Lecture8.2_Hierarchical_Indexing

March 5, 2025

1 Pandas: Multi-indexing / hierarchical indexing

[144]:
```
import pandas as pd
import os
```

[145]:
```
# change current working library
directory = "C:/Users/cepedazk/Jupyter Notebook/Datasets/"
os.chdir(directory)
```

1.1 What is hierarchical indexing, and why is it useful?

- The index of a DataFrame is a series of labels that identify each row.
  - It is used to access or filter data.
  - When you create a DataFrame, a range of index numbers is automatically assigned when no specifying indexes.
- Hierarchical indexing (also known as Multi-indexing) is when your DataFrame has two or more index columns that can be used to rows.
- You can use hierarchical indexing to specify a unique and meaningful identifier for each row
- Multi-indexing is also useful when having hierarchical data:
  - *Hierarchical data* is when items are linked in a *parent-child* relationship in said column.
  - The hierarchical indexes hold parent/child relationships to one another.
  - Examples:
    - `Country → State → City`
    - `Company → Department → Employee`
- Indexes do not follow the rules of 'tidy data' but they make filtering and aggregating the data easier.
- Hierarchical indexing allows us to do useful things like instantly organise our data into groups without performing groupbys.

NOTE: When a column becomes an index, the original "column" is dropped, and an index is added to our DataFrame with the values that were contained When we select and/or aggregate into an index, the level of index we're working against is omitted from the result to avoid being redundant.

1.2 Creating a DataFrame with hierarchical indexing

How to create and name multi-level indexes for a data frame that you specify:

- You can use the parameter `index` when creating a `DataFrame` object.
  - Use `df.index.name` to indicate the name of the indexes (like the name of columns).

  - Assigning a name to the indexes will help you to identify what each level represents.
- You can use `MultiIndex` method to indicate the indexes after creating the `DataFrame` object, or as well within `index`parameter.

1.2.1 Use `index` parameter in `DataFrame`

[146]:
```
df = pd.DataFrame({"data1": [12, 3, 6, 10, 11],
                   "data2": [3, 5, 11, 10, 12]},
                  index=[['a', 'a', 'b', 'b', 'b'], [1, 2, 1, 2, 3]])
display(df)
```

```
     data1  data2
a 1     12      3
  2      3      5
b 1      6     11
  2     10     10
  3     11     12
```

[147]:
```
df.index
```

[147]:
```
MultiIndex([('a', 1),
            ('a', 2),
            ('b', 1),
            ('b', 2),
            ('b', 3)],
           )
```

[148]:
```
# just add the name of the indexes (like the column names)
#  using index.name. As we are sending multiple columns, use list
#  to indicate each name of the indexes. Order matters!

df.index.names = ['group', 'patient']
display(df)
```

```
               data1  data2
group patient
a     1           12      3
      2            3      5
b     1            6     11
      2           10     10
      3           11     12
```

1.2.2 Use `MultiIndex` for multi level indexing

- Use MultiIndex with the following parameters:
  - `levels` parameter refers to the distinct values for each level.
  - `codes` are the integer indices that represent the position of each element in the corresponding level

- `names` represent the name of each level in the multi indexing. This will help to identify each level represents.

[149]:
```
df = pd.DataFrame({"data1": [12, 3, 6, 10],
                   "data2": [3, 5, 11, 10]})

df.index = pd.MultiIndex(levels = [['a', 'b'], [1, 2, 3, 4]],
                         codes = [[0, 0, 1, 1], [0, 1, 2, 3]],
                         names = ['group', 'patient'])

display(df)
```

```
               data1  data2
group patient
a     1           12      3
      2            3      5
b     3            6     11
      4           10     10
```

[150]:
```
df.index
```

[150]:
```
MultiIndex([('a', 1),
            ('a', 2),
            ('b', 3),
            ('b', 4)],
           names=['group', 'patient'])
```

1.3 Example of Hierarchical indexing on a DataFrame

For the remainder of the lecture, we will use the dataset containing data on all games from the Premier League for the 2017/18 and 2018/19 seasons. The main columns we will use are: 1. Season: Whether the game was played in the 2017/18 or 2018/19 season 2. Date: The date on which the game was played 3. HomeTeam: The team that played at home in the match 4. AwayTeam: The team that played away from home in the match 5. FTHG: Full Time Home Goals 6. FTAG: Full Time Away Goals 7. Referee: The referee name

1.3.1 Reading in the dataset in as pl

[151]:
```
# Import data again
pl = pd.read_csv("pl_2seasons.csv")
display(pl.head())
```

```
     Season        Date        HomeTeam      AwayTeam  FTHG  FTAG FTR  HTHG  \
0  20172018  11/08/2017         Arsenal     Leicester     4     3   H     2
1  20172018  12/08/2017        Brighton      Man City     0     2   A     0
2  20172018  12/08/2017         Chelsea       Burnley     2     3   A     0
3  20172018  12/08/2017  Crystal Palace  Huddersfield     0     3   A     0
4  20172018  12/08/2017         Everton         Stoke     1     0   H     1
```

```
   HTAG HTR  … HST  AST  HF  AF  HC  AC  HY  AY  HR  AR
0     2   D  …  10    3   9  12   9   4   0   1   0   0
1     0   D  …   2    4   6   9   3  10   0   2   0   0
2     3   A  …   6    5  16  11   8   5   3   3   2   0
3     2   A  …   4    6   7  19  12   9   1   3   0   0
4     0   H  …   4    1  13  10   6   7   1   1   0   0

[5 rows x 23 columns]
```

1.3.2 Specifying multi-indexes

When you already have an existing columns within your DataFrame that can represent multi-indexes, use the method `.set_index()` and send a list of the name columns.

In the example below, we indicate `Season` and `HomeTeam` are indexes of the dataset Premier League just **for illustration purposes**.

[152]:
```
pl.set_index(['Season', 'HomeTeam'])
```

[152]:
```
                                  Date      AwayTeam  FTHG  FTAG FTR  HTHG  HTAG  \
Season   HomeTeam
20172018 Arsenal          11/08/2017     Leicester     4     3   H     2     2
         Brighton         12/08/2017      Man City     0     2   A     0     0
         Chelsea          12/08/2017       Burnley     2     3   A     0     3
         Crystal Palace   12/08/2017  Huddersfield     0     3   A     0     2
         Everton          12/08/2017         Stoke     1     0   H     1     0
…                                 …             …   …   …   …  ..   …   …
20182019 Liverpool        12/05/2019        Wolves     2     0   H     1     0
         Man United       12/05/2019       Cardiff     0     2   A     0     1
         Southampton      12/05/2019  Huddersfield     1     1   D     1     0
         Tottenham        12/05/2019       Everton     2     2   D     1     0
         Watford          12/05/2019      West Ham     1     4   A     0     2

                           HTR      Referee  HS  …  HST  AST  HF  AF  HC  AC  \
Season   HomeTeam                                …
20172018 Arsenal             D       M Dean  27  …   10    3   9  12   9   4
         Brighton            D     M Oliver   6  …    2    4   6   9   3  10
         Chelsea             A     C Pawson  19  …    6    5  16  11   8   5
         Crystal Palace      A       J Moss  14  …    4    6   7  19  12   9
         Everton             H  N Swarbrick   9  …    4    1  13  10   6   7
…                          ..           …  ..  …  …  …  ..  ..  ..  ..  ..
20182019 Liverpool           H   M Atkinson  13  …    5    2   3  11   4   1
         Man United          A       J Moss  26  …   10    4   9   6  11   2
         Southampton         H    L Probert  10  …    3    3   8   6   4   3
         Tottenham           H   A Marriner  11  …    3    9  10  13   7   4
         Watford             A   C Kavanagh  17  …    8    9  10  10   7   2

                           HY  AY  HR  AR
```

```
Season   HomeTeam
20172018 Arsenal          0   1   0   0
         Brighton         0   2   0   0
         Chelsea          3   3   2   0
         Crystal Palace   1   3   0   0
         Everton          1   1   0   0
…                       ..  ..  ..  ..
20182019 Liverpool        0   2   0   0
         Man United       3   3   0   0
         Southampton      0   1   0   0
         Tottenham        0   2   0   0
         Watford          1   0   1   0

[760 rows x 21 columns]
```

- *What is the difference between the cell above and below?*

[153]:
```
pl.set_index(['Season', 'HomeTeam'], inplace = True)
```

[154]:
```
pl.head()
```

[154]:
```
                                Date      AwayTeam  FTHG  FTAG FTR  HTHG  HTAG  \
Season   HomeTeam
20172018 Arsenal          11/08/2017     Leicester     4     3   H     2     2
         Brighton         12/08/2017      Man City     0     2   A     0     0
         Chelsea          12/08/2017       Burnley     2     3   A     0     3
         Crystal Palace   12/08/2017  Huddersfield     0     3   A     0     2
         Everton          12/08/2017         Stoke     1     0   H     1     0

                        HTR      Referee  HS  …  HST  AST  HF  AF  HC  AC  \
Season   HomeTeam                                …
20172018 Arsenal          D       M Dean  27  …   10    3   9  12   9   4
         Brighton         D     M Oliver   6  …    2    4   6   9   3  10
         Chelsea          A     C Pawson  19  …    6    5  16  11   8   5
         Crystal Palace   A       J Moss  14  …    4    6   7  19  12   9
         Everton          H  N Swarbrick   9  …    4    1  13  10   6   7

                         HY  AY  HR  AR
Season   HomeTeam
20172018 Arsenal          0   1   0   0
         Brighton         0   2   0   0
         Chelsea          3   3   2   0
         Crystal Palace   1   3   0   0
         Everton          1   1   0   0

[5 rows x 21 columns]
```

1.3.3 Sorting multi-indexes

It is almost always a good idea to sort multi-indexes!

The Pandas documentation says: 'Indexing will work even if the data are not sorted, but will be rather inefficient (and show a PerformanceWarning). It will also return a copy of the data rather than a view.'

Use the command `df.sort_index()` to sort the indexes of a DataFrame df.

Again use `inplace = True` to update version of the DataFrame in the Python console.

[155]:
```
display(pl.sort_index().head())

#  Explanation: In this case, the DataFrame is sorted by the "Season" level and returns a copy of the dataframe pl
```

		Date	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	\
Season	HomeTeam									
20172018	Arsenal	11/08/2017	Leicester	4	3	H	2	2	D	
	Arsenal	09/09/2017	Bournemouth	3	0	H	2	0	H	
	Arsenal	25/09/2017	West Brom	2	0	H	1	0	H	
	Arsenal	01/10/2017	Brighton	2	0	H	1	0	H	
	Arsenal	28/10/2017	Swansea	2	1	H	0	1	A	

		Referee	HS	…	HST	AST	HF	AF	HC	AC	HY	AY	HR	AR
Season	HomeTeam			…										
20172018	Arsenal	M Dean	27	…	10	3	9	12	9	4	0	1	0	0
	Arsenal	A Taylor	15	…	8	1	14	8	8	2	0	1	0	0
	Arsenal	R Madley	16	…	6	3	8	17	7	4	1	4	0	0
	Arsenal	K Friend	25	…	12	1	7	8	6	5	0	2	0	0
	Arsenal	L Mason	17	…	5	2	9	9	5	2	0	0	0	0

[5 rows x 21 columns]

[156]:
```
pl.sort_index(inplace = True) # if you want to apply the sorting, use inplace = True
display(pl.head())
```

		Date	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	\
Season	HomeTeam									
20172018	Arsenal	11/08/2017	Leicester	4	3	H	2	2	D	
	Arsenal	09/09/2017	Bournemouth	3	0	H	2	0	H	
	Arsenal	25/09/2017	West Brom	2	0	H	1	0	H	
	Arsenal	01/10/2017	Brighton	2	0	H	1	0	H	
	Arsenal	28/10/2017	Swansea	2	1	H	0	1	A	

		Referee	HS	…	HST	AST	HF	AF	HC	AC	HY	AY	HR	AR
Season	HomeTeam			…										
20172018	Arsenal	M Dean	27	…	10	3	9	12	9	4	0	1	0	0
	Arsenal	A Taylor	15	…	8	1	14	8	8	2	0	1	0	0

```
          Arsenal   R Madley  16  …    6    3   8  17   7   4   1   4   0   0
          Arsenal   K Friend  25  …   12    1   7   8   6   5   0   2   0   0
          Arsenal    L Mason  17  …    5    2   9   9   5   2   0   0   0   0

[5 rows x 21 columns]
```

- What if you want to sort different indexes in different ways? Use the parameter `ascending` and boolean list to indicate `True` for ascending or `False` for descending

[157]:
```
pl.sort_index(level = ['Season', 'HomeTeam'], ascending = [False, False])
```

[157]:
```
                          Date     AwayTeam  FTHG  FTAG FTR  HTHG  HTAG HTR  \
Season   HomeTeam
20182019 Wolves    11/08/2018      Everton     2     2   D     1     1   D
         Wolves    25/08/2018     Man City     1     1   D     0     0   D
         Wolves    16/09/2018      Burnley     1     0   H     0     0   D
         Wolves    29/09/2018  Southampton     2     0   H     0     0   D
         Wolves    20/10/2018      Watford     0     2   A     0     2   A
…                        …          …   …   …  ..   …   …  ..
20172018 Arsenal   11/03/2018      Watford     3     0   H     1     0   H
         Arsenal   01/04/2018        Stoke     3     0   H     0     0   D
         Arsenal   08/04/2018  Southampton     3     2   H     2     1   H
         Arsenal   22/04/2018     West Ham     4     1   H     0     0   D
         Arsenal   06/05/2018      Burnley     5     0   H     2     0   H

                           Referee  HS  …  HST  AST  HF  AF  HC  AC  HY  AY  HR  \
Season   HomeTeam                   …
20182019 Wolves      C Pawson  11  …    4    5   8   7   3   6   0   1   0
         Wolves    M Atkinson  11  …    2    6  13   8   5   9   1   2   0
         Wolves    A Marriner  30  …    7    2  10   9   8   2   2   4   0
         Wolves     S Attwell  14  …    6    6  11   7   8   6   3   1   0
         Wolves        L Mason  10  …    1    3  23  13   8   2   3   1   0
…                        …  ..  …  …  …  ..  ..  ..  ..  ..  ..  ..
20172018 Arsenal   M Atkinson  11  …    7    4  12   9   4   9   2   1   0
         Arsenal     C Pawson  24  …   11    2   9  13   6   5   1   2   0
         Arsenal   A Marriner  13  …    7    8  11   7   8   6   2   1   1
         Arsenal      L Mason  20  …    8    4  11   9   8   6   3   2   0
         Arsenal   A Marriner  16  …    8    2   6   7   4   5   0   1   0

                   AR
Season   HomeTeam
20182019 Wolves     1
         Wolves     0
         Wolves     0
         Wolves     0
         Wolves     0
…                ..
20172018 Arsenal    0
```

```
Arsenal    0
Arsenal    1
Arsenal    0
Arsenal    0

[760 rows x 21 columns]
```

1.3.4 What do multi-indexes look like?

To see the indexes of a DataFrame, use the property `.index` to return a list of indexes. As this is multi-indexes, a list of tuples is returned.

[158]:
```
pl.index
```

[158]:
```
MultiIndex([(20172018, 'Arsenal'),
            (20172018, 'Arsenal'),
            (20172018, 'Arsenal'),
            (20172018, 'Arsenal'),
            (20172018, 'Arsenal'),
            (20172018, 'Arsenal'),
            (20172018, 'Arsenal'),
            (20172018, 'Arsenal'),
            (20172018, 'Arsenal'),
            (20172018, 'Arsenal'),
            …
            (20182019,  'Wolves'),
            (20182019,  'Wolves'),
            (20182019,  'Wolves'),
            (20182019,  'Wolves'),
            (20182019,  'Wolves'),
            (20182019,  'Wolves'),
            (20182019,  'Wolves'),
            (20182019,  'Wolves'),
            (20182019,  'Wolves'),
            (20182019,  'Wolves')],
           names=['Season', 'HomeTeam'], length=760)
```

1.3.5 Choosing hierarchical indexes

Ask yourself the following questions when applying hierarchical indexes to your data: * Have I chosen the best hierarchical indexes for my data? * Can I uniquely select any row based on the indexes used?

Short example:

- Assume that we are trying to find out about the performance of each team at home. We could include `AwayTeam` in the indexing if we want to see the away performance of teams.

What columns would make better choices?

1.3.6 Changing the indexing

To change indexes, we need to use `.reset_index()` to return numerical indexes on the dataset. This makes the multi-indexes become columns. After that, apply again `.set_index()` on the columns you would like.

Exmple on Premier League dataset:

Adding `Date` makes the indexing unique so we can pick out any row uniquely using indexes.

First, we must reset the indexing. Otherwise the old indexing columns will be lost / we will run into problems.

[159]:
```
pl.reset_index(inplace = True) # remove the indexes
display(pl.head())
```

```
     Season HomeTeam        Date     AwayTeam  FTHG  FTAG FTR  HTHG  HTAG HTR  \
0  20172018  Arsenal  11/08/2017    Leicester     4     3   H     2     2   D
1  20172018  Arsenal  09/09/2017  Bournemouth     3     0   H     2     0   H
2  20172018  Arsenal  25/09/2017    West Brom     2     0   H     1     0   H
3  20172018  Arsenal  01/10/2017     Brighton     2     0   H     1     0   H
4  20172018  Arsenal  28/10/2017      Swansea     2     1   H     0     1   A

   … HST  AST  HF  AF  HC  AC  HY  AY  HR  AR
0  …  10    3   9  12   9   4   0   1   0   0
1  …   8    1  14   8   8   2   0   1   0   0
2  …   6    3   8  17   7   4   1   4   0   0
3  …  12    1   7   8   6   5   0   2   0   0
4  …   5    2   9   9   5   2   0   0   0   0

[5 rows x 23 columns]
```

[160]:
```
pl.set_index(['Season', 'HomeTeam', 'Date'], inplace = True) # establish again␣
↪the indexes, selecting the columns you would like for multi-indexing

pl.sort_index(inplace = True) # apply sorting

display(pl.head())
```

```
                                  AwayTeam  FTHG  FTAG FTR  HTHG  HTAG HTR  \
Season   HomeTeam Date
20172018 Arsenal  01/03/2018    Man City     0     3   A     0     3   A
                  01/04/2018       Stoke     3     0   H     0     0   D
                  01/10/2017    Brighton     2     0   H     1     0   H
                  02/12/2017  Man United     1     3   A     0     2   A
                  03/01/2018     Chelsea     2     2   D     0     0   D

                                   Referee  HS  AS  HST  AST  HF  AF  HC  AC  \
Season   HomeTeam Date
20172018 Arsenal  01/03/2018  A Marriner  10   9    5    5  11  11   6   1
                  01/04/2018    C Pawson  24   8   11    2   9  13   6   5
```

```
                    01/10/2017    K Friend  25   9   12    1   7   8   6   5
                    02/12/2017  A Marriner  33   8   15    4  11  10  12   1
                    03/01/2018    A Taylor  14  19    6    6  11  11  10   8

                                       HY  AY  HR  AR
Season   HomeTeam Date
20172018 Arsenal  01/03/2018            1   1   0   0
                  01/04/2018            1   2   0   0
                  01/10/2017            0   2   0   0
                  02/12/2017            3   2   0   1
                  03/01/2018            3   2   0   0
```

[161]:
```
pl.index
```

[161]:
```
MultiIndex([(20172018, 'Arsenal', '01/03/2018'),
            (20172018, 'Arsenal', '01/04/2018'),
            (20172018, 'Arsenal', '01/10/2017'),
            (20172018, 'Arsenal', '02/12/2017'),
            (20172018, 'Arsenal', '03/01/2018'),
            (20172018, 'Arsenal', '03/02/2018'),
            (20172018, 'Arsenal', '06/05/2018'),
            (20172018, 'Arsenal', '08/04/2018'),
            (20172018, 'Arsenal', '09/09/2017'),
            (20172018, 'Arsenal', '11/03/2018'),
            …
            (20182019,  'Wolves', '16/09/2018'),
            (20182019,  'Wolves', '19/01/2019'),
            (20182019,  'Wolves', '20/04/2019'),
            (20182019,  'Wolves', '20/10/2018'),
            (20182019,  'Wolves', '21/12/2018'),
            (20182019,  'Wolves', '24/04/2019'),
            (20182019,  'Wolves', '25/08/2018'),
            (20182019,  'Wolves', '25/11/2018'),
            (20182019,  'Wolves', '29/01/2019'),
            (20182019,  'Wolves', '29/09/2018')],
           names=['Season', 'HomeTeam', 'Date'], length=760)
```

1.3.7 Renaming index levels

Use `.index.names` to establish names of indexes, but also to rename them

If you try the code below, will give an error:

`pl.index.names[1] = ['Home'] gives an error: 'FrozenList' does not support mutable operations.`

All index names must be changed at once because indexes are immutable.

[162]:
```
pl.index.names
```

[162]:
```
FrozenList(['Season', 'HomeTeam', 'Date'])
```

[163]:
```
pl.index.names = ['Season', 'Home', 'Date']

pl.head()
```

[163]:

Season	Home	Date	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR
20172018	Arsenal	01/03/2018	Man City	0	3	A	0	3	A
		01/04/2018	Stoke	3	0	H	0	0	D
		01/10/2017	Brighton	2	0	H	1	0	H
		02/12/2017	Man United	1	3	A	0	2	A
		03/01/2018	Chelsea	2	2	D	0	0	D

Season	Home	Date	Referee	HS	AS	HST	AST	HF	AF	HC	AC	HY
20172018	Arsenal	01/03/2018	A Marriner	10	9	5	5	11	11	6	1	1
		01/04/2018	C Pawson	24	8	11	2	9	13	6	5	1
		01/10/2017	K Friend	25	9	12	1	7	8	6	5	0
		02/12/2017	A Marriner	33	8	15	4	11	10	12	1	3
		03/01/2018	A Taylor	14	19	6	6	11	11	10	8	3

Season	Home	Date	AY	HR	AR
20172018	Arsenal	01/03/2018	1	0	0
		01/04/2018	2	0	0
		01/10/2017	2	0	0
		02/12/2017	2	0	1
		03/01/2018	2	0	0

[164]:
```
# pl.index.names[0] = ['SeasonNumber'] # error
```

1.3.8 Swapping index levels

You can swap the levels of indexes using a command like:

`pl.swaplevel(i = 'HomeTeam', j = 'Season')`

Use `inplace = True` to swap the indexes permanently.

However, it is usually best to reset the indexes and set them again.

[165]:
```
# help(pl.swaplevel)
```

[166]:
```
pl.swaplevel(i = 'Date', j = 'Season')
```

[166]:

Date	Home	Season	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR
01/03/2018	Arsenal	20172018	Man City	0	3	A	0	3	A

```
01/04/2018 Arsenal 20172018         Stoke     3     0   H     0     0   D
01/10/2017 Arsenal 20172018      Brighton     2     0   H     1     0   H
02/12/2017 Arsenal 20172018    Man United     1     3   A     0     2   A
03/01/2018 Arsenal 20172018       Chelsea     2     2   D     0     0   D
…                                   …   …   …  ..   …   …  ..
24/04/2019 Wolves  20182019       Arsenal     3     1   H     3     0   H
25/08/2018 Wolves  20182019      Man City     1     1   D     0     0   D
25/11/2018 Wolves  20182019  Huddersfield     0     2   A     0     1   A
29/01/2019 Wolves  20182019      West Ham     3     0   H     0     0   D
29/09/2018 Wolves  20182019   Southampton     2     0   H     0     0   D

                                 Referee  HS  AS  HST  AST  HF  AF  HC  AC  HY  \
Date       Home    Season
01/03/2018 Arsenal 20172018  A Marriner  10   9    5    5  11  11   6   1   1
01/04/2018 Arsenal 20172018    C Pawson  24   8   11    2   9  13   6   5   1
01/10/2017 Arsenal 20172018    K Friend  25   9   12    1   7   8   6   5   0
02/12/2017 Arsenal 20172018  A Marriner  33   8   15    4  11  10  12   1   3
03/01/2018 Arsenal 20172018    A Taylor  14  19    6    6  11  11  10   8   3
…                                   …  ..  ..  …  …  ..  ..  ..  ..  ..
24/04/2019 Wolves  20182019   S Attwell  11  11    3    1  12   9   5   5   2
25/08/2018 Wolves  20182019  M Atkinson  11  18    2    6  13   8   5   9   1
25/11/2018 Wolves  20182019    K Friend  12  14    3    6   9   8   3   5   1
29/01/2019 Wolves  20182019     D Coote  20   4    9    0   8  10   5   1   4
29/09/2018 Wolves  20182019   S Attwell  14  17    6    6  11   7   8   6   3

                           AY  HR  AR
Date       Home    Season
01/03/2018 Arsenal 20172018   1   0   0
01/04/2018 Arsenal 20172018   2   0   0
01/10/2017 Arsenal 20172018   2   0   0
02/12/2017 Arsenal 20172018   2   0   1
03/01/2018 Arsenal 20172018   2   0   0
…                            ..  ..  ..
24/04/2019 Wolves  20182019   3   0   0
25/08/2018 Wolves  20182019   2   0   0
25/11/2018 Wolves  20182019   2   0   0
29/01/2019 Wolves  20182019   1   0   0
29/09/2018 Wolves  20182019   1   0   0

[760 rows x 20 columns]
```

1.4 Selecting, slicing and aggregating using hierarchical indexing

Now for the fun part!

- When we have our indexes set up, we can use them to select data, slice data, and use `groupby` commands to aggregate data.
- We can use `.loc` to select data in the way you would expect, but slicing does not work in the

way we would expect.
- We can either use the slice command or the `pd.IndexSlice` function.
- .xs (short for cross-section) can be used to specify certain values for indexes.

1.4.1 Selecting data with hierarchical indexing

- What if we extract all values from one season?
- We must use `.loc[]` to select rows by index.
- Since season is the first index column, we can do the following, both of which give all values from the `20172018` season:
  - Nevertheless, you have to watch out the data type of the indexes as well.
  - Best practices (part of Data Management) is to ensure the correct data type on columns
  - For indexes, before using set_index, ensure the data type is correct

[167]:
```
pl.head()
```

[167]:

Season	Home	Date	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR
20172018	Arsenal	01/03/2018	Man City	0	3	A	0	3	A
		01/04/2018	Stoke	3	0	H	0	0	D
		01/10/2017	Brighton	2	0	H	1	0	H
		02/12/2017	Man United	1	3	A	0	2	A
		03/01/2018	Chelsea	2	2	D	0	0	D

Season	Home	Date	Referee	HS	AS	HST	AST	HF	AF	HC	AC	HY
20172018	Arsenal	01/03/2018	A Marriner	10	9	5	5	11	11	6	1	1
		01/04/2018	C Pawson	24	8	11	2	9	13	6	5	1
		01/10/2017	K Friend	25	9	12	1	7	8	6	5	0
		02/12/2017	A Marriner	33	8	15	4	11	10	12	1	3
		03/01/2018	A Taylor	14	19	6	6	11	11	10	8	3

Season	Home	Date	AY	HR	AR
20172018	Arsenal	01/03/2018	1	0	0
		01/04/2018	2	0	0
		01/10/2017	2	0	0
		02/12/2017	2	0	1
		03/01/2018	2	0	0

[168]:
```
# pl.loc["20172018"]  # error

# This works with pl.loc[20172018] as index Season is integer
# If we want select Season as a string with value "20172018", we must convert␣
 ↪the column before set it as index

# We must convert the Season column to contain strings/characters
```

[169]:
```
# Import data again or apply reset_index()
pl = pd.read_csv("pl_2seasons.csv")
```

[170]:
```
pl.dtypes # check out types.

# We need to convert Season and Date to correct types
# Season as a number does not make sense, it is better to keep it as a string,␣
 ↪but you can also leave it as an int
```

[170]:
```
Season       int64
Date        object
HomeTeam    object
AwayTeam    object
FTHG         int64
FTAG         int64
FTR         object
HTHG         int64
HTAG         int64
HTR         object
Referee     object
HS           int64
AS           int64
HST          int64
AST          int64
HF           int64
AF           int64
HC           int64
AC           int64
HY           int64
AY           int64
HR           int64
AR           int64
dtype: object
```

[171]:
```
pl.Season = pl.Season.astype(str)
```

[172]:
```
pl.set_index(['Season', 'HomeTeam', 'Date'], inplace = True) # define the␣
 ↪columns desired as indexes again
```

[173]:
```
pl.sort_index(inplace=True)
```

[174]:
```
pl.loc["20172018"]
```

[174]:

		AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR
HomeTeam	Date							
Arsenal	01/03/2018	Man City	0	3	A	0	3	A
	01/04/2018	Stoke	3	0	H	0	0	D

```
          01/10/2017        Brighton     2     0   H     1     0   H
          02/12/2017      Man United     1     3   A     0     2   A
          03/01/2018         Chelsea     2     2   D     0     0   D
…                                …   …   …  ..   …   …  ..
West Ham  24/11/2017       Leicester     1     1   D     1     1   D
          29/04/2018        Man City     1     4   A     1     2   A
          30/01/2018  Crystal Palace     1     1   D     1     1   D
          30/09/2017         Swansea     1     0   H     0     0   D
          31/03/2018     Southampton     3     0   H     3     0   H

                            Referee  HS  AS  HST  AST  HF  AF  HC  AC  HY  AY  \
HomeTeam Date
Arsenal  01/03/2018    A Marriner  10   9    5    5  11  11   6   1   1   1
         01/04/2018      C Pawson  24   8   11    2   9  13   6   5   1   2
         01/10/2017      K Friend  25   9   12    1   7   8   6   5   0   2
         02/12/2017    A Marriner  33   8   15    4  11  10  12   1   3   2
         03/01/2018      A Taylor  14  19    6    6  11  11  10   8   3   2
…                              …  ..  ..  …  …  ..  ..  ..  ..  ..  ..
West Ham 24/11/2017    M Atkinson   8   7    4    2   9  12   5   5   1   1
         29/04/2018   N Swarbrick   4  19    1    7   5  11   0   7   0   1
         30/01/2018   N Swarbrick   8   9    3    3  11  15   5   3   1   3
         30/09/2017        R East   9   6    4    1  15  13   2   1   3   2
         31/03/2018        J Moss  13   9    5    0   9  16   5   9   0   3

                           HR  AR
HomeTeam Date
Arsenal  01/03/2018   0   0
         01/04/2018   0   0
         01/10/2017   0   0
         02/12/2017   0   1
         03/01/2018   0   0
…                   ..  ..
West Ham 24/11/2017   0   0
         29/04/2018   0   0
         30/01/2018   0   0
         30/09/2017   0   0
         31/03/2018   0   0

[380 rows x 20 columns]
```

1.4.2 Short Exercise

What do you think the following code will return? What columns and rows will show? 1. `pl.loc['20172018', 'Arsenal']` 2. `pl.loc['20172018', 'Arsenal', '11/03/2018']` 3. `pl.loc[:, ['AwayTeam', 'FTHG', 'FTAG']]` 4. `pl.loc[:, ('AwayTeam', 'FTHG', 'FTAG')]` 5. `pl.loc[['20172018', 'Arsenal', '11/03/2018'], ['AwayTeam', 'FTHG', 'FTAG']]` 6. `pl.loc[('20172018', 'Arsenal', '11/03/2018'), ('AwayTeam', 'FTHG', 'FTAG')]`

[175]:
```
# pl.loc['20172018', 'Arsenal'] # does not work
```

[176]:
```
# pl.loc['20172018', 'Arsenal', '11/03/2018'] # does not work
```

[177]:
```
# pl.loc[:, ['AwayTeam', 'FTHG', 'FTAG']] # works
```

[178]:
```
# pl.loc[:, ('AwayTeam', 'FTHG', 'FTAG')] # works
```

[179]:
```
# pl.loc[['20172018', 'Arsenal', '11/03/2018'], ['AwayTeam', 'FTHG', 'FTAG']] #␣
 ↪does not work
```

[180]:
```
# pl.loc[('20172018', 'Arsenal', '11/03/2018'), ['AwayTeam', 'FTHG', 'FTAG']] #␣
 ↪does not work
```

1.4.3 Slicing data in a hierarchical index

If we want all home games for Arsenal across the two seasons, **none** of the following work:

[181]:
```
# display(pl.loc['Arsenal',:])
# display(pl.loc[(:, 'Arsenal')])
# display(pl.loc[:, 'Arsenal', :])
# display(pl.loc["20172018":"20182019", 'Arsenal', :])
```

These commands to find three columns for all Arsenal home games in 20172018 **don't work** either:

[182]:
```
# display(pl.loc[['20172018', 'Arsenal', :], ['AwayTeam', 'FTHG', 'FTAG']])
# display(pl.loc[('20172018', 'Arsenal', :), ('AwayTeam', 'FTHG', 'FTAG')])
```

This is because slicing works slightly differently for hierarchical indexing!

We can either use the slice command `slice()` or the `pd.IndexSlice` function.

IndexSlice function First, we will see the `pd.IndexSlice` function.

Save it as idx for short: `idx = pd.IndexSlice`

`pl.loc[idx['20172018', 'Arsenal', :], ['AwayTeam', 'FTHG', 'FTAG']]`

where for `idx[]`: * Comma (,): Separates the values (or slices) for each index level in MultiIndex. * Colon (:): A slice operator used to select all items from a particular index level.

[189]:
```
idx = pd.IndexSlice # Import IndexSlice

# pl.loc[ row , column ] # remember

pl.loc[idx["20172018", 'Arsenal',:], ['AwayTeam', 'FTHG', 'FTAG']].head() #␣
 ↪works!

# where idx says => select all values of first index is 20172018, for the␣
 ↪second index level equal to Arsenal
#   and for third level, all indexes
```

[189]:
```
                                AwayTeam  FTHG  FTAG
Season   HomeTeam Date
20172018 Arsenal  01/03/2018    Man City     0     3
                  01/04/2018       Stoke     3     0
                  01/10/2017    Brighton     2     0
                  02/12/2017  Man United     1     3
                  03/01/2018     Chelsea     2     2
```

Slicing data using pd.IndexSlice Now we can slice in a similar way to what we have seen before.

For example:

[187]:
```
pl.loc[idx["20172018", 'Arsenal':'Chelsea', :], idx[:'FTAG']].head()
```

[187]:
```
                                AwayTeam  FTHG  FTAG
Season   HomeTeam Date
20172018 Arsenal  01/03/2018    Man City     0     3
                  01/04/2018       Stoke     3     0
                  01/10/2017    Brighton     2     0
                  02/12/2017  Man United     1     3
                  03/01/2018     Chelsea     2     2
```

[188]:
```
pl.loc[idx["20172018", 'Arsenal':'Chelsea', :], idx['FTHG':'HTR']].head()
```

[188]:
```
                              FTHG  FTAG FTR  HTHG  HTAG HTR
Season   HomeTeam Date
20172018 Arsenal  01/03/2018     0     3   A     0     3   A
                  01/04/2018     3     0   H     0     0   D
                  01/10/2017     2     0   H     1     0   H
                  02/12/2017     1     3   A     0     2   A
                  03/01/2018     2     2   D     0     0   D
```

[196]:
```
pl.loc[idx[:, 'Arsenal', '01/03/2018'], idx['FTHG':'FTAG']]
```

[196]:
```
                              FTHG  FTAG
Season   HomeTeam Date
20172018 Arsenal  01/03/2018     0     3
```

[191]:
```
pl.loc[idx[:, 'Arsenal', :], idx[:]].head()

# first idx, all first indexes, second index equal 'Arsenal', and all third␣
  ↪indexes
# second idx[:] is all columns
```

[191]:
```
                                AwayTeam  FTHG  FTAG FTR  HTHG  HTAG HTR  \
Season   HomeTeam Date
20172018 Arsenal  01/03/2018    Man City     0     3   A     0     3   A
```

```
                    01/04/2018       Stoke     3     0   H     0     0   D
                    01/10/2017    Brighton     2     0   H     1     0   H
                    02/12/2017  Man United     1     3   A     0     2   A
                    03/01/2018     Chelsea     2     2   D     0     0   D

                                     Referee  HS  AS  HST  AST  HF  AF  HC  AC  \
Season   HomeTeam Date
20172018 Arsenal  01/03/2018  A Marriner  10   9    5    5  11  11   6   1
                  01/04/2018    C Pawson  24   8   11    2   9  13   6   5
                  01/10/2017    K Friend  25   9   12    1   7   8   6   5
                  02/12/2017  A Marriner  33   8   15    4  11  10  12   1
                  03/01/2018    A Taylor  14  19    6    6  11  11  10   8

                                    HY  AY  HR  AR
Season   HomeTeam Date
20172018 Arsenal  01/03/2018   1   1   0   0
                  01/04/2018   1   2   0   0
                  01/10/2017   0   2   0   0
                  02/12/2017   3   2   0   1
                  03/01/2018   3   2   0   0
```

[195]:
```
pl.loc[idx[:, 'Arsenal', :], :].head()

# this is the same as previous one, but just using ":"
```

[195]:
```
                                AwayTeam  FTHG  FTAG FTR  HTHG  HTAG HTR  \
Season   HomeTeam Date
20172018 Arsenal  01/03/2018    Man City     0     3   A     0     3   A
                  01/04/2018       Stoke     3     0   H     0     0   D
                  01/10/2017    Brighton     2     0   H     1     0   H
                  02/12/2017  Man United     1     3   A     0     2   A
                  03/01/2018     Chelsea     2     2   D     0     0   D

                                     Referee  HS  AS  HST  AST  HF  AF  HC  AC  \
Season   HomeTeam Date
20172018 Arsenal  01/03/2018  A Marriner  10   9    5    5  11  11   6   1
                  01/04/2018    C Pawson  24   8   11    2   9  13   6   5
                  01/10/2017    K Friend  25   9   12    1   7   8   6   5
                  02/12/2017  A Marriner  33   8   15    4  11  10  12   1
                  03/01/2018    A Taylor  14  19    6    6  11  11  10   8

                                    HY  AY  HR  AR
Season   HomeTeam Date
20172018 Arsenal  01/03/2018   1   1   0   0
                  01/04/2018   1   2   0   0
                  01/10/2017   0   2   0   0
                  02/12/2017   3   2   0   1
```

```
                    03/01/2018   3   2   0   0
```

[193]:
```
pl.loc[idx[:, :, '02/12/2018'], :]
```

[193]:
```
                                AwayTeam  FTHG  FTAG FTR  HTHG  HTAG HTR  \
Season   HomeTeam  Date
20182019 Arsenal   02/12/2018  Tottenham     4     2   H     1     2   A
         Chelsea   02/12/2018     Fulham     2     0   H     1     0   H
         Liverpool 02/12/2018    Everton     1     0   H     0     0   D

                                  Referee  HS  AS  HST  AST  HF  AF  HC  AC  \
Season   HomeTeam  Date
20182019 Arsenal   02/12/2018      M Dean  22  11    7    6  15  17   8   5
         Chelsea   02/12/2018    C Pawson  16   9    9    4   8  17   4   6
         Liverpool 02/12/2018  C Kavanagh  16   9    3    3  12   7   8   1

                               HY  AY  HR  AR
Season   HomeTeam  Date
20182019 Arsenal   02/12/2018   3   3   0   1
         Chelsea   02/12/2018   2   1   0   0
         Liverpool 02/12/2018   3   2   0   0
```

Slicing data using `slice()` The `slice()` function returns a slice object, which can be passed into the `.loc[]` indexers to select a subset of the DataFrame.

Notice below that the slices appear in round brackets, and the start and end of the slice are separated by a comma rather than a colon.

Instead of a colon (:) to indicate that we use all values of a certain index, we use `slice(None)`.

For example:

```
pl.loc[(slice(None), slice('Arsenal','Liverpool'), '28/10/2017'), :]
```

[197]:
```
# Import data again
pl.loc[(slice(None), slice('Arsenal','Liverpool'), '28/10/2017'), :]

# pl.loc[idx[:, Arsenal':'Liverpool', '28/10/2017'], :]
```

[197]:
```
                                      AwayTeam  FTHG  FTAG FTR  HTHG  HTAG  \
Season   HomeTeam       Date
20172018 Arsenal        28/10/2017     Swansea     2     1   H     0     1
         Bournemouth    28/10/2017     Chelsea     0     1   A     0     0
         Crystal Palace 28/10/2017   West Ham     2     2   D     0     2
         Liverpool      28/10/2017  Huddersfield     3     0   H     0     0

                                     HTR  Referee  HS  AS  HST  AST  HF  AF  \
Season   HomeTeam       Date
20172018 Arsenal        28/10/2017   A  L Mason  17   4    5    2   9   9
```

```
          Bournemouth    28/10/2017   D  C Pawson   7  18    1    5   5   5
          Crystal Palace 28/10/2017   A  R Madley  19   6    9    2  18  15
          Liverpool      28/10/2017   D  K Friend  16   1    8    0  10   8

                                     HC  AC  HY  AY  HR  AR
Season   HomeTeam       Date
20172018 Arsenal        28/10/2017   5   2   0   0   0   0
         Bournemouth    28/10/2017   5   5   2   0   0   0
         Crystal Palace 28/10/2017  11   2   1   2   0   0
         Liverpool      28/10/2017   9   2   0   1   0   0
```

Slicing on dates What if we try to slice on dates?

This command does not work because the dates are not in the correct Python format:

```
pl.loc[(slice(None), slice('Arsenal','Liverpool'), slice('2017-10-28', '2017-11-06')), :]
```

The command above gives an empty DataFrame.

[253]:
```
pl = pd.read_csv("pl_2seasons.csv")
pl.set_index(['Season','HomeTeam','Date'], inplace=True)
#pl.sort_index(level = ['Date', 'HomeTeam', 'Season'], inplace=True)
pl.sort_index(inplace=True)
#pl.sort_index(level = ['Date', 'HomeTeam', 'Season'], inplace=True)
```

[254]:
```
# help(pl.sort_index)
```

[255]:
```
pl.loc[(slice(None), slice('Arsenal','Liverpool'), slice('2017-10-28',␣
  ↪'2017-11-06')), :] # but this does not work, why?
```

[255]:
```
Empty DataFrame
Columns: [AwayTeam, FTHG, FTAG, FTR, HTHG, HTAG, HTR, Referee, HS, AS, HST, AST,
HF, AF, HC, AC, HY, AY, HR, AR]
Index: []
```

Dates in Python

- Python dates should be in the form `YYYY-MM-DD`, and they should be a datetime object so that `Python` recognises them as dates.
- For example, `2022-10-18` is the correct format for today's date.
- These objects can also contain times.
- We use the function `pd.to_datetime` to convert the date into the correct format.
- Use the format argument of `pd.to_datetime` to tell it what format your date is in before it converts it into the form `YYYY-MM-DD`.

For the Premier League dataset

- We **cannot** convert the dates when they are an index column.
- We must reset the index, change the date, and set the indexing again.

[381]:
```
pl.reset_index(inplace=True) # remove columns from indexing
pl.Date = pd.to_datetime(pl.Date, format='%d/%m/%Y') # convert date to datetime
pl.Season = pl.Season.astype(str) # convert season column as well
```

[274]:
```
pl.dtypes
```

[274]:
```
Season              object
HomeTeam            object
Date        datetime64[ns]
AwayTeam            object
FTHG                 int64
FTAG                 int64
FTR                 object
HTHG                 int64
HTAG                 int64
HTR                 object
Referee             object
HS                   int64
AS                   int64
HST                  int64
AST                  int64
HF                   int64
AF                   int64
HC                   int64
AC                   int64
HY                   int64
AY                   int64
HR                   int64
AR                   int64
dtype: object
```

[382]:
```
pl.set_index(['Season','HomeTeam','Date'], inplace=True) # return columns to␣
↪indexes
```

[383]:
```
pl.sort_index(inplace=True)
```

[384]:
```
pl.loc[(slice(None), slice('Arsenal','Liverpool'), slice('2017-10-28',␣
↪'2017-11-06')), :]
```

[384]:

			FTR	AwayTeam	FTHG	FTAG	HTHG	HTAG	\
Season	HomeTeam	Date							
20172018	Arsenal	2017-10-28	H	Swansea	2	1	0	1	
	Bournemouth	2017-10-28	A	Chelsea	0	1	0	0	
	Brighton	2017-10-29	D	Southampton	1	1	0	1	
	Burnley	2017-10-30	H	Newcastle	1	0	0	0	
	Chelsea	2017-11-05	H	Man United	1	0	0	0	
	Crystal Palace	2017-10-28	D	West Ham	2	2	0	2	

```
          Everton        2017-11-05   H       Watford     3     2     0     0
          Huddersfield   2017-11-04   H     West Brom     1     0     1     0
          Leicester      2017-10-29   H       Everton     2     0     2     0
          Liverpool      2017-10-28   H  Huddersfield     3     0     0     0

                                    HTR      Referee  HS  AS  HST  AST  HF  AF  \
Season   HomeTeam       Date
20172018 Arsenal        2017-10-28   A      L Mason  17   4    5    2   9   9
         Bournemouth    2017-10-28   D     C Pawson   7  18    1    5   5   5
         Brighton       2017-10-29   A  N Swarbrick   7   6    2    1   9  10
         Burnley        2017-10-30   D       M Dean  12  12    5    5   9  10
         Chelsea        2017-11-05   D     A Taylor  18  10    8    2  16  20
         Crystal Palace 2017-10-28   A     R Madley  19   6    9    2  18  15
         Everton        2017-11-05   D      G Scott  10  11    5    3  14  12
         Huddersfield   2017-11-04   H       R East   7   9    3    3   8  15
         Leicester      2017-10-29   H   A Marriner   9  16    3    2   6  10
         Liverpool      2017-10-28   D     K Friend  16   1    8    0  10   8

                                    HC  AC  HY  AY  HR  AR
Season   HomeTeam       Date
20172018 Arsenal        2017-10-28   5   2   0   0   0   0
         Bournemouth    2017-10-28   5   5   2   0   0   0
         Brighton       2017-10-29   2   7   3   1   0   0
         Burnley        2017-10-30   5   3   3   0   0   0
         Chelsea        2017-11-05   4   7   1   3   0   0
         Crystal Palace 2017-10-28  11   2   1   2   0   0
         Everton        2017-11-05   2   5   0   2   0   0
         Huddersfield   2017-11-04   3   5   1   4   1   0
         Leicester      2017-10-29   3  10   0   2   0   0
         Liverpool      2017-10-28   9   2   0   1   0   0
```

Slicing with dates

- When our dates are in the correct Python format, we can do all sorts of neat tricks.
- For example if we specify '2017' in our slice for the Date index, Python will know we want all dates from the year 2017.
- Similarly, we can specify a slice of month-year combinations and Python will give us all rows from the start month-year to the end month-year inclusive.
- We will see two examples of these next.

[ ]:
```
# filter by year using slice
pl.loc[(slice(None), slice('Arsenal','Liverpool'), slice('2017')), :].head()
```

[ ]:
```
# filter by year-month
pl.loc[(slice(None), slice('Arsenal','Liverpool'), slice('2017-01','2017-10')),␣
↪:].head()
```

1.4.4 Using .xs to select using multi-indexes

`df.xs()` can be used to cross-section the data. This can make it easier to access certain slices.

[300]:
```
pl.xs("20182019").head()
# Gives all rows that are from season '20182019'
```

[300]:

HomeTeam	Date	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	Referee	\
Arsenal	2018-08-12	Man City	0	2	A	0	1	A	M Oliver	
	2018-08-25	West Ham	3	1	H	1	1	D	G Scott	
	2018-09-23	Everton	2	0	H	0	0	D	J Moss	
	2018-09-29	Watford	2	0	H	0	0	D	A Taylor	
	2018-10-22	Leicester	3	1	H	1	1	D	C Kavanagh	

HomeTeam	Date	HS	AS	HST	AST	HF	AF	HC	AC	HY	AY	HR	AR
Arsenal	2018-08-12	9	17	3	8	11	14	2	9	2	2	0	0
	2018-08-25	17	13	10	5	16	13	10	2	1	3	0	0
	2018-09-23	9	9	5	6	17	12	5	9	2	1	0	0
	2018-09-29	9	13	2	4	11	17	6	6	2	2	0	0
	2018-10-22	19	8	6	2	10	10	6	4	2	2	0	0

- The command below does not work. Why?

[ ]:
```
# pl.xs('Chelsea') # error
```

- We must specify the level. This can be done by index `level` number or name.
- Use the parameter `level` in function `.xs()` to indicate the index level to slice
- Notice again that the specified level disappears.

[302]:
```
pl.xs('Chelsea', level = 1).head()

# level = 1 indicates that you are slicing by "HomeTeam"
# this is the same if you write:
#  pl.xs('Chelsea', level = 'HomeTeam').head()
```

[302]:

Season	Date	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	Referee	HS	\
20172018	2017-08-12	Burnley	2	3	A	0	3	A	C Pawson	19	
	2017-08-27	Everton	2	0	H	2	0	H	J Moss	18	
	2017-09-17	Arsenal	0	0	D	0	0	D	M Oliver	13	
	2017-09-30	Man City	0	1	A	0	0	D	M Atkinson	4	
	2017-10-21	Watford	4	2	H	1	1	D	J Moss	14	

Season	Date	AS	HST	AST	HF	AF	HC	AC	HY	AY	HR	AR
20172018	2017-08-12	10	6	5	16	11	8	5	3	3	2	0
	2017-08-27	7	7	0	12	7	6	3	2	2	0	0

```
2017-09-17  11    4    2  11  15   5   1   1   3   1   0
2017-09-30  17    2    6   8  13   4   8   0   2   0   0
2017-10-21  16    8    5  13  16   6   6   2   3   0   0
```

[308]:
```
pl.xs('Chelsea', level = 'HomeTeam').head()
```

[308]:
```
                    AwayTeam  FTHG  FTAG FTR  HTHG  HTAG HTR     Referee  HS  \
Season   Date
20172018 2017-08-12   Burnley     2     3   A     0     3   A    C Pawson  19
         2017-08-27   Everton     2     0   H     2     0   H      J Moss  18
         2017-09-17   Arsenal     0     0   D     0     0   D    M Oliver  13
         2017-09-30  Man City     0     1   A     0     0   D  M Atkinson   4
         2017-10-21   Watford     4     2   H     1     1   D      J Moss  14

                    AS  HST  AST  HF  AF  HC  AC  HY  AY  HR  AR
Season   Date
20172018 2017-08-12  10    6    5  16  11   8   5   3   3   2   0
         2017-08-27   7    7    0  12   7   6   3   2   2   0   0
         2017-09-17  11    4    2  11  15   5   1   1   3   1   0
         2017-09-30  17    2    6   8  13   4   8   0   2   0   0
         2017-10-21  16    8    5  13  16   6   6   2   3   0   0
```

- What do you think the code below will show?

[ ]:
```
# help(pl.xs)
```

[304]:
```
pl.xs('2018-08-18', level = "Date")
```

[304]:
```
                      AwayTeam  FTHG  FTAG FTR  HTHG  HTAG HTR     Referee  \
Season   HomeTeam
20182019 Cardiff       Newcastle     0     0   D     0     0   D    C Pawson
         Chelsea         Arsenal     3     2   H     2     2   D  M Atkinson
         Everton     Southampton     2     1   H     2     0   H     L Mason
         Leicester       Wolves     2     0   H     2     0   H      M Dean
         Tottenham       Fulham     3     1   H     1     0   H    A Taylor
         West Ham    Bournemouth     1     2   A     1     0   H   S Attwell

                     HS  AS  HST  AST  HF  AF  HC  AC  HY  AY  HR  AR
Season   HomeTeam
20182019 Cardiff     12  12    1    6  14  16   5   5   2   2   0   1
         Chelsea     24  15   11    6  12   9   5   1   0   2   0   0
         Everton     13  15    7    4   8  20   2   5   0   5   0   0
         Leicester    6  11    2    3  10   8   1   9   2   1   1   0
         Tottenham   25  10   11    3   9   5   5   2   0   0   0   0
         West Ham    11  12    5    5  14  10   6   4   6   2   0   0
```

- What do you think the code below will do?

[305]:
```
pl.xs(('Chelsea', '2018-08-18'), level = ['HomeTeam','Date'])
```

[305]:
```
          AwayTeam  FTHG  FTAG FTR  HTHG  HTAG HTR     Referee  HS  AS  HST  \
Season
20182019  Arsenal     3     2   H     2     2   D  M Atkinson  24  15   11

          AST  HF  AF  HC  AC  HY  AY  HR  AR
Season
20182019    6  12   9   5   1   0   2   0   0
```

[306]:
```
pl.loc[(slice(None), slice('Chelsea'), slice('2018-08-12','2018-08-30')), :]
```

[306]:
```
                                  AwayTeam  FTHG  FTAG FTR  HTHG  HTAG HTR  \
Season   HomeTeam    Date
20182019 Arsenal     2018-08-12    Man City     0     2   A     0     1   A
                     2018-08-25    West Ham     3     1   H     1     1   D
         Bournemouth 2018-08-25     Everton     2     2   D     0     0   D
         Brighton    2018-08-19  Man United     3     2   H     3     1   H
         Burnley     2018-08-19     Watford     1     3   A     1     1   D
         Cardiff     2018-08-18   Newcastle     0     0   D     0     0   D
         Chelsea     2018-08-18     Arsenal     3     2   H     2     2   D

                                    Referee  HS  AS  HST  AST  HF  AF  HC  AC  \
Season   HomeTeam    Date
20182019 Arsenal     2018-08-12    M Oliver   9  17    3    8  11  14   2   9
                     2018-08-25     G Scott  17  13   10    5  16  13  10   2
         Bournemouth 2018-08-25   L Probert  17  11    5    3  12  10   6   2
         Brighton    2018-08-19    K Friend   6   9    3    3  16  13   3   5
         Burnley     2018-08-19   P Tierney   8   9    3    6   8  19   5   2
         Cardiff     2018-08-18    C Pawson  12  12    1    6  14  16   5   5
         Chelsea     2018-08-18  M Atkinson  24  15   11    6  12   9   5   1

                                  HY  AY  HR  AR
Season   HomeTeam    Date
20182019 Arsenal     2018-08-12   2   2   0   0
                     2018-08-25   1   3   0   0
         Bournemouth 2018-08-25   0   3   1   1
         Brighton    2018-08-19   1   1   0   0
         Burnley     2018-08-19   1   2   0   0
         Cardiff     2018-08-18   2   2   0   1
         Chelsea     2018-08-18   0   2   0   0
```

1.4.5 Using .xs to select using multi-indexes

Remember earlier we had problems using .loc to find all home games for Arsenal across the two seasons?

Now we can simply use the command: `pl.xs('Arsenal', level = 1)`

[309]:
```
pl.xs('Arsenal', level = 1).head() # now you can select a specific index level
```

[309]:

Season	Date	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	Referee
20172018	2017-08-11	Leicester	4	3	H	2	2	D	M Dean
	2017-09-09	Bournemouth	3	0	H	2	0	H	A Taylor
	2017-09-25	West Brom	2	0	H	1	0	H	R Madley
	2017-10-01	Brighton	2	0	H	1	0	H	K Friend
	2017-10-28	Swansea	2	1	H	0	1	A	L Mason

Season	Date	HS	AS	HST	AST	HF	AF	HC	AC	HY	AY	HR	AR
20172018	2017-08-11	27	6	10	3	9	12	9	4	0	1	0	0
	2017-09-09	15	6	8	1	14	8	8	2	0	1	0	0
	2017-09-25	16	7	6	3	8	17	7	4	1	4	0	0
	2017-10-01	25	9	12	1	7	8	6	5	0	2	0	0
	2017-10-28	17	4	5	2	9	9	5	2	0	0	0	0

1.5 Aggregating data using hierarchical indexing

- We can use `.groupby()` in conjunction with hierarchical indexing to quickly find summary statistics from our data.
- Use the parameter `level` to indicate the index level to use to apply a aggregation function on a specific group.
- For example, how many goals did each team score at home in the 20182019 season?

[339]:
```
season_1819 = pl.xs('20182019')
season_1819.groupby(level='HomeTeam')[['FTHG', 'FTAG','HTHG','HTAG']].sum()
```

[339]:

HomeTeam	FTHG	FTAG	HTHG	HTAG
Arsenal	42	16	14	8
Bournemouth	30	25	15	10
Brighton	19	28	10	12
Burnley	24	32	14	12
Cardiff	21	38	8	18
Chelsea	39	12	16	6
Crystal Palace	19	23	8	6
Everton	30	21	14	11
Fulham	22	36	9	20
Huddersfield	10	31	4	18
Leicester	24	20	10	12
Liverpool	55	10	26	5
Man City	57	12	28	7
Man United	33	25	16	7
Newcastle	24	25	10	12
Southampton	27	30	14	13
Tottenham	34	16	13	5

```
Watford           26    28     7    14
West Ham          32    27    12    13
Wolves            28    21    10     9
```

Suppose we are asked: 'How many times did Arsenal win, lose and draw at home in each of the two seasons?'

The FTR column indicates H for a home win, A for an away win and D for a draw. It will be useful for this question.

[348]:
```
pl.xs('Arsenal', level = "HomeTeam").groupby(level='Season')['FTR'].
  ↪value_counts()
```

[348]:
```
Season    FTR
20172018  H      15
          A       2
          D       2
20182019  H      14
          D       3
          A       2
Name: count, dtype: int64
```

- You could also do this by grouping by FTR and finding the number in each group.
- To do this you would change the indexing to have Season, HomeTeam and FTR in the indexing, since they are the 3 variables we will be slicing / grouping over.
- The `.size()` function returns the number of rows in each group as a Series.

[356]:
```
# change index levels
pl.reset_index(inplace = True)
pl.set_index(['Season', 'HomeTeam', 'FTR'], inplace = True)

# sort
pl.sort_index(inplace = True)

# select level HomeTeam Arsenal, and then group by season and full time results
#  function size will return
pl.xs('Arsenal', level = 1).groupby(['Season','FTR']).size()
```

[356]:
```
Season    FTR
20172018  A       2
          D       2
          H      15
20182019  A       2
          D       3
          H      14
dtype: int64
```

1.6 Stacking and unstacking indexes for *reshaping*

- `df.unstack()` removes index levels.
  - Returns a DataFrame having a new level of column labels whose inner-most level consists of the pivoted index labels.
- `df.unstack(level = 0)` separates each column into separate columns for each value of index level 0.
- `df.unstack(level = [0,1])` separates each column into separate columns for each value of index level 0 and each value of index level 1.
- `df.stack()` stacks every value on top of each other to make all of the data appear in one tall DataFrame. Both of the following commands do this:
  - `df.stack(level = 0)`
  - `df.stack()`

These commands are complicated for the pl data, so we will go back to the simple df we created earlier.

[370]:
```
# lets create another dataframe
df = pd.DataFrame({"2018": [12, 3, 6, 10],
                   "2017": [3, 5, 11, 10]})

df.index = pd.MultiIndex(levels=[['a', 'b'], [1, 2]],
                         codes=[[0, 0, 1, 1], [0, 1, 0, 1]],
                         names = ['group', 'patient'])

df
```

[370]:

		2018	2017
group	patient		
a	1	12	3
	2	3	5
b	1	6	11
	2	10	10

[371]:
```
df.unstack(level = 0)
```

[371]:

	2018		2017	
group	a	b	a	b
patient				
1	12	6	3	11
2	3	10	5	10

[364]:
```
df.unstack(level = [1])
```

[364]:

	2018		2017	
patient	1	2	1	2
group				
a	12	3	3	5
b	6	10	11	10

[374]:
```
# we have years as column names, this is not a best practice!
# reshape using stack, easy when setting indexes
# values in indexes are kept

df.stack()
```

[374]:
```
group  patient
a      1        2018    12
                2017     3
       2        2018     3
                2017     5
b      1        2018     6
                2017    11
       2        2018    10
                2017    10
dtype: int64
```

[378]:
```
# you can return indexes as columns using reset_index()
df2 = df.stack().reset_index()
df2.columns = ['group','patient','year','sleep_time']
df2
```

[378]:

	group	patient	year	sleep_time
0	a	1	2018	12
1	a	1	2017	3
2	a	2	2018	3
3	a	2	2017	5
4	b	1	2018	6
5	b	1	2017	11
6	b	2	2018	10
7	b	2	2017	10

[389]:
```
# If we do not use indexing, all columns and values are stacked
df3 = pd.DataFrame({"2018": [12, 3, 6, 10],
                    "2017": [3, 5, 11, 10],
                    "group":['a', 'a', 'b', 'b'],
                    "patient":[1, 2, 1, 2]})

df3.stack()
```

[389]:
```
0  2018       12
   2017        3
   group       a
   patient     1
1  2018        3
   2017        5
   group       a
```

```
   patient     2
2  2018        6
   2017       11
   group       b
   patient     1
3  2018       10
   2017       10
   group       b
   patient     2
dtype: object
```

1.7 Short Exercises

Create a Multi-indexing Dataframe:

- Read in the pl_2seasons csv file from Moodle
- Change the date format to the correct Python date format
- Make sure the entries in the Season and AwayTeam columns are strings
- Set up multi-indexing with indexes `['Season', 'AwayTeam', 'Date']`
- Remember to sort the indexes using `sort_index`

Filtering and Slicing: *Tip: Use `pd.IndexSlice` or `slice()`* * Find all away games that Liverpool played in the 20172018 season * Find all away games that Liverpool played in either of the 2 seasons * Find all away games that were played by Brighton and Chelsea between 2019-02-04 and 2019-04-04

Using aggregation on Multi-indexing: *Tip: You can use `.xs()` and `groupby(level=)`* * Which team scored the most away goals in the 20172018 season? * Which team won the most away games over the 2 seasons? * Which referee officiated over the most games in the 20182019 season? * How many goals were scored in the Premier League on 2017-10-21?