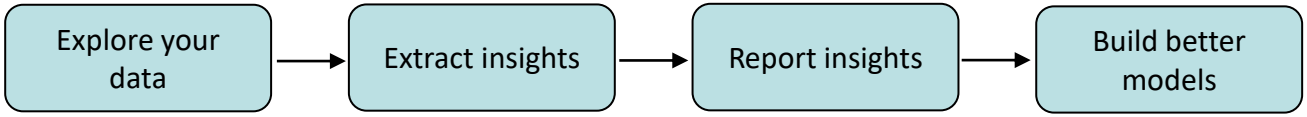
Creating plots with `matplotlib`

Data visualization in Python

- Data visualization is an essential part in the data analysis process:



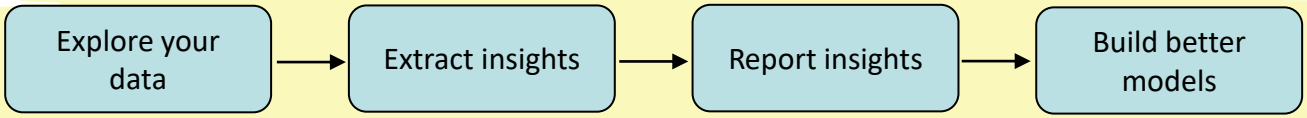
- Python's plotting modules and packages enable customized graphs and more:

Packages	Description
Matplotlib (pyplot interface)	- Open Source plotting library - Interactive plotting - Syntax fimiliar to Matlab
Seaborn	- Visualization library based on matplotlib with simple functions - Provides good default values and integration with Pandas
Plotly	- Create Interactive Web Graphics via "plotly.js"
ggplot	- Based on R's ggplot2 - "Grammar of Graphics": build your plot from various layers

https://wiki.python.org/moin/NumericAndScientific/Plotting

Data visualization in Python

- Data visualization is an essential part in the data analysis process:



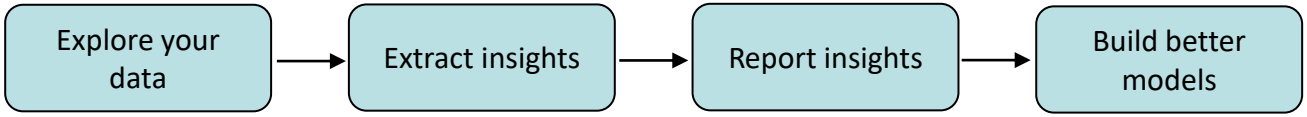
- Python's plotting modules and packages enable customized graphs and more:

Packages	Description
Matplotlib (pyplot interface)	- Open Source plotting library - Interactive plotting - Syntax fimiliar to Matlab
Seaborn	- Visualization library based on matplotlib with simple functions - Provides good default values and integration with Pandas
Plotly	- Create Interactive Web Graphics via "plotly.js"
ggplot	- Based on R's ggplot2 - "Grammar of Graphics": build your plot from various layers

https://wiki.python.org/moin/NumericAndScientific/Plotting

Data visualization in Python

- Data visualization is an essential part in the data analysis process:



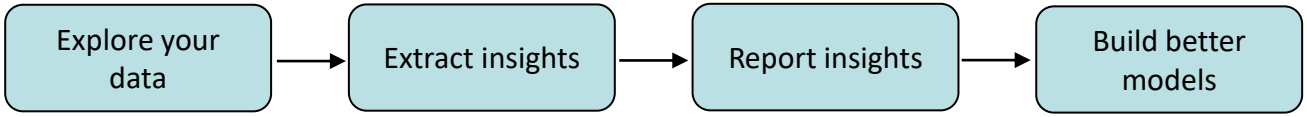
- Python's plotting modules and packages enable customized graphs and more:

Packages	Description
Matplotlib (pyplot interface)	- Open Source plotting library - Interactive plotting - Syntax fimiliar to Matlab
Seaborn	- Visualization library based on matplotlib with simple functions - Provides good default values and integration with Pandas
Plotly	- Create Interactive Web Graphics via "plotly.js"
ggplot	- Based on R's ggplot2 - "Grammar of Graphics": build your plot from various layers

https://wiki.python.org/moin/NumericAndScientific/Plotting

Data visualization in Python

- Data visualization is an essential part in the data analysis process:



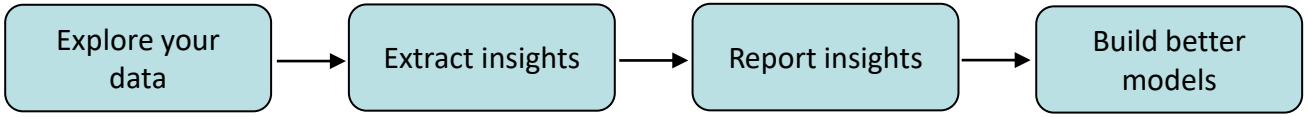
- Python's plotting modules and packages enable customized graphs and more:

Packages	Description
Matplotlib (pyplot interface)	- Open Source plotting library - Interactive plotting - Syntax fimiliar to Matlab
Seaborn	- Visualization library based on matplotlib with simple functions - Provides good default values and integration with Pandas
Plotly	- Create Interactive Web Graphics via "plotly.js"
ggplot	- Based on R's ggplot2 - "Grammar of Graphics": build your plot from various layers

https://wiki.python.org/moin/NumericAndScientific/Plotting

Data visualization in Python

- Data visualization is an essential part in the data analysis process:



- Python's plotting modules and packages enable customized graphs and more:

Packages	Description
Matplotlib (pyplot interface)	- Open Source plotting library - Interactive plotting - Syntax fimiliar to Matlab
Seaborn	- Visualization library based on matplotlib with simple functions - Provides good default values and integration with Pandas
Plotly	- Create Interactive Web Graphics via "plotly.js"
ggplot	- Based on R's ggplot2 - "Grammar of Graphics": build your plot from various layers

https://wiki.python.org/moin/NumericAndScientific/Plotting

Data visualization in Python
Why use matplotlib and the pyplot interface?

- The matplotlib library:
  - Provides an easy but flexible Python 2D plotting library which visualizes figures in an interactive environment across platforms.
- The corresponding **pyplot** interface:

Import the interface with:
```
import matplotlib.pyplot as plt
```

  - Collection of command style functions that offer plotting in a Matlab-like interface.
  - Each `pyplot` function makes some changes to a figure, e.g. creates a figure, adds some lines, labels the axes of the plot, etc.
  - Keeps track of the current figure and plotting area when adding functions or features to the plot.

Data visualization in Python
Why use matplotlib and the pyplot interface?

- The matplotlib library:
  - Provides an easy but flexible Python 2D plotting library which visualizes figures in an interactive environment across platforms.
- The corresponding **pyplot** interface:

Import the interface with:
```
import matplotlib.pyplot as plt
```

  - Collection of command style functions that offer plotting in a Matlab-like interface.
  - Each `pyplot` function makes some changes to a figure, e.g. creates a figure, adds some lines, labels the axes of the plot, etc.
  - Keeps track of the current figure and plotting area when adding functions or features to the plot.

Data visualization in Python
Why use matplotlib and the pyplot interface?

- The  library:
  - Provides an easy but flexible Python 2D plotting library which visualizes figures in an interactive environment across platforms.
- The corresponding **pyplot** interface:
  - Collection of command style functions that offer plotting in a Matlab-like interface.
  - Each `pyplot` function makes some changes to a figure, e.g. creates a figure, adds some lines, labels the axes of the plot, etc.
  - Keeps track of the current figure and plotting area when adding functions or features to the plot.

Import the interface with:
```
import matplotlib.pyplot as plt
```

Good plots have 3 characteristics

Plots should be:

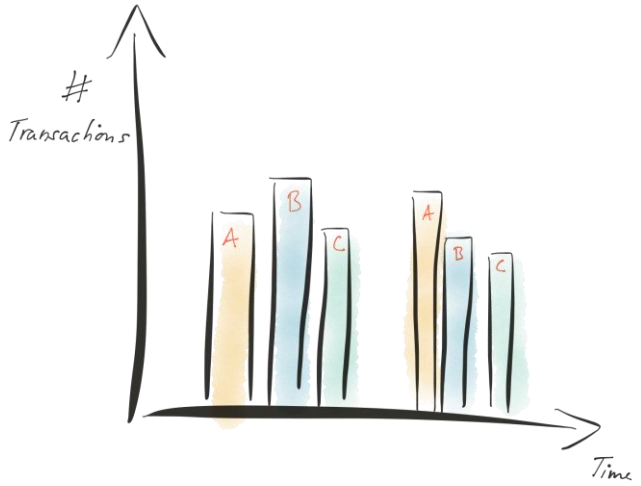
- Informative
- Easy to understand
- Visually appealing

How to plot:
Steps

1. Choose the plot type
2. Find the appropriate matplotlib function
3. Transform data
4. Create the plot
5. Improve aesthetic features of the plot
6. Save plot

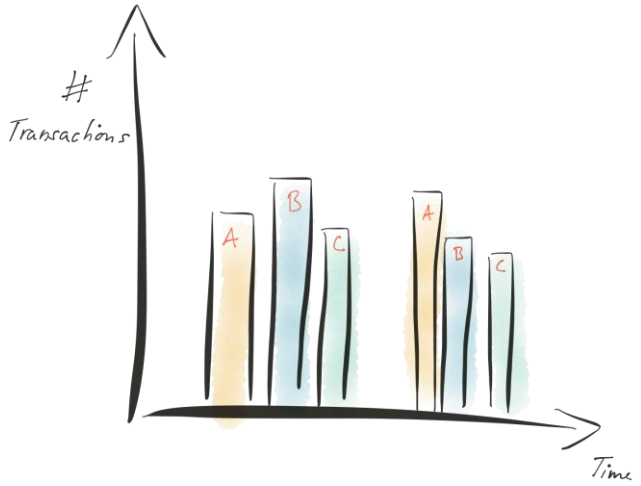
Step 1: Choose the plot type
Decide the best way to convey the information

- What do you want to show?
  - A single variable?
  - The relationship between multiple variables?
- Is your data continuous or discrete?

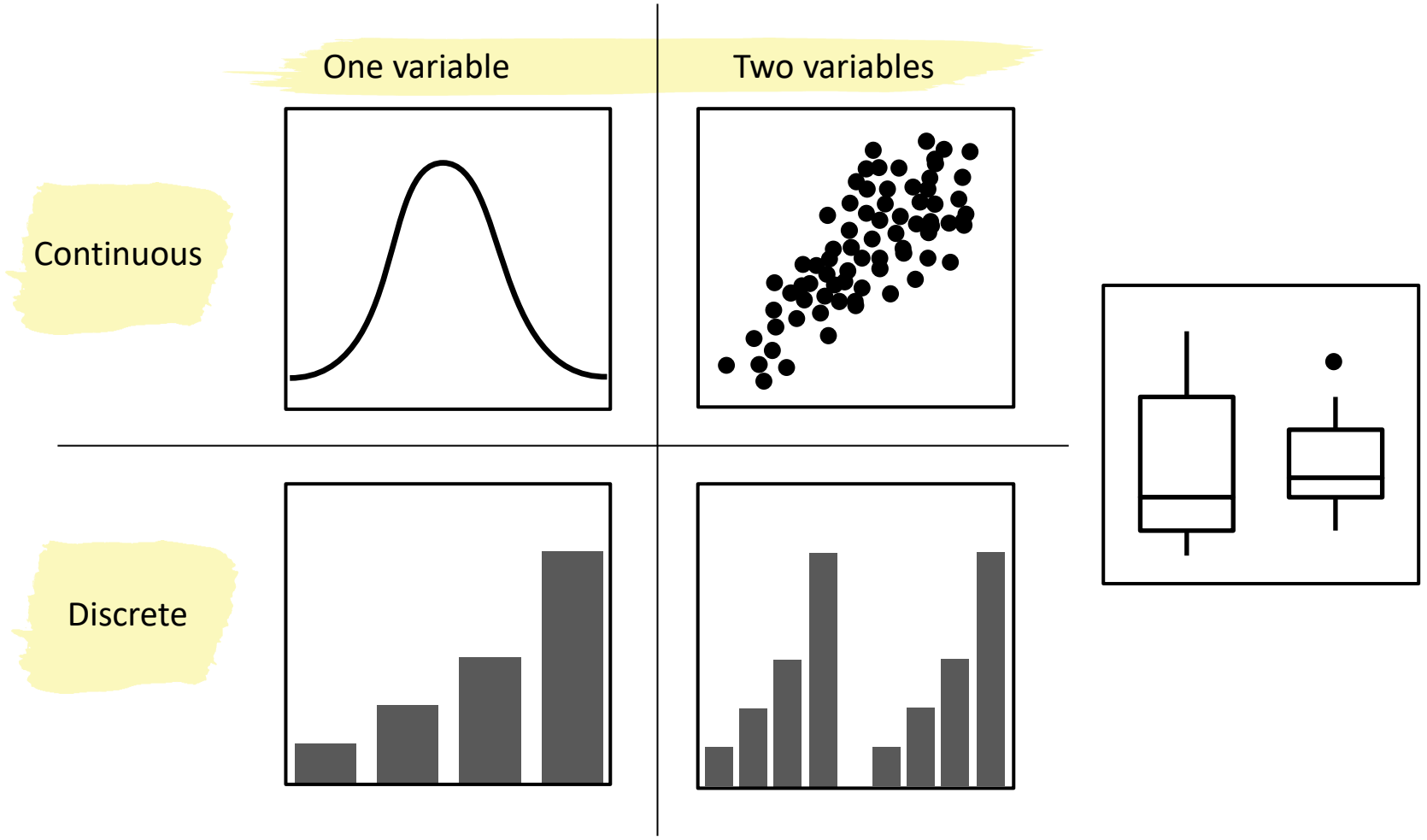


Step 1: Choose the plot type
Decide the best way to convey the information

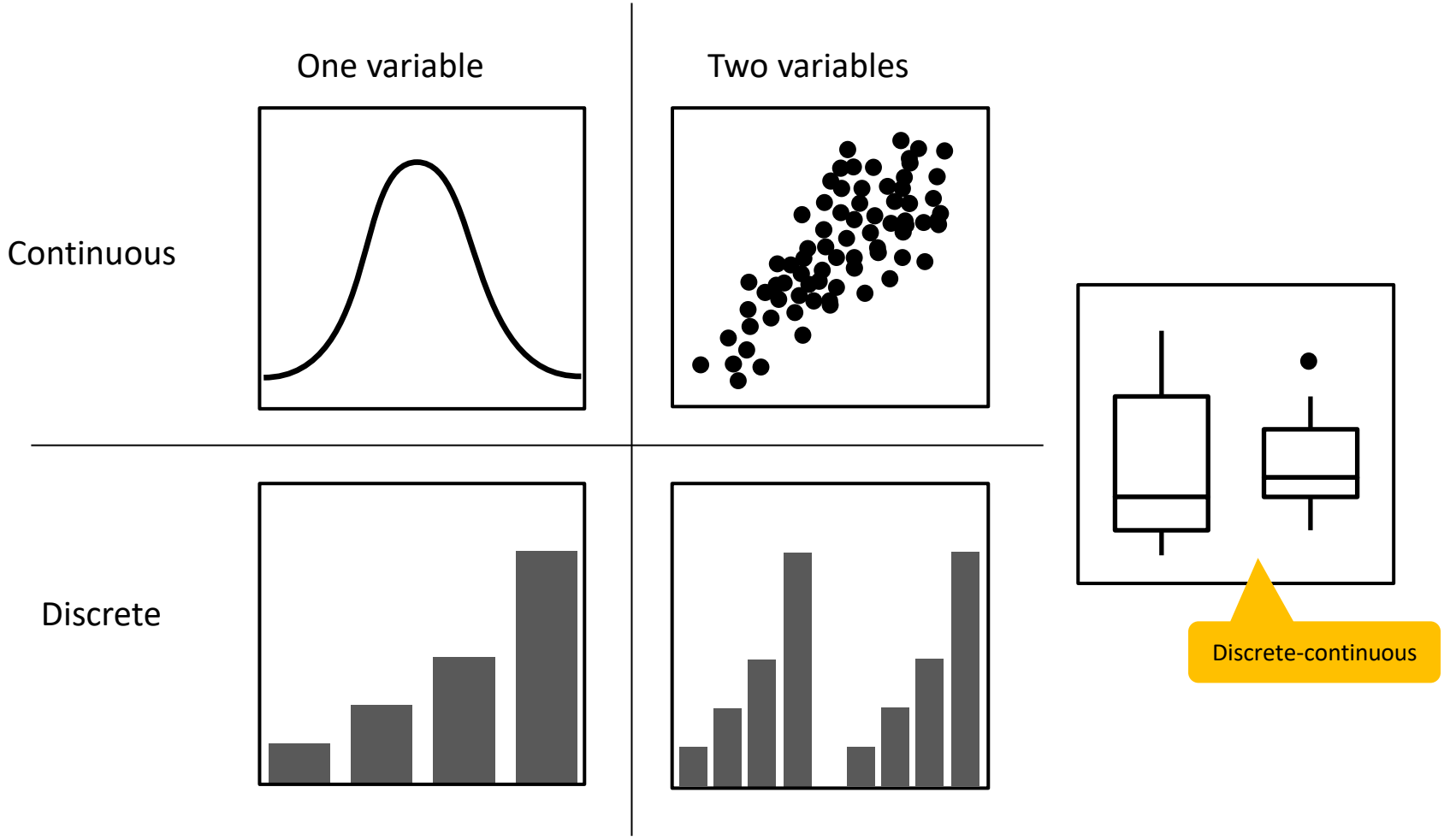
- What do you want to show?
  - A single variable?
  - The relationship between multiple variables?
- Is your data continuous or discrete?



Different combinations of variables can be portrayed with different plot types

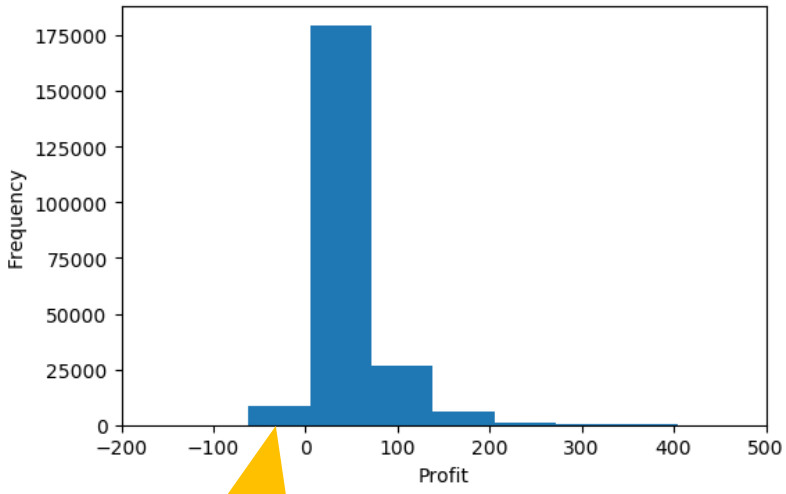


Different combinations of variables can be portrayed with different plot types



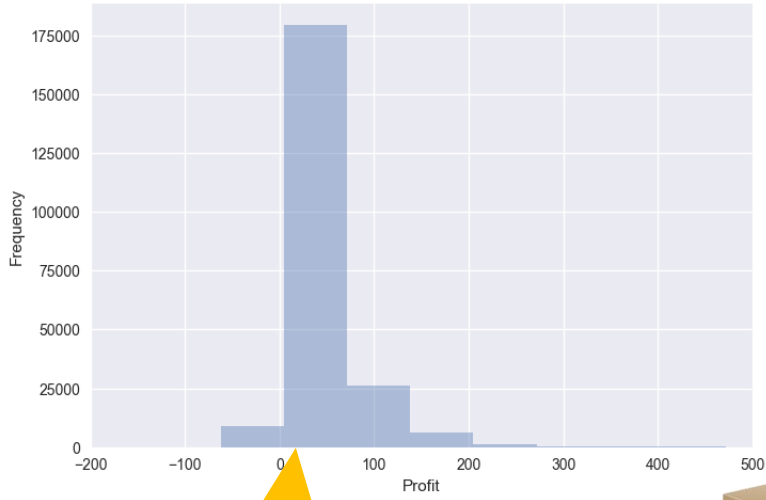
Step 2: Find the function – Pyplot and Seaborn (both Matplotlib-based) are the most used plotting tools

Pyplot



`Pyplot` is a collection of command style functions. Each pyplot function makes some change to a figure: e.g., creates a figure, adds a title, labels the axes, etc. ❶

Seaborn

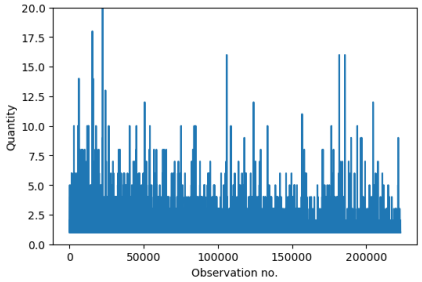


`Seaborn` provides easy functions and good defaults, such as the background and axis styles. It can be extended with `pyplot` functionality. ❷

Step 2: Find the function – Matplotlib with the pyplot interface is the most used plotting tool in Python

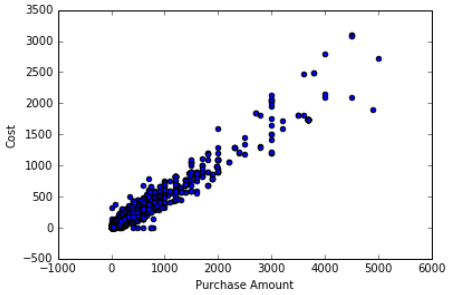
Regular plots (lines, density)

```
plt.plot(x,...)
```



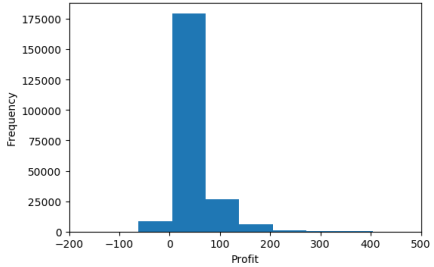
Scatterplot

```
plt.scatter(
    x,y,...)
```



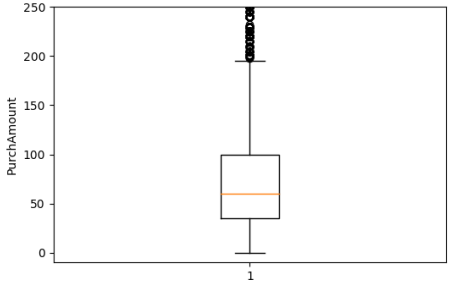
Histogram

```
plt.hist(x,...)
```



Boxplot

```
plt.boxplot(
    x,...)
```



`pyplot` is a very popular interface that facilitates making plots by deconstructing them into layers

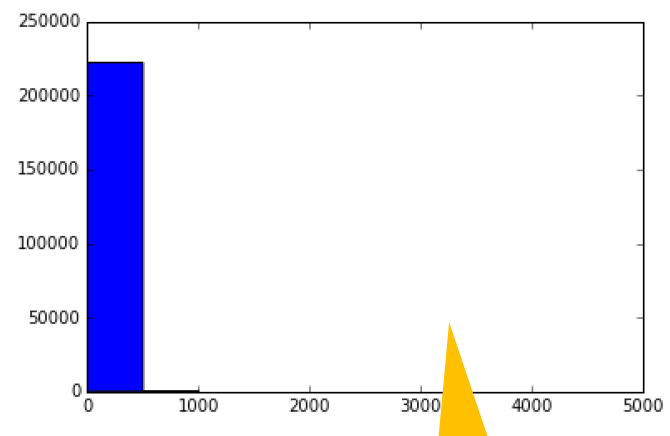
… there's more!

Step 3: Transform data
Some graphs might require transformed data input

- It is quite rare that you can plot your data right away, i.e. certain **plots have requirements** on how the data should look like.
- In most cases it is **necessary to transform** your data before plotting it.
- Examples:
  - Transform times and dates for aggregation of month or years (Lecture 2, Lecture 7)
  - Group data for better overview
  - Logarithmic transformations for nicer distributions

Step 4: Create the plot (1/2)

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
...	...	...	...	...	...



A meaningful interpretation is hardly possible.

Multilayer principle:

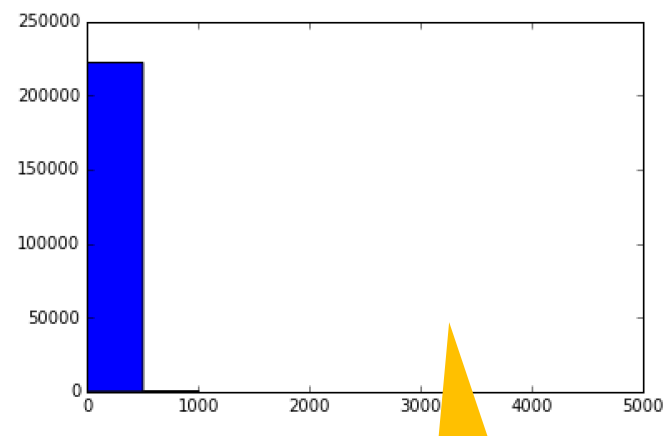
1 Determine how and what to draw

```
plt.hist(x=myData["PurchAmount"], bins=10)
plt.show()
```

2 Command to actually plot the figure

Step 4: Create the plot (1/2)

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
...	...	...	...	...	...



A meaningful interpretation is hardly possible.

Multilayer principle:

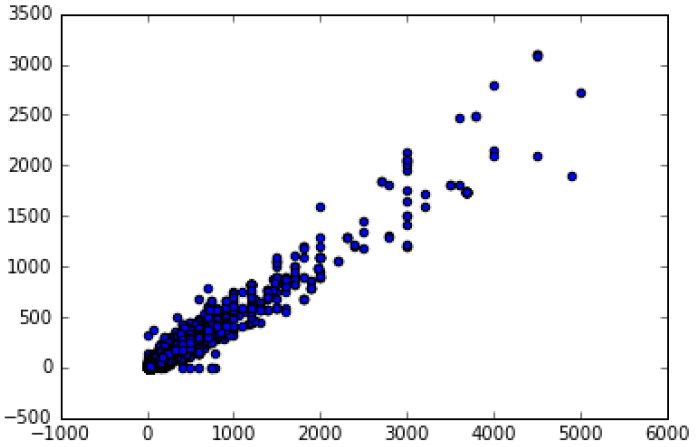
1. Determine how and what to draw

```
plt.hist(x=myData["PurchAmount"], bins=10)
plt.show()
```

2. Command to actually plot the figure

Step 4: Create the plot (2/2)

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
...	...	...	...	...	...



x variable 1

y variable 2

```
plt.scatter(x=myData["PurchAmount"], y=myData["Cost"])
```

Creating plots with `matplotlib`