Selecting rows

By selecting data from our dataset, we can answer the following questions

- Which customers joined in 2015?
- Which customers spent the most on a single transaction?
- Which transactions had a purchase amount greater than 100?

General command structure for addressing rows and columns in a pandas DataFrame

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...

Accepts only integers (single integer, list, or slice object) or boolean arrays as input. ❶

Option 1: `iloc`
Integer based selection (e.g., 3)

Option 2: `loc`
Label based selection (e.g., 5 or 'a')

Accepts only labels (single label, list, or slice object) or boolean arrays as input. Note that 5 is interpreted as a label of the index, and never as an integer position along the index. ❷

Row Indexer ❹

Column Indexer ❺

```
myData.(i)loc[ , ]
```

Name of DataFrame ❸

Square brackets ❻

There are multiple ways of selecting rows

1. Selecting rows by row numbers
2. Selecting rows by conditions

Selecting rows by row numbers
Select the first row

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...

Select the first row

Row number(s) to be selected (1)

```
myData.iloc [0, ]
```

Customer	149332
TransDate	15.11.2005
Quantity	1
PurchAmount	199.95
Cost	107.00
TransID	127998739

Returns a series (2)

Returns a DataFrame (4)

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739

```
myData.iloc [[0], ]
```

Row number(s) to be selected (3)

Sidenote: Selecting does not make changes to the original DataFrame

```
myData.iloc[[0],]
```

Select first row 1

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739

The output of select operations need to be stored via "=" 2

```
myData
```

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	12.06.2007	1	79.95	35.00	128670302

myData was not changed 3

Selecting rows by row numbers
Select the first 3 rows

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...

Select the first 3 rows

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247

```
myData.iloc[0:3, ]
```

Row numbers to be selected.
":" generates a sequence for slicing.

Sidenote: "0"-based indexing in Python

Python uses 0-based indexing, i.e. the **first index is 0** (not 1).

Python Basics: Use the colon operator (:) for selection procedures

":" generates a regular sequence

```
> myData.iloc[0:5]
```

	Customer	TransDate	Quantity	PurchAmount	Cost	TransID
0	149332	15.11.2005	1	199.95	107.00	127998739
1	172951	29.08.2008	1	199.95	108.00	128888288
2	120621	19.10.2007	1	99.95	49.00	125375247
3	149236	14.11.2005	1	39.95	18.95	128670302
4	149236	12.06.2007	1	79.95	35.00	128670302

```
> myData.iloc[0:0]
Empty DataFrame
```

Selecting rows by row numbers
Select the first 3 and the 5th row

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
…	…	…	…	…	…

Select the first 3 rows and the 5th

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	12.06.2007	1	79.95	35.00	128670302

```
myData.iloc[[0,1,2,4], ]
```

Combine integers in a **vector**

Python Basics: Understand the dimensions of your DataFrame

Important functions to determine dimensions:

- Number of rows/columns:

```
myData.shape[0]
myData.shape[1]
```

- Length of a vector:

```
len([2,3,5])
```

- Length of string:

```
len("hello")
```

Python Basics: Understand the dimensions of your DataFrame

Important functions to determine dimensions:

- Number of rows/columns:

```
myData.shape[0]
myData.shape[1]
```

- Length of a vector:

```
len([2,3,5])
```

- Length of string:

```
len("hello")
```

Selecting rows by row numbers
Select the last row

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...
199542	17.09.2012	1	39.95	10.50	13197336

Select the last row

Customer	TransDate	Quantity	PurchAmount	Cost
199542	17.09.2012	1	39.95	10.50

1 `len()` gives the number of rows. To obtain the last one we have to correct our index by- 1 (reason: zero-based indexing)

```
myData.iloc[[len(myData)-1], ]
```

```
myData.tail(1)
```

2 Brackets are needed if representation of the output as Data.Frame is desired.

Selecting rows by row numbers
Select the last row

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...
199542	17.09.2012	1	39.95	10.50	13197336

Select the last row

Customer	TransDate	Quantity	PurchAmount	Cost
199542	17.09.2012	1	39.95	10.50

1 `len()` gives the number of rows. To obtain the last one we have to correct our index by- 1 (reason: zero-based indexing)

```
myData.iloc[[len(myData)-1], ]
```

```
myData.tail(1)
```

2 Brackets are needed if representation of the output as Data.Frame is desired.

Sidenote: How to sort your DataFrame

- **To sort your DataFrame according to transaction dates (increasing), use:**

```
myData.sort_values(["TransDate"])
```

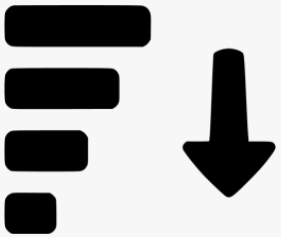
- **Order first according to transaction dates and then according to customers:**

```
myData.sort_values(["TransDate", "Customer"])
```

- **Order decreasing:**

```
myData.sort_values(["TransDate", "Customer"], ascending=[0,0])
```

Specify decreasing order explicitly
(by default: 1 = increasing).



Selecting rows by condition
Identify transactions greater than $100

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...

Select transactions with value > 100

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288

```
myData.loc[myData["PurchAmount"] > 100, ]
```

Select all transactions > $100

Python Basics: Logical Operators

Sign	Description	Example
`<`	less than	`a < 0`
`<=`	less than or equal than	`a <= 3`
`>`	greater than	`a > 0`
`>=`	greater than or equal than	`a >= 3`
`==`	equal to	`a == 0`
`!=`	not equal to	`!= 0`
`not`	logical negotiation (NOT)	`not x`
`&`	logical AND	`x & y`
`|`	logical OR	`x | y`

Selecting rows by condition
Select the transactions of a single customer

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...

Select transactions where `Customer` is `149332`

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
...	...	...	...	...	...

Note: Variable `Customer` is of type integer (1)

```
myData.loc[myData["Customer"] == 149332, ]
```

Selects all observations where `Customer` is equal to `149332` (2)

Selecting rows by condition
Select the transactions of multiple customers

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...

Select transactions where `Customer` is `149332` or `172951`

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
...	...	...	...	...	...
172951	29.08.2008	1	199.95	108.00	12888288
...	...	...	...	...	...

```
myData.loc[myData["Customer"].isin([149332, 172951]), ]
```

Selects all observations where `Customer` is either `149332` **or** `172951`

Tilde operator (~) precedes an index vector to negate the condition

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...

Select transactions where `Customer` is **NOT** `149332` or `172951`

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...

```
myData.loc[~ myData["Customer"].isin([149332, 172951]), ]
```

"Not in"

Combining conditions

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...
140729	28.04.2012	1	89.95	35.00	12937773
...	...	...	...	...	...

Select transactions where date after 24.10.2007 **and** `PurchAmount` > 100

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
172951	29.08.2008	1	199.95	108.00	128888288
...	...	...	...	...	...

```
myData.loc[(myData["TransDate"]>
           pd.to_datetime("2007-10-24")) &
           (myData["PurchAmount"]>100), ]
```

1. Create a `date`
2. Combine multiple conditions
3. Round brackets

Combining conditions

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
149332	15.11.2005	1	199.95	107.00	127998739
172951	29.08.2008	1	199.95	108.00	128888288
120621	19.10.2007	1	99.95	49.00	125375247
149236	14.11.2005	1	39.95	18.95	127996226
149236	12.06.2007	1	79.95	35.00	128670302
...	...	...	...	...	...
140729	28.04.2012	1	89.95	35.00	12937773
...	...	...	...	...	...

Select transactions where date after 24.10.2007 **and** `PurchAmount` > 100

Customer	TransDate	Quantity	PurchAmount	Cost	TransID
172951	29.08.2008	1	199.95	108.00	128888288
...	...	...	...	...	...

```
myData.loc[(myData["TransDate"]>
           pd.to_datetime("2007-10-24")) &
           (myData["PurchAmount"]>100), ]
```

1. Create a `date`
2. Combine multiple conditions
3. Round brackets

Selecting rows