232…
```
# Get first 4 rows of pl
pl.iloc[0:4, :]
```

Out[232]:

	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	Referee	...	HST	AST	HF	AF	HC
0	10/08/2018	Man United	Leicester	2	1	H	1	0	H	A Marriner	...	6	4	11	8	2
1	11/08/2018	Bournemouth	Cardiff	2	0	H	1	0	H	K Friend	...	4	1	11	9	7
2	11/08/2018	Fulham	Crystal Palace	0	2	A	0	1	A	M Dean	...	6	9	9	11	5
3	11/08/2018	Huddersfield	Chelsea	0	3	A	0	2	A	C Kavanagh	...	1	4	9	8	2

4 rows × 22 columns

In [233…
```
pl.iloc[: , 2:5]     # Get columns indexed 2, 3 and 4 from pl
```

Out[233]:

	AwayTeam	FTHG	FTAG
0	Leicester	2	1
1	Cardiff	2	0
2	Crystal Palace	0	2
3	Chelsea	0	3
4	Tottenham	1	2
...	...	...	...
375	Wolves	2	0
376	Cardiff	0	2
377	Huddersfield	1	1
378	Everton	2	2
379	West Ham	1	4

380 rows × 3 columns

In [234…
```
pl.iloc[ 0:4, 2:5 ]     # Get columns 2,3,4 of the first 4 rows of pl
```

Out[234]:

	AwayTeam	FTHG	FTAG
0	Leicester	2	1
1	Cardiff	2	0
2	Crystal Palace	0	2
3	Chelsea	0	3

- The rows or columns you want to select will not always be consecutive. To select certain rows or columns, use lists:

In [235…
```
pl.iloc[ [0, 2, 4], 2:5 ]
```

Out[235]:

	AwayTeam	FTHG	FTAG
0	Leicester	2	1
2	Crystal Palace	0	2
4	Tottenham	1	2

Filtering DataFrames with conditions

- We will use the Premier League dataset to show how to filter a DataFrame: extract values or rows that satisfy certain conditions.
- We can use conditions to filter values within a column in a data frame.
  - `==` for equal to
  - `>` for greater than
  - `>=` for greater than or equal to
  - `<` for less than
  - `<=` for less than or equal to
  - `!=` for not equal to

In [236…
```
import os
os.getcwd()
os.chdir(directory)
pl = pd.read_csv("premier_league_1819.csv")

display(pl.head(10))
```

	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	Referee	…	HST	AST	HF	AF	HC
0	10/08/2018	Man United	Leicester	2	1	H	1	0	H	A Marriner	…	6	4	11	8	2
1	11/08/2018	Bournemouth	Cardiff	2	0	H	1	0	H	K Friend	…	4	1	11	9	7
2	11/08/2018	Fulham	Crystal Palace	0	2	A	0	1	A	M Dean	…	6	9	9	11	5
3	11/08/2018	Huddersfield	Chelsea	0	3	A	0	2	A	C Kavanagh	…	1	4	9	8	2
4	11/08/2018	Newcastle	Tottenham	1	2	A	1	2	A	M Atkinson	…	2	5	11	12	3
5	11/08/2018	Watford	Brighton	2	0	H	1	0	H	J Moss	…	5	0	10	16	8
6	11/08/2018	Wolves	Everton	2	2	D	1	1	D	C Pawson	…	4	5	8	7	3
7	12/08/2018	Arsenal	Man City	0	2	A	0	1	A	M Oliver	…	3	8	11	14	2
8	12/08/2018	Liverpool	West Ham	4	0	H	2	0	H	A Taylor	…	8	2	14	9	5
9	12/08/2018	Southampton	Burnley	0	0	D	0	0	D	G Scott	…	3	6	10	9	8

10 rows × 22 columns

In [237…
```
pl.FTHG == 4
# Gives True and False values to values that are equal to 4
# It returns a Series of type boolean
```

Out[237]:
```
0      False
1      False
2      False
3      False
4      False
       ...
375    False
376    False
377    False
378    False
379    False
Name: FTHG, Length: 380, dtype: bool
```

In [238…
```
sum(pl.FTHG == 4) # when we sum up booleans, we are summing up True as "1" and False as "0",
# The results shows that there are 22 True values, in order words, there are 22 FTHG equal to 4.
```

Out[238]:
```
22
```

- To return the rows where the column "FTHG" has a value of 4, use the Boolean indexing `pl.FTHG == 4` to filter the DataFrame, keeping only rows where "FTHG" is 4.

In [239…
```
condition = pl.FTHG == 4
print(type(condition))
print(condition)
```

```
<class 'pandas.core.series.Series'>
0      False
1      False
2      False
3      False
4      False
       ...
375    False
376    False
377    False
378    False
379    False
Name: FTHG, Length: 380, dtype: bool
```

In [240…
```
pl[ condition ]     # Gives all rows where FTHG equals 4
```

Out[240]:

	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	Referee	...	HST	AST	HF	AF
8	12/08/2018	Liverpool	West Ham	4	0	H	2	0	H	A Taylor	...	8	2	14	9
26	26/08/2018	Fulham	Burnley	4	2	H	3	2	H	D Coote	...	12	2	11	8
40	15/09/2018	Bournemouth	Leicester	4	2	H	3	0	H	C Pawson	...	5	8	13	15
41	15/09/2018	Chelsea	Cardiff	4	1	H	2	1	H	J Moss	...	7	2	8	10
51	22/09/2018	Burnley	Bournemouth	4	0	H	2	0	H	A Taylor	...	5	5	17	6
81	20/10/2018	Cardiff	Fulham	4	2	H	2	2	D	K Friend	...	5	4	15	16
93	27/10/2018	Liverpool	Cardiff	4	1	H	1	0	H	S Attwell	...	7	1	6	4
105	03/11/2018	West Ham	Burnley	4	2	H	1	1	D	R East	...	10	3	7	9
137	02/12/2018	Arsenal	Tottenham	4	2	H	1	2	A	M Dean	...	7	6	15	17
156	08/12/2018	Man United	Fulham	4	1	H	3	0	H	L Probert	...	11	4	11	15
185	26/12/2018	Liverpool	Newcastle	4	0	H	1	0	H	G Scott	...	8	2	7	9
198	30/12/2018	Man United	Bournemouth	4	1	H	3	1	H	L Mason	...	8	3	10	7
200	01/01/2019	Arsenal	Fulham	4	1	H	1	0	H	G Scott	...	9	4	7	12
222	19/01/2019	Liverpool	Crystal Palace	4	3	H	0	1	A	J Moss	...	9	3	6	8
227	19/01/2019	Wolves	Leicester	4	3	H	2	0	H	C Kavanagh	...	7	6	11	10
231	29/01/2019	Fulham	Brighton	4	2	H	0	2	A	L Probert	...	7	6	10	5
236	30/01/2019	Bournemouth	Chelsea	4	0	H	0	0	D	R East	...	7	7	8	6
298	10/03/2019	Liverpool	Burnley	4	2	H	2	1	H	A Marriner	...	5	2	4	7
301	16/03/2019	West Ham	Huddersfield	4	3	H	1	2	A	J Moss	...	5	5	7	15
314	02/04/2019	Watford	Fulham	4	1	H	1	1	D	R East	...	7	7	12	5
331	13/04/2019	Tottenham	Huddersfield	4	0	H	2	0	H	L Mason	...	5	1	10	12
344	21/04/2019	Everton	Man United	4	0	H	2	0	H	P Tierney	...	8	1	11	7

22 rows × 22 columns

In [241…
```
pl[ pl.FTHG == 4 ] # this is the same as the previous
```

Out[241]:

	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	Referee	...	HST	AST	HF	AF
8	12/08/2018	Liverpool	West Ham	4	0	H	2	0	H	A Taylor	...	8	2	14	9
26	26/08/2018	Fulham	Burnley	4	2	H	3	2	H	D Coote	...	12	2	11	8
40	15/09/2018	Bournemouth	Leicester	4	2	H	3	0	H	C Pawson	...	5	8	13	15
41	15/09/2018	Chelsea	Cardiff	4	1	H	2	1	H	J Moss	...	7	2	8	10
51	22/09/2018	Burnley	Bournemouth	4	0	H	2	0	H	A Taylor	...	5	5	17	6
81	20/10/2018	Cardiff	Fulham	4	2	H	2	2	D	K Friend	...	5	4	15	16
93	27/10/2018	Liverpool	Cardiff	4	1	H	1	0	H	S Attwell	...	7	1	6	4
105	03/11/2018	West Ham	Burnley	4	2	H	1	1	D	R East	...	10	3	7	9
137	02/12/2018	Arsenal	Tottenham	4	2	H	1	2	A	M Dean	...	7	6	15	17
156	08/12/2018	Man United	Fulham	4	1	H	3	0	H	L Probert	...	11	4	11	15
185	26/12/2018	Liverpool	Newcastle	4	0	H	1	0	H	G Scott	...	8	2	7	9
198	30/12/2018	Man United	Bournemouth	4	1	H	3	1	H	L Mason	...	8	3	10	7
200	01/01/2019	Arsenal	Fulham	4	1	H	1	0	H	G Scott	...	9	4	7	12
222	19/01/2019	Liverpool	Crystal Palace	4	3	H	0	1	A	J Moss	...	9	3	6	8
227	19/01/2019	Wolves	Leicester	4	3	H	2	0	H	C Kavanagh	...	7	6	11	10
231	29/01/2019	Fulham	Brighton	4	2	H	0	2	A	L Probert	...	7	6	10	5
236	30/01/2019	Bournemouth	Chelsea	4	0	H	0	0	D	R East	...	7	7	8	6
298	10/03/2019	Liverpool	Burnley	4	2	H	2	1	H	A Marriner	...	5	2	4	7
301	16/03/2019	West Ham	Huddersfield	4	3	H	1	2	A	J Moss	...	5	5	7	15
314	02/04/2019	Watford	Fulham	4	1	H	1	1	D	R East	...	7	7	12	5
331	13/04/2019	Tottenham	Huddersfield	4	0	H	2	0	H	L Mason	...	5	1	10	12
344	21/04/2019	Everton	Man United	4	0	H	2	0	H	P Tierney	...	8	1	11	7

22 rows × 22 columns

- We can select single columns in a DataFrame by using `.<column_name>`

In [242…

```
print(pl["HTHG"])
```

```
0      1
1      1
2      0
3      0
4      1
      ..
375    1
376    0
377    1
378    1
379    0
Name: HTHG, Length: 380, dtype: int64
```

- And we can use this to select from a column, e.g., `HTHG`, values based on a condition, e.g., `pl.FTHG == 4`

In [243…

```
print(pl.HTHG[pl.FTHG == 4]) # select values from column HTHG where FTHG is equal to 4
```

```
8      2
26     3
40     3
41     2
51     2
81     2
93     1
105    1
137    1
156    3
185    1
198    3
200    1
222    0
227    2
231    0
236    0
298    2
301    1
314    1
331    2
344    2
Name: HTHG, dtype: int64
```

In [244…

```
# Remember: first set of square brackets is rows, second square brackets is columns

#  In this example, you are filtering rows,
#  and then selecting specific column names using a list

display(pl[pl.FTHG == 4][["HomeTeam", "AwayTeam","HTHG", "HTAG", "FTAG"]])
```

	HomeTeam	AwayTeam	HTHG	HTAG	FTAG
8	Liverpool	West Ham	2	0	0
26	Fulham	Burnley	3	2	2
40	Bournemouth	Leicester	3	0	2
41	Chelsea	Cardiff	2	1	1
51	Burnley	Bournemouth	2	0	0
81	Cardiff	Fulham	2	2	2
93	Liverpool	Cardiff	1	0	1
105	West Ham	Burnley	1	1	2
137	Arsenal	Tottenham	1	2	2
156	Man United	Fulham	3	0	1
185	Liverpool	Newcastle	1	0	0
198	Man United	Bournemouth	3	1	1
200	Arsenal	Fulham	1	0	1
222	Liverpool	Crystal Palace	0	1	3
227	Wolves	Leicester	2	0	3
231	Fulham	Brighton	0	2	2
236	Bournemouth	Chelsea	0	0	0
298	Liverpool	Burnley	2	1	2
301	West Ham	Huddersfield	1	2	3
314	Watford	Fulham	1	1	1
331	Tottenham	Huddersfield	2	0	0
344	Everton	Man United	2	0	0

Examples with greater than `>` and greather than or equal to `>=`

In [245…
```
print(pl.FTHG > 4)      # Gives True and False values
```

```
0      False
1      False
2      False
3      False
4      False
       ...
375    False
376    False
377    False
378    False
379    False
Name: FTHG, Length: 380, dtype: bool
```

In [246…
```
display(pl[pl.FTHG > 4])      # Gives all rows where FTHG > 4
```

	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	Referee	...	HST	AST	HF	AF
18	19/08/2018	Man City	Huddersfield	6	1	H	3	1	H	A Marriner	...	14	1	9	4
84	20/10/2018	Man City	Burnley	5	0	H	1	0	H	J Moss	...	10	0	11	5
108	04/11/2018	Man City	Southampton	6	1	H	4	1	H	L Mason	...	8	6	14	9
187	26/12/2018	Tottenham	Bournemouth	5	0	H	3	0	H	C Kavanagh	...	7	4	4	8
193	29/12/2018	Liverpool	Arsenal	5	1	H	4	1	H	M Oliver	...	10	2	8	13
243	02/02/2019	Chelsea	Huddersfield	5	0	H	2	0	H	P Tierney	...	7	2	8	5
258	10/02/2019	Man City	Chelsea	6	0	H	4	0	H	M Dean	...	9	4	9	13
273	27/02/2019	Arsenal	Bournemouth	5	1	H	2	1	H	C Kavanagh	...	6	5	11	9
276	27/02/2019	Liverpool	Watford	5	0	H	2	0	H	G Scott	...	10	3	5	7
350	26/04/2019	Liverpool	Huddersfield	5	0	H	3	0	H	K Friend	...	7	1	5	14
372	12/05/2019	Crystal Palace	Bournemouth	5	3	H	3	1	H	R East	...	8	8	11	8

11 rows × 22 columns

In [247…
```
pl[pl.FTHG >= 5]      # Gives the same as the last command. Why?
```

Out[247]:

	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	Referee	...	HST	AST	HF	AF
18	19/08/2018	Man City	Huddersfield	6	1	H	3	1	H	A Marriner	...	14	1	9	4
84	20/10/2018	Man City	Burnley	5	0	H	1	0	H	J Moss	...	10	0	11	5
108	04/11/2018	Man City	Southampton	6	1	H	4	1	H	L Mason	...	8	6	14	9
187	26/12/2018	Tottenham	Bournemouth	5	0	H	3	0	H	C Kavanagh	...	7	4	4	8
193	29/12/2018	Liverpool	Arsenal	5	1	H	4	1	H	M Oliver	...	10	2	8	13
243	02/02/2019	Chelsea	Huddersfield	5	0	H	2	0	H	P Tierney	...	7	2	8	5
258	10/02/2019	Man City	Chelsea	6	0	H	4	0	H	M Dean	...	9	4	9	13
273	27/02/2019	Arsenal	Bournemouth	5	1	H	2	1	H	C Kavanagh	...	6	5	11	9
276	27/02/2019	Liverpool	Watford	5	0	H	2	0	H	G Scott	...	10	3	5	7
350	26/04/2019	Liverpool	Huddersfield	5	0	H	3	0	H	K Friend	...	7	1	5	14
372	12/05/2019	Crystal Palace	Bournemouth	5	3	H	3	1	H	R East	...	8	8	11	8

11 rows × 22 columns

In [248…

```
print(pl.HTHG[pl.FTHG > 4]) # Gives HTHG values when FTHG > 4
```

```
18     3
84     1
108    4
187    3
193    4
243    2
258    4
273    2
276    2
350    3
372    3
Name: HTHG, dtype: int64
```

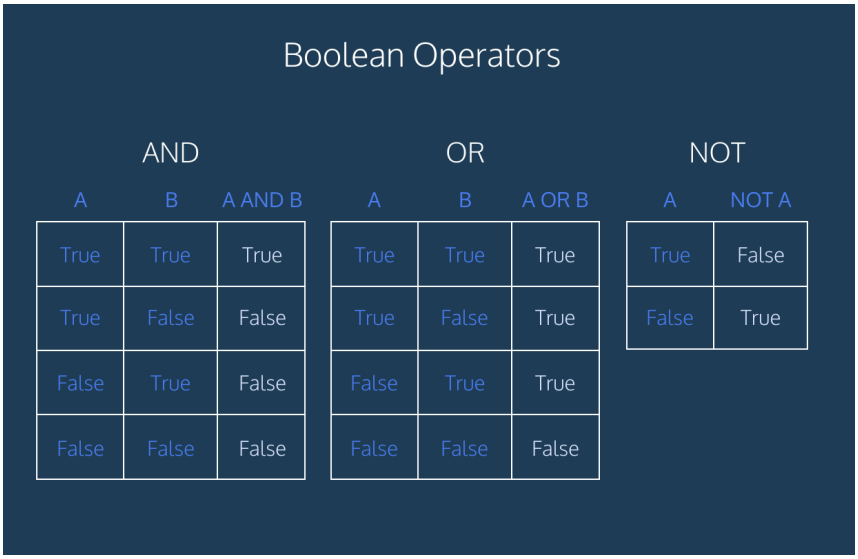
In [249…

```
display(pl[pl.FTHG > 4][["HTHG", "HTAG"]])   # Gives HTHG and HTAG values when FTHG > 4
```

	HTHG	HTAG
18	3	1
84	1	0
108	4	1
187	3	0
193	4	1
243	2	0
258	4	0
273	2	1
276	2	0
350	3	0
372	3	1

Filtering based on multiple conditions

- Combine conditions using AND `&`, OR `|` and NOT `!`
- Use brackets when combining conditions
- To include more than one condition, use parentheses `( )`



In [250…
```
# The command below gives an error because parentheses were not used to separate conditions

# pl.FTHG > 4 & pl.FTAG > 1
```

In [251…
```
pl
```

Out[251]:

	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	Referee	...	HST	AST	HF	AF
0	10/08/2018	Man United	Leicester	2	1	H	1	0	H	A Marriner	...	6	4	11	8
1	11/08/2018	Bournemouth	Cardiff	2	0	H	1	0	H	K Friend	...	4	1	11	9
2	11/08/2018	Fulham	Crystal Palace	0	2	A	0	1	A	M Dean	...	6	9	9	11
3	11/08/2018	Huddersfield	Chelsea	0	3	A	0	2	A	C Kavanagh	...	1	4	9	8
4	11/08/2018	Newcastle	Tottenham	1	2	A	1	2	A	M Atkinson	...	2	5	11	12
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
375	12/05/2019	Liverpool	Wolves	2	0	H	1	0	H	M Atkinson	...	5	2	3	11
376	12/05/2019	Man United	Cardiff	0	2	A	0	1	A	J Moss	...	10	4	9	6
377	12/05/2019	Southampton	Huddersfield	1	1	D	1	0	H	L Probert	...	3	3	8	6
378	12/05/2019	Tottenham	Everton	2	2	D	1	0	H	A Marriner	...	3	9	10	13
379	12/05/2019	Watford	West Ham	1	4	A	0	2	A	C Kavanagh	...	8	9	10	10

380 rows × 22 columns

In [252…
```
(pl.FTHG > 4) & (pl.FTAG > 1)   # Gives True when both conditions are satisfied
```

Out[252]:
```
0      False
1      False
2      False
3      False
4      False
       ...
375    False
376    False
377    False
378    False
379    False
Length: 380, dtype: bool
```

In [253…
```
display(pl[(pl.FTHG > 4) & (pl.FTAG > 1)])     # Gives all rows where both conditions are satisfied
```

	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	Referee	...	HST	AST	HF	AF	H
372	12/05/2019	Crystal Palace	Bournemouth	5	3	H	3	1	H	R East	...	8	8	11	8	

1 rows × 22 columns

In [254…
```
print(pl.HTHG[(pl.FTHG > 4) & (pl.FTAG > 1)]) # Gives HTHG values when both conditions are satisfied
```

```
372    3
Name: HTHG, dtype: int64
```

- You can add more conditions, remember to use the parentheses `( )`

In [255…
```
display(pl[(pl.FTHG > 4) & (pl.FTAG > 1) & (pl.HTHG > 2)][["HTHG", "HTAG", "Referee"]])

# Gives HTHG, HTAG and Referee values when both conditions are satisfied
```

	HTHG	HTAG	Referee
372	3	1	R East

In [256…
```
# Gives an error because brackets were not used to separate conditions

# pl.FTHG > 5 | pl.FTAG > 4
```

In [257…
```
(pl.FTHG > 5) | (pl.FTAG > 4)    # Gives True when either condition is satisfied
```

Out[257]:
```
0      False
1      False
2      False
3      False
4      False
       ...
375    False
376    False
377    False
378    False
379    False
Length: 380, dtype: bool
```

In [258…
```
pl[(pl.FTHG > 5) | (pl.FTAG > 4)]     # Gives all rows where either condition is satisfied
```

Out[258]:

	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	Referee	...	HST	AST	HF	AF	H
18	19/08/2018	Man City	Huddersfield	6	1	H	3	1	H	A Marriner	...	14	1	9	4	
52	22/09/2018	Cardiff	Man City	0	5	A	0	3	A	M Oliver	...	2	10	6	4	
77	07/10/2018	Fulham	Arsenal	1	5	A	1	1	D	P Tierney	...	4	7	11	12	
108	04/11/2018	Man City	Southampton	6	1	H	4	1	H	L Mason	...	8	6	14	9	
173	22/12/2018	Cardiff	Man United	1	5	A	1	3	A	M Oliver	...	3	9	13	13	
179	23/12/2018	Everton	Tottenham	2	6	A	1	3	A	P Tierney	...	3	8	13	9	
181	26/12/2018	Burnley	Everton	1	5	A	1	3	A	M Oliver	...	4	6	11	19	
258	10/02/2019	Man City	Chelsea	6	0	H	4	0	H	M Dean	...	9	4	9	13	
261	22/02/2019	Cardiff	Watford	1	5	A	0	1	A	S Hooper	...	6	7	10	11	
326	13/04/2019	Brighton	Bournemouth	0	5	A	0	1	A	K Friend	...	1	7	10	6	

10 rows × 22 columns

In [259…
```
pl.HTHG[(pl.FTHG > 5) | (pl.FTAG > 4)] # Gives HTHG values when either condition is satisfied
```

Out[259]:
```
18     3
52     0
77     1
108    4
173    1
179    1
181    1
258    4
261    0
326    0
Name: HTHG, dtype: int64
```

In [260…
```
pl[(pl.FTHG > 5) | (pl.FTAG > 4)][["HTHG", "HTAG", "Referee"]]

# Gives HTHG, HTAG and Referee values when either condition is satisfied
```

Out[260]:

	HTHG	HTAG	Referee
18	3	1	A Marriner
52	0	3	M Oliver
77	1	1	P Tierney
108	4	1	L Mason
173	1	3	M Oliver
179	1	3	P Tierney
181	1	3	M Oliver
258	4	0	M Dean
261	0	1	S Hooper
326	0	1	K Friend

In [261…
```
print(pl.FTHG != 0)   # Gives True when FTHG is not equal to zero
```

```
0       True
1       True
2      False
3      False
4       True
       ...
375     True
376    False
377     True
378     True
379     True
Name: FTHG, Length: 380, dtype: bool
```

In [262…
```
pl[pl.FTHG != 0]     # Gives all rows where FTHG is not equal to zero
```

Out[262]:

	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	Referee	...	HST	AST	HF	AF
0	10/08/2018	Man United	Leicester	2	1	H	1	0	H	A Marriner	...	6	4	11	8
1	11/08/2018	Bournemouth	Cardiff	2	0	H	1	0	H	K Friend	...	4	1	11	9
4	11/08/2018	Newcastle	Tottenham	1	2	A	1	2	A	M Atkinson	...	2	5	11	12
5	11/08/2018	Watford	Brighton	2	0	H	1	0	H	J Moss	...	5	0	10	16
6	11/08/2018	Wolves	Everton	2	2	D	1	1	D	C Pawson	...	4	5	8	7
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
372	12/05/2019	Crystal Palace	Bournemouth	5	3	H	3	1	H	R East	...	8	8	11	8
375	12/05/2019	Liverpool	Wolves	2	0	H	1	0	H	M Atkinson	...	5	2	3	11
377	12/05/2019	Southampton	Huddersfield	1	1	D	1	0	H	L Probert	...	3	3	8	6
378	12/05/2019	Tottenham	Everton	2	2	D	1	0	H	A Marriner	...	3	9	10	13
379	12/05/2019	Watford	West Ham	1	4	A	0	2	A	C Kavanagh	...	8	9	10	10

292 rows × 22 columns

Transforming DataFrames

Transforming columns

To increase values by one you can use `+=1`

In [263…
```
x=1
print("value of x is", x)
x+=1
print("value increased by 1, x is", x)
```

```
value of x is 1
value increased by 1, x is 2
```

- We can do the same with Dataframes.
- We can add `1` to each of the values in the column `FTHG` where the `HomeTeam` is `Tottenham` and the `Referee` is `'M Dean'`:

In [264…
```
pl.FTHG[(pl.HomeTeam == 'Tottenham') & (pl.Referee == 'M Dean')]
```

Out[264]:
```
75     1
218    0
Name: FTHG, dtype: int64
```

In [298…
```
pl.loc[(pl.HomeTeam == 'Tottenham') & (pl.Referee == 'M Dean'), "FTHG"] += 1 # to increase by one

# pl.FTHG[(pl.HomeTeam == 'Tottenham') & (pl.Referee == 'M Dean')] += 1 # this also works but will se
```

In [266…
```
pl.FTHG[(pl.HomeTeam == 'Tottenham') & (pl.Referee == 'M Dean')]
```

Out[266]:
```
75     2
218    1
Name: FTHG, dtype: int64
```

In [267…
```
# Add the column values in each row to find total goals in each game.
total_goals = pl.FTHG + pl.FTAG
print(total_goals)
```

```
0      3
1      2
2      2
3      3
4      3
      ..
375    2
376    2
377    2
378    4
379    5
Length: 380, dtype: int64
```

In [268…
```
# Create a new column called “total_goals”.
pl["total_goals"] = pl.FTHG + pl.FTAG
display(pl)
```

	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	Referee	…	AST	HF	AF	HC
0	10/08/2018	Man United	Leicester	2	1	H	1	0	H	A Marriner	…	4	11	8	2
1	11/08/2018	Bournemouth	Cardiff	2	0	H	1	0	H	K Friend	…	1	11	9	7
2	11/08/2018	Fulham	Crystal Palace	0	2	A	0	1	A	M Dean	…	9	9	11	5
3	11/08/2018	Huddersfield	Chelsea	0	3	A	0	2	A	C Kavanagh	…	4	9	8	2
4	11/08/2018	Newcastle	Tottenham	1	2	A	1	2	A	M Atkinson	…	5	11	12	3
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
375	12/05/2019	Liverpool	Wolves	2	0	H	1	0	H	M Atkinson	…	2	3	11	4
376	12/05/2019	Man United	Cardiff	0	2	A	0	1	A	J Moss	…	4	9	6	11
377	12/05/2019	Southampton	Huddersfield	1	1	D	1	0	H	L Probert	…	3	8	6	4
378	12/05/2019	Tottenham	Everton	2	2	D	1	0	H	A Marriner	…	9	10	13	7
379	12/05/2019	Watford	West Ham	1	4	A	0	2	A	C Kavanagh	…	9	10	10	7

380 rows × 23 columns

In [269…
```
# You can also perform more math operations and create a new column:
pl["percent_home_goals"] = (pl.FTHG / pl.total_goals) * 100
display(pl) # show new column
```

	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	Referee	...	HF	AF	HC	AC	H
0	10/08/2018	Man United	Leicester	2	1	H	1	0	H	A Marriner	...	11	8	2	5	
1	11/08/2018	Bournemouth	Cardiff	2	0	H	1	0	H	K Friend	...	11	9	7	4	
2	11/08/2018	Fulham	Crystal Palace	0	2	A	0	1	A	M Dean	...	9	11	5	5	
3	11/08/2018	Huddersfield	Chelsea	0	3	A	0	2	A	C Kavanagh	...	9	8	2	5	
4	11/08/2018	Newcastle	Tottenham	1	2	A	1	2	A	M Atkinson	...	11	12	3	5	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
375	12/05/2019	Liverpool	Wolves	2	0	H	1	0	H	M Atkinson	...	3	11	4	1	
376	12/05/2019	Man United	Cardiff	0	2	A	0	1	A	J Moss	...	9	6	11	2	
377	12/05/2019	Southampton	Huddersfield	1	1	D	1	0	H	L Probert	...	8	6	4	3	
378	12/05/2019	Tottenham	Everton	2	2	D	1	0	H	A Marriner	...	10	13	7	4	
379	12/05/2019	Watford	West Ham	1	4	A	0	2	A	C Kavanagh	...	10	10	7	2	

380 rows × 24 columns

Summary Statistics

Details and summary statistics on data frames

- pl is the Premier League DataFrame read in to Python
- `pl.info()` for information on columns
- `pl.shape` for dimensions
- `pl.describe()` for summary statistics
- `pl.mean()` finds mean of each column.
- `pl.values` gives all values in a 2D Numpy array
- `pl.columns` gives column names
- `pl.index` gives row/index names
- `pl.sort_values('AwayTeam', ascending = True)` sorts the dataset by AwayTeam in alphabetical order.
- `pl.sort_values('HomeTeam', ascending = False)` sorts the dataset by HomeTeam in reverse alphabetical order.

In [270…

```
print(pl.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 380 entries, 0 to 379
Data columns (total 24 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Date                380 non-null    object 
 1   HomeTeam            380 non-null    object 
 2   AwayTeam            380 non-null    object 
 3   FTHG                380 non-null    int64  
 4   FTAG                380 non-null    int64  
 5   FTR                 380 non-null    object 
 6   HTHG                380 non-null    int64  
 7   HTAG                380 non-null    int64  
 8   HTR                 380 non-null    object 
 9   Referee             380 non-null    object 
 10  HS                  380 non-null    int64  
 11  AS                  380 non-null    int64  
 12  HST                 380 non-null    int64  
 13  AST                 380 non-null    int64  
 14  HF                  380 non-null    int64  
 15  AF                  380 non-null    int64  
 16  HC                  380 non-null    int64  
 17  AC                  380 non-null    int64  
 18  HY                  380 non-null    int64  
 19  AY                  380 non-null    int64  
 20  HR                  380 non-null    int64  
 21  AR                  380 non-null    int64  
 22  total_goals         380 non-null    int64  
 23  percent_home_goals  358 non-null    float64
dtypes: float64(1), int64(17), object(6)
memory usage: 71.4+ KB
None
```

In [271…

```
print(pl.shape)
```

```
(380, 24)
```

In [272…

```
print(pl.describe())
```

```
             FTHG        FTAG        HTHG        HTAG          HS          AS  \
count  380.000000  380.000000  380.000000  380.000000  380.000000  380.000000
mean     1.573684    1.252632    0.678947    0.573684   14.134211   11.144737
std      1.310539    1.180031    0.860802    0.766958    5.855371    4.654002
min      0.000000    0.000000    0.000000    0.000000    0.000000    2.000000
25%      1.000000    0.000000    0.000000    0.000000   10.000000    8.000000
50%      1.000000    1.000000    0.000000    0.000000   14.000000   11.000000
75%      2.000000    2.000000    1.000000    1.000000   18.000000   14.000000
max      6.000000    6.000000    4.000000    3.000000   36.000000   25.000000

             HST         AST         HF          AF          HC          AC  \
count  380.000000  380.000000  380.000000  380.000000  380.000000  380.000000
mean     4.778947    3.928947   10.152632   10.305263    5.705263    4.552632
std      2.677686    2.283982    3.293532    3.503707    2.971718    2.730627
min      0.000000    0.000000    0.000000    3.000000    0.000000    0.000000
25%      3.000000    2.000000    8.000000    8.000000    4.000000    2.750000
50%      5.000000    4.000000   10.000000   10.000000    5.000000    4.000000
75%      6.000000    5.250000   12.000000   13.000000    8.000000    6.000000
max     14.000000   12.000000   23.000000   21.000000   16.000000   14.000000

              HY          AY          HR          AR  total_goals  \
count  380.000000  380.000000  380.000000  380.000000   380.000000
mean     1.526316    1.684211    0.047368    0.076316     2.826316
std      1.222844    1.209140    0.212706    0.275599     1.596944
min      0.000000    0.000000    0.000000    0.000000     0.000000
25%      1.000000    1.000000    0.000000    0.000000     2.000000
50%      1.000000    2.000000    0.000000    0.000000     3.000000
75%      2.000000    2.000000    0.000000    0.000000     4.000000
max      6.000000    5.000000    1.000000    2.000000     8.000000

       percent_home_goals
count          358.000000
mean            55.666733
std             35.805951
min              0.000000
25%             33.333333
50%             57.142857
75%            100.000000
max            100.000000
```

In [273…
```
pl.HTHG.median()
```

Out[273]:
```
0.0
```

In [274…
```
pl.loc[:,["FTHG","FTAG","HTHG","HTAG"]].mean() # Do not try to get the mean of a column type string,
```

Out[274]:
```
FTHG    1.573684
FTAG    1.252632
HTHG    0.678947
HTAG    0.573684
dtype: float64
```

In [275…
```
print(pl.values)
```

```
[['10/08/2018' 'Man United' 'Leicester' ... 0 3 66.66666666666666]
 ['11/08/2018' 'Bournemouth' 'Cardiff' ... 0 2 100.0]
 ['11/08/2018' 'Fulham' 'Crystal Palace' ... 0 2 0.0]
 ...
 ['12/05/2019' 'Southampton' 'Huddersfield' ... 0 2 50.0]
 ['12/05/2019' 'Tottenham' 'Everton' ... 0 4 50.0]
 ['12/05/2019' 'Watford' 'West Ham' ... 0 5 20.0]]
```

In [276…
```
pl.columns
```

Out[276]:
```
Index(['Date', 'HomeTeam', 'AwayTeam', 'FTHG', 'FTAG', 'FTR', 'HTHG', 'HTAG',
       'HTR', 'Referee', 'HS', 'AS', 'HST', 'AST', 'HF', 'AF', 'HC', 'AC',
       'HY', 'AY', 'HR', 'AR', 'total_goals', 'percent_home_goals'],
      dtype='object')
```

In [277…
```
pl.index
```

Out[277]:
```
RangeIndex(start=0, stop=380, step=1)
```

In [278…
```
pl.sort_values("AwayTeam", ascending = True)
```

Out[278]:

	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	Referee	...	HF	AF	HC	AC	HY
180	26/12/2018	Brighton	Arsenal	1	1	D	1	1	D	A Taylor	...	10	4	4	9	2
283	02/03/2019	Tottenham	Arsenal	1	1	D	0	1	A	A Taylor	...	15	14	3	4	3
169	16/12/2018	Southampton	Arsenal	3	2	H	2	1	H	C Kavanagh	...	12	10	4	5	3
193	29/12/2018	Liverpool	Arsenal	5	1	H	4	1	H	M Oliver	...	8	13	5	3	1
349	24/04/2019	Wolves	Arsenal	3	1	H	3	0	H	S Attwell	...	12	9	5	5	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
263	23/02/2019	Bournemouth	Wolves	1	1	D	1	0	H	R East	...	10	15	9	8	4
158	09/12/2018	Newcastle	Wolves	1	2	A	1	1	D	M Dean	...	10	17	4	6	2
330	13/04/2019	Southampton	Wolves	3	1	H	2	1	H	J Moss	...	13	8	4	9	1
194	29/12/2018	Tottenham	Wolves	1	3	A	1	0	H	S Attwell	...	7	7	6	7	3
245	02/02/2019	Everton	Wolves	1	3	A	1	2	A	L Mason	...	12	14	3	1	3

380 rows × 24 columns

In [279…]:
```
pl.sort_values("HomeTeam", ascending = True)
```

Out[279]:

	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	Referee	...	HF	AF	HC	AC	H
89	22/10/2018	Arsenal	Leicester	3	1	H	1	1	D	C Kavanagh	...	10	10	6	4	
200	01/01/2019	Arsenal	Fulham	4	1	H	1	0	H	G Scott	...	7	12	8	3	
171	22/12/2018	Arsenal	Burnley	3	1	H	1	0	H	K Friend	...	10	14	1	3	
313	01/04/2019	Arsenal	Newcastle	2	0	H	1	0	H	A Taylor	...	11	10	6	2	
58	23/09/2018	Arsenal	Everton	2	0	H	0	0	D	J Moss	...	17	12	5	9	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
349	24/04/2019	Wolves	Arsenal	3	1	H	3	0	H	S Attwell	...	12	9	5	5	
315	02/04/2019	Wolves	Man United	2	1	H	1	1	D	M Dean	...	5	11	3	5	
208	02/01/2019	Wolves	Crystal Palace	0	2	A	0	0	D	R East	...	9	7	3	10	
166	15/12/2018	Wolves	Bournemouth	2	0	H	1	0	H	S Hooper	...	15	7	5	3	
285	02/03/2019	Wolves	Cardiff	2	0	H	2	0	H	A Marriner	...	11	6	7	8	

380 rows × 24 columns

Statistics on columns of data frames

In [280…]:
```
pl.HTHG.mean()
```

Out[280]:
```
0.6789473684210526
```

In [281…]:
```
pl['HTHG'].median()
```

Out[281]:
```
0.0
```

In [282…]:
```
pl['HTHG'].sum()
```

Out[282]:
```
258
```

In [283…]:
```
pl['HTHG'].min()
```

Out[283]:
```
0
```

In [284…
```
pl['HTHG'].max()
```

Out[284]:
```
4
```

In [285…
```
pl['HTHG'].max()

pl[pl.HTHG == pl['HTHG'].max()]
```

Out[285]:

	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	Referee	...	HF	AF	HC	AC	HY
108	04/11/2018	Man City	Southampton	6	1	H	4	1	H	L Mason	...	14	9	4	4	1
193	29/12/2018	Liverpool	Arsenal	5	1	H	4	1	H	M Oliver	...	8	13	5	3	1
258	10/02/2019	Man City	Chelsea	6	0	H	4	0	H	M Dean	...	9	13	2	2	1

3 rows × 24 columns

In [286…
```
pl['HTHG'].quantile(0.75)
```

Out[286]:
```
1.0
```

In [287…
```
pl['HTHG'].quantile(0.5)
```

Out[287]:
```
0.0
```

- In the code below, use a pre-defined function on the 'HTHG' column. We will learn more about functions later.

In [289…
```
#
# pl['HTHG'].agg(function)
```

In [290…
```
pl['HTHG'].cumsum()
```

Out[290]:
```
0        1
1        2
2        2
3        2
4        3
      ...
375    256
376    256
377    257
378    258
379    258
Name: HTHG, Length: 380, dtype: int64
```

In [291…
```
pl['HTHG'].cummax()
```

Out[291]:
```
0      1
1      1
2      1
3      1
4      1
      ..
375    4
376    4
377    4
378    4
379    4
Name: HTHG, Length: 380, dtype: int64
```

In [292…
```
pl['HTHG'].cummin()
```

Out[292]:
```
0      1
1      1
2      0
3      0
4      0
      ..
375    0
376    0
377    0
378    0
379    0
Name: HTHG, Length: 380, dtype: int64
```

In [293…
```
pl[['HTHG','HTAG']].max().max()
```

Out[293]:
```
4
```

In [294…
```
pl[(pl.HTHG == pl[['HTHG','HTAG']].max().max()) | (pl.HTAG == pl[['HTHG','HTAG']].max().max())]
```

Out[294]:

	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	Referee	...	HF	AF	HC	AC	HY
108	04/11/2018	Man City	Southampton	6	1	H	4	1	H	L Mason	...	14	9	4	4	1
193	29/12/2018	Liverpool	Arsenal	5	1	H	4	1	H	M Oliver	...	8	13	5	3	1
258	10/02/2019	Man City	Chelsea	6	0	H	4	0	H	M Dean	...	9	13	2	2	1

3 rows × 24 columns

In [295…
```
pl['HTHG'].cumprod()
```

Out[295]:
```
0      1
1      1
2      0
3      0
4      0
      ..
375    0
376    0
377    0
378    0
379    0
Name: HTHG, Length: 380, dtype: int64
```

Pandas exercises

- Read in the Premier League data from the csv file on Moodle.
- We will not be looking at any of the columns after Referee. Remove the other columns.
- Find all games that were away wins.
- How many away wins were there in the whole season?
- Find the highest number of goals scored by a team at half time.
- Find all games that were refereed by J Moss or M Oliver.
- How many games did Burnley win in the season? Will need to use | to indicate 'or'.
- How many games involving Everton did A Taylor referee?
- How many times did a team losing at half time win the match?
- Find all teams that competed in the league this season.
- Sort the dataset from the highest to lowest number of goals for the home team at full time.
- Sort the dataset from the highest to lowest number of goals for the home team at full time, then highest to lowest number of goals for the away team at full time.

In [ ]:
```
```