Position and Label Based Indexing: `df.iloc` and `df.loc`

You have seen some ways of selecting rows and columns from dataframes. Let's now see some other ways of indexing dataframes, which pandas recommends, since they are more explicit (and less ambiguous).

There are two main ways of indexing dataframes:

1. Position based indexing using `df.iloc`
2. Label based indexing using `df.loc`

Using both the methods, we will do the following indexing operations on a dataframe:

- Selecting single elements/cells
- Selecting single and multiple rows
- Selecting single and multiple columns
- Selecting multiple rows and columns

In [1]:
```
# Loading libraries and reading the data
import numpy as np
import pandas as pd

market_df = pd.read_csv("../global_sales_data/market_fact.csv")
market_df.head()
```

Out[1]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01	23	-30.51
1	Ord_5406	Prod_13	SHP_7549	Cust_1818	42.27	0.01	13	4.56
2	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.69	0.00	26	1148.90
3	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.89	0.09	43	729.34
4	Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.15	0.08	35	1219.87

Position (Integer) Based Indexing

Pandas provides the `df.iloc` functionality to index dataframes **using integer indices**.

```
In [2]: help(pd.DataFrame.iloc)
```

```
Help on property:

    Purely integer-location based indexing for selection by position.

    ``.iloc[]`` is primarily integer position based (from ``0`` to
    ``length-1`` of the axis), but may also be used with a boolean
    array.

    Allowed inputs are:

    - An integer, e.g. ``5``.
    - A list or array of integers, e.g. ``[4, 3, 0]``.
    - A slice object with ints, e.g. ``1:7``.
    - A boolean array.
    - A ``callable`` function with one argument (the calling Series, DataFram
e
      or Panel) and that returns valid output for indexing (one of the above)

    ``.iloc`` will raise ``IndexError`` if a requested indexer is
    out-of-bounds, except *slice* indexers which allow out-of-bounds
    indexing (this conforms with python/numpy *slice* semantics).

    See more at :ref:`Selection by Position <indexing.integer>`
```

As mentioned in the documentation, the inputs x, y to `df.iloc[x, y]` can be:

- An integer, e.g. `3`
- A list or array of integers, e.g. `[3, 7, 8]`
- An integer range, i.e. `3:8`
- A boolean array

Let's see some examples.

```
In [3]: # Selecting a single element
        # Note that 2, 4 corresponds to the third row and fifth column (Sales)
        market_df.iloc[2, 4]
```

```
Out[3]: 4701.6899999999996
```

Note that simply writing `df[2, 4]` will throw an error, since pandas gets confused whether the 2 is an integer index (the third row), or is it a row with label = 2?

On the other hand, `df.iloc[2, 4]` tells pandas explicitly that it should assume **integer indices**.

In [4]:
```
# Selecting a single row, and all columns
# Select the 6th row, with label (and index) = 5
market_df.iloc[5]
```

Out[4]:
```
Ord_id                  Ord_5446
Prod_id                   Prod_6
Ship_id                 SHP_7608
Cust_id                Cust_1818
Sales                     164.02
Discount                    0.03
Order_Quantity                23
Profit                    -47.64
Shipping_Cost               6.15
Product_Base_Margin         0.37
Name: 5, dtype: object
```

In [5]:
```
# The above is equivalent to this
# The ":" indicates "all rows/columns"
market_df.iloc[5, :]

# equivalent to market_df.iloc[5, ]
```

Out[5]:
```
Ord_id                  Ord_5446
Prod_id                   Prod_6
Ship_id                 SHP_7608
Cust_id                Cust_1818
Sales                     164.02
Discount                    0.03
Order_Quantity                23
Profit                    -47.64
Shipping_Cost               6.15
Product_Base_Margin         0.37
Name: 5, dtype: object
```

In [6]:
```
# Select multiple rows using a list of indices
market_df.iloc[[3, 7, 8]]
```

Out[6]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Pro
3	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.8900	0.09	43	729.3
7	Ord_4725	Prod_4	SHP_6593	Cust_1641	3410.1575	0.10	48	1137.
8	Ord_4725	Prod_13	SHP_6593	Cust_1641	162.0000	0.01	33	45.84

In [7]:
```
# Equivalently, you can use:
market_df.iloc[[3, 7, 8], :]

# same as market_df.iloc[[3, 7, 8], ]
```

Out[7]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Pro
3	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.8900	0.09	43	729.3
7	Ord_4725	Prod_4	SHP_6593	Cust_1641	3410.1575	0.10	48	1137.
8	Ord_4725	Prod_13	SHP_6593	Cust_1641	162.0000	0.01	33	45.84

In [8]:
```
# Selecting rows using a range of integer indices
# Notice that 4 is included, 8 is not
market_df.iloc[4:8]
```

Out[8]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Pro
4	Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.1500	0.08	35	1219.
5	Ord_5446	Prod_6	SHP_7608	Cust_1818	164.0200	0.03	23	-47.64
6	Ord_31	Prod_12	SHP_41	Cust_26	14.7600	0.01	5	1.32
7	Ord_4725	Prod_4	SHP_6593	Cust_1641	3410.1575	0.10	48	1137.

In [9]:
```
# or equivalently
market_df.iloc[4:8, :]

# or market_df.iloc[4:8, ]
```

Out[9]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Pro
4	Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.1500	0.08	35	1219.
5	Ord_5446	Prod_6	SHP_7608	Cust_1818	164.0200	0.03	23	-47.64
6	Ord_31	Prod_12	SHP_41	Cust_26	14.7600	0.01	5	1.32
7	Ord_4725	Prod_4	SHP_6593	Cust_1641	3410.1575	0.10	48	1137.

```
In [10]: # Selecting a single column
         # Notice that the column index starts at 0, and 2 represents the third column
          (Cust_id)
         market_df.iloc[:, 2]
```

```
Out[10]: 0       SHP_7609
         1       SHP_7549
         2       SHP_7610
         3       SHP_7625
         4       SHP_7664
         5       SHP_7608
         6         SHP_41
         7       SHP_6593
         8       SHP_6593
         9       SHP_6593
         10      SHP_6615
         11      SHP_2637
         12      SHP_4112
         13      SHP_3093
         14      SHP_3006
         15      SHP_3114
         16      SHP_3122
         17      SHP_6228
         18      SHP_6171
         19      SHP_1378
         20      SHP_1378
         21      SHP_1378
         22      SHP_1377
         23      SHP_1378
         24      SHP_3525
         25      SHP_3204
         26      SHP_3367
         27      SHP_3300
         28      SHP_3527
         29      SHP_3395
                   ...
         8369    SHP_5031
         8370    SHP_3690
         8371    SHP_3591
         8372    SHP_3806
         8373    SHP_3560
         8374    SHP_3637
         8375    SHP_3806
         8376    SHP_3590
         8377    SHP_3729
         8378    SHP_3705
         8379    SHP_3730
         8380    SHP_3807
         8381    SHP_3691
         8382    SHP_3636
         8383    SHP_3731
         8384    SHP_6435
         8385    SHP_2527
         8386    SHP_3189
         8387    SHP_3019
         8388    SHP_6165
         8389    SHP_6192
         8390    SHP_7594
         8391    SHP_7594
         8392    SHP_7519
         8393    SHP_7470
         8394    SHP_7479
```

```
8395    SHP_7555
8396    SHP_7524
8397    SHP_7469
8398    SHP_7628
Name: Ship_id, Length: 8399, dtype: object
```

```
In [11]: # Selecting multiple columns
         market_df.iloc[:, 3:8]
```

Out[11]:

	Cust_id	Sales	Discount	Order_Quantity	Profit
0	Cust_1818	136.8100	0.01	23	-30.51
1	Cust_1818	42.2700	0.01	13	4.56
2	Cust_1818	4701.6900	0.00	26	1148.90
3	Cust_1818	2337.8900	0.09	43	729.34
4	Cust_1818	4233.1500	0.08	35	1219.87
5	Cust_1818	164.0200	0.03	23	-47.64
6	Cust_26	14.7600	0.01	5	1.32
7	Cust_1641	3410.1575	0.10	48	1137.91
8	Cust_1641	162.0000	0.01	33	45.84
9	Cust_1641	57.2200	0.07	8	-27.72
10	Cust_1641	4072.0100	0.01	43	1675.98
11	Cust_708	465.9000	0.05	38	79.34
12	Cust_1088	305.0500	0.04	27	23.12
13	Cust_839	3364.2480	0.10	15	-693.23
14	Cust_839	1410.9300	0.08	10	-317.48
15	Cust_839	460.6900	0.06	48	-103.48
16	Cust_839	443.4600	0.06	30	193.12
17	Cust_1521	13255.9300	0.02	25	4089.27
18	Cust_1521	283.1300	0.08	45	-141.26
19	Cust_371	41.9700	0.05	12	-37.03
20	Cust_371	57.1700	0.08	18	-24.03
21	Cust_371	81.2500	0.01	11	-44.54
22	Cust_371	3202.2500	0.09	44	991.26
23	Cust_371	35.6400	0.05	10	-0.71
24	Cust_931	197.6100	0.08	13	3.46
25	Cust_931	38.2600	0.03	22	-2.34
26	Cust_931	109.5800	0.00	13	31.32
27	Cust_931	1062.6900	0.01	28	401.80
28	Cust_931	3594.7435	0.05	38	1016.97
29	Cust_931	139.9800	0.07	33	-140.54
...	...	...	...	...	...
8369	Cust_1274	1169.2600	0.02	41	515.62

	Cust_id	Sales	Discount	Order_Quantity	Profit
8370	Cust_1006	62.7800	0.04	20	-17.75
8371	Cust_1006	4924.1350	0.07	28	1049.54
8372	Cust_1006	56.9000	0.03	7	12.64
8373	Cust_1006	106.6400	0.10	30	-31.95
8374	Cust_1006	1082.6600	0.08	14	-256.93
8375	Cust_1006	1413.8200	0.10	47	226.53
8376	Cust_1006	1211.0000	0.00	36	-27.99
8377	Cust_1006	34.0100	0.00	12	10.58
8378	Cust_1006	1361.9100	0.05	20	312.52
8379	Cust_1006	1008.9500	0.04	41	69.31
8380	Cust_1006	308.9200	0.04	45	-143.58
8381	Cust_1006	2836.0505	0.01	25	561.13
8382	Cust_1006	120.9800	0.00	28	-92.85
8383	Cust_1006	3508.3300	0.04	21	-546.98
8384	Cust_1577	59.6200	0.04	10	-56.30
8385	Cust_637	611.1600	0.04	46	100.22
8386	Cust_851	121.8700	0.07	39	11.32
8387	Cust_851	41.0600	0.04	4	-16.39
8388	Cust_1519	994.0400	0.03	10	-335.06
8389	Cust_1519	159.4100	0.00	44	34.68
8390	Cust_1798	316.9900	0.04	47	-276.54
8391	Cust_1798	1991.8985	0.07	20	88.36
8392	Cust_1798	181.5000	0.08	43	-6.24
8393	Cust_1798	356.7200	0.07	9	12.61
8394	Cust_1798	2841.4395	0.08	28	374.63
8395	Cust_1798	127.1600	0.10	20	-74.03
8396	Cust_1798	243.0500	0.02	39	-70.85
8397	Cust_1798	3872.8700	0.03	23	565.34
8398	Cust_1798	603.6900	0.00	47	131.39

8399 rows × 5 columns

In [12]:
```
# Selecting multiple rows and columns
market_df.iloc[3:6, 2:5]
```

Out[12]:

	Ship_id	Cust_id	Sales
3	SHP_7625	Cust_1818	2337.89
4	SHP_7664	Cust_1818	4233.15
5	SHP_7608	Cust_1818	164.02

In [13]:
```
# Using booleans
# This selects the rows corresponding to True
market_df.iloc[[True, True, False, True, True, False, True]]
```

Out[13]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01	23	-30.51
1	Ord_5406	Prod_13	SHP_7549	Cust_1818	42.27	0.01	13	4.56
3	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.89	0.09	43	729.34
4	Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.15	0.08	35	1219.87
6	Ord_31	Prod_12	SHP_41	Cust_26	14.76	0.01	5	1.32

To summarise, `df.iloc[x, y]` uses integer indices starting at 0.

The other common way of indexing is the **label based** indexing, which uses `df.loc[]`.

Label Based Indexing

Pandas provides the `df.loc[]` functionality to index dataframes **using labels**.

In [14]:
```
help(pd.DataFrame.loc)
```

```
Help on property:

    Purely label-location based indexer for selection by label.

    ``.loc[]`` is primarily label based, but may also be used with a
    boolean array.

    Allowed inputs are:

    - A single label, e.g. ``5`` or ``'a'``, (note that ``5`` is
      interpreted as a *label* of the index, and **never** as an
      integer position along the index).
    - A list or array of labels, e.g. ``['a', 'b', 'c']``.
    - A slice object with labels, e.g. ``'a':'f'`` (note that contrary
      to usual python slices, **both** the start and the stop are included!).
    - A boolean array.
    - A ``callable`` function with one argument (the calling Series, DataFram
e
      or Panel) and that returns valid output for indexing (one of the above)

    ``.loc`` will raise a ``KeyError`` when the items are not found.

    See more at :ref:`Selection by Label <indexing.label>`
```

As mentioned in the documentation, the inputs x, y to df.loc[x, y] can be:

- A single label, e.g. `'3'` or `'row_index'`
- A list or array of labels, e.g. `['3', '7', '8']`
- A range of labels, where `row_x` and `row_y` **both are included**, i.e. `'row_x':'row_y'`
- A boolean array Let's see some examples.

In [15]:
```
# Selecting a single element
# Select row label = 2 and column label = 'Sales
market_df.loc[2, 'Sales']
```

Out[15]:
```
4701.6899999999996
```

In [16]:
```
# Selecting a single row using a single label
# df.loc reads 5 as a label, not index
market_df.loc[5]
```

Out[16]:
```
Ord_id                    Ord_5446
Prod_id                     Prod_6
Ship_id                   SHP_7608
Cust_id                  Cust_1818
Sales                       164.02
Discount                      0.03
Order_Quantity                  23
Profit                      -47.64
Shipping_Cost                 6.15
Product_Base_Margin           0.37
Name: 5, dtype: object
```

In [17]:
```
# or equivalently
market_df.loc[5, :]

# or market_df.loc[5, ]
```

Out[17]:
```
Ord_id                    Ord_5446
Prod_id                     Prod_6
Ship_id                   SHP_7608
Cust_id                  Cust_1818
Sales                       164.02
Discount                      0.03
Order_Quantity                  23
Profit                      -47.64
Shipping_Cost                 6.15
Product_Base_Margin           0.37
Name: 5, dtype: object
```

In [18]:
```
# Select multiple rows using a list of row labels
market_df.loc[[3, 7, 8]]
```

Out[18]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Pro
3	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.8900	0.09	43	729.3
7	Ord_4725	Prod_4	SHP_6593	Cust_1641	3410.1575	0.10	48	1137.
8	Ord_4725	Prod_13	SHP_6593	Cust_1641	162.0000	0.01	33	45.84

In [19]:
```
# Or equivalently
market_df.loc[[3, 7, 8], :]
```

Out[19]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Pro
3	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.8900	0.09	43	729.3
7	Ord_4725	Prod_4	SHP_6593	Cust_1641	3410.1575	0.10	48	1137.
8	Ord_4725	Prod_13	SHP_6593	Cust_1641	162.0000	0.01	33	45.84

In [20]:
```
# Selecting rows using a range of labels
# Notice that with df.loc, both 4 and 8 are included, unlike with df.iloc
# This is an important difference between iloc and loc
market_df.loc[4:8]
```

Out[20]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Pro
4	Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.1500	0.08	35	1219.
5	Ord_5446	Prod_6	SHP_7608	Cust_1818	164.0200	0.03	23	-47.64
6	Ord_31	Prod_12	SHP_41	Cust_26	14.7600	0.01	5	1.32
7	Ord_4725	Prod_4	SHP_6593	Cust_1641	3410.1575	0.10	48	1137.
8	Ord_4725	Prod_13	SHP_6593	Cust_1641	162.0000	0.01	33	45.84

In [21]:
```
# Or equivalently
market_df.loc[4:8, ]
```

Out[21]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Pro
4	Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.1500	0.08	35	1219.
5	Ord_5446	Prod_6	SHP_7608	Cust_1818	164.0200	0.03	23	-47.64
6	Ord_31	Prod_12	SHP_41	Cust_26	14.7600	0.01	5	1.32
7	Ord_4725	Prod_4	SHP_6593	Cust_1641	3410.1575	0.10	48	1137.
8	Ord_4725	Prod_13	SHP_6593	Cust_1641	162.0000	0.01	33	45.84

In [22]:
```
# Or equivalently
market_df.loc[4:8, :]
```

Out[22]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Pro
4	Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.1500	0.08	35	1219.8
5	Ord_5446	Prod_6	SHP_7608	Cust_1818	164.0200	0.03	23	-47.64
6	Ord_31	Prod_12	SHP_41	Cust_26	14.7600	0.01	5	1.32
7	Ord_4725	Prod_4	SHP_6593	Cust_1641	3410.1575	0.10	48	1137.9
8	Ord_4725	Prod_13	SHP_6593	Cust_1641	162.0000	0.01	33	45.84

In [23]:
```
# The use of label based indexing will be more clear when we have custom row i
ndices
# Let's change the indices to Ord_id
market_df.set_index('Ord_id', inplace = True)
market_df.head()
```

Out[23]:

	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	S
Ord_id								
Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01	23	-30.51	3.
Ord_5406	Prod_13	SHP_7549	Cust_1818	42.27	0.01	13	4.56	0.
Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.69	0.00	26	1148.90	2.
Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.89	0.09	43	729.34	1
Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.15	0.08	35	1219.87	2

In [24]:
```
# Select Ord_id = Ord_5406 and some columns
market_df.loc['Ord_5406', ['Sales', 'Profit', 'Cust_id']]
```

Out[24]:
```
Sales          42.27
Profit          4.56
Cust_id    Cust_1818
Name: Ord_5406, dtype: object
```

In [25]:
```python
# Select multiple orders using labels, and some columns
market_df.loc[['Ord_5406', 'Ord_5446', 'Ord_5485'], 'Sales':'Profit']
```

Out[25]:

	Sales	Discount	Order_Quantity	Profit
Ord_id				
Ord_5406	42.27	0.01	13	4.56
Ord_5446	136.81	0.01	23	-30.51
Ord_5446	4701.69	0.00	26	1148.90
Ord_5446	164.02	0.03	23	-47.64
Ord_5485	4233.15	0.08	35	1219.87

In [26]:
```python
# Using booleans
# This selects the rows corresponding to True
market_df.loc[[True, True, False, True, True, False, True]]
```

Out[26]:

	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	S
Ord_id								
Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01	23	-30.51	3.
Ord_5406	Prod_13	SHP_7549	Cust_1818	42.27	0.01	13	4.56	0.
Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.89	0.09	43	729.34	1
Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.15	0.08	35	1219.87	2
Ord_31	Prod_12	SHP_41	Cust_26	14.76	0.01	5	1.32	0.

To summarise, we discussed two **explicit ways of indexing dataframes** - `df.iloc[]` and `df.loc[]`. Next, let's study how to slice and dice sections of dataframes.