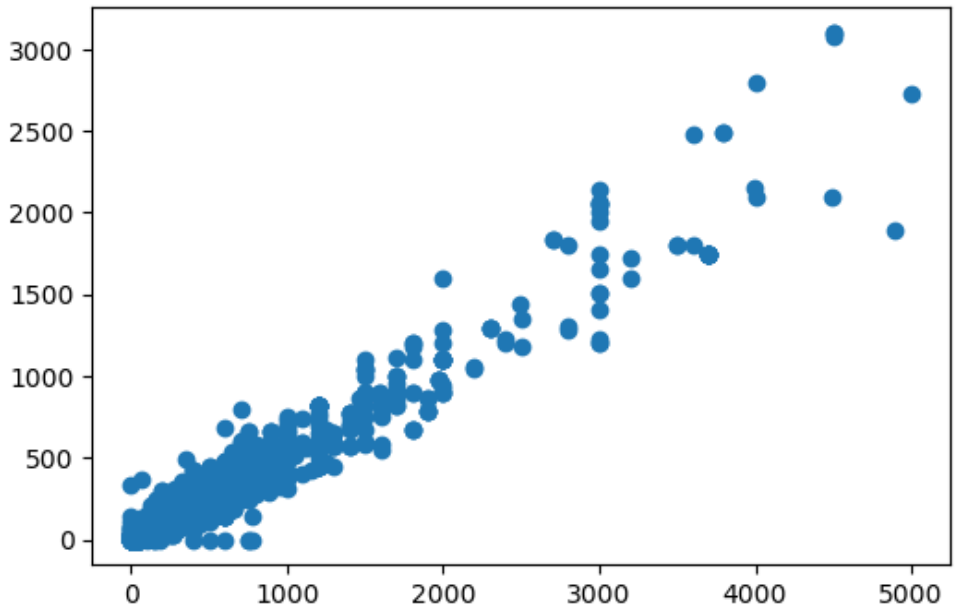
Formatting plots

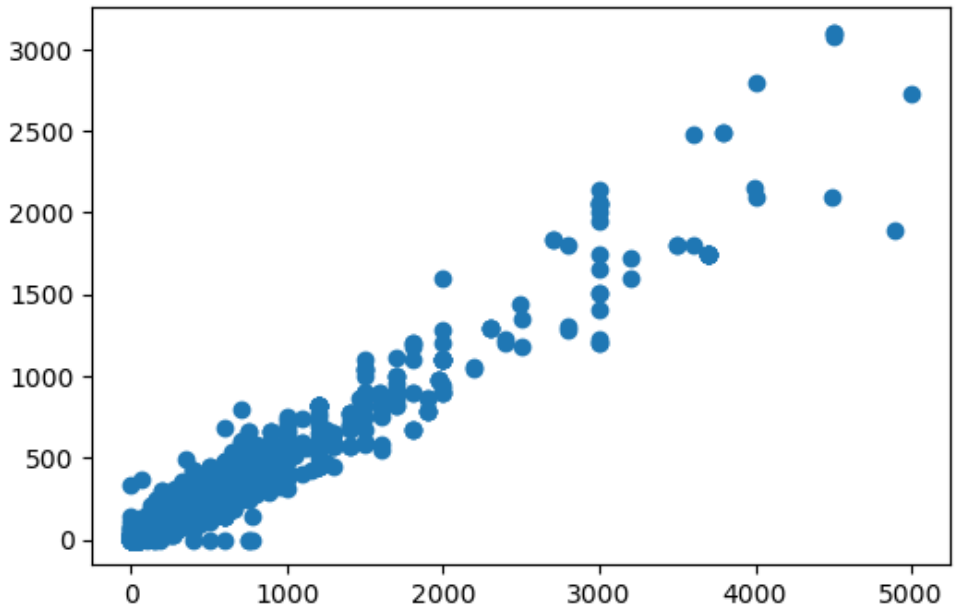
Step 5: Improve aesthetic features of the plot
The standard plot output



1 We switch to `plt.plot()`, since `plt.scatter()` offers less options

```
plt.plot(myData["PurchAmount"], myData["Cost"],"o")
plt.show()
```

Step 5: Improve aesthetic features of the plot
The standard plot output

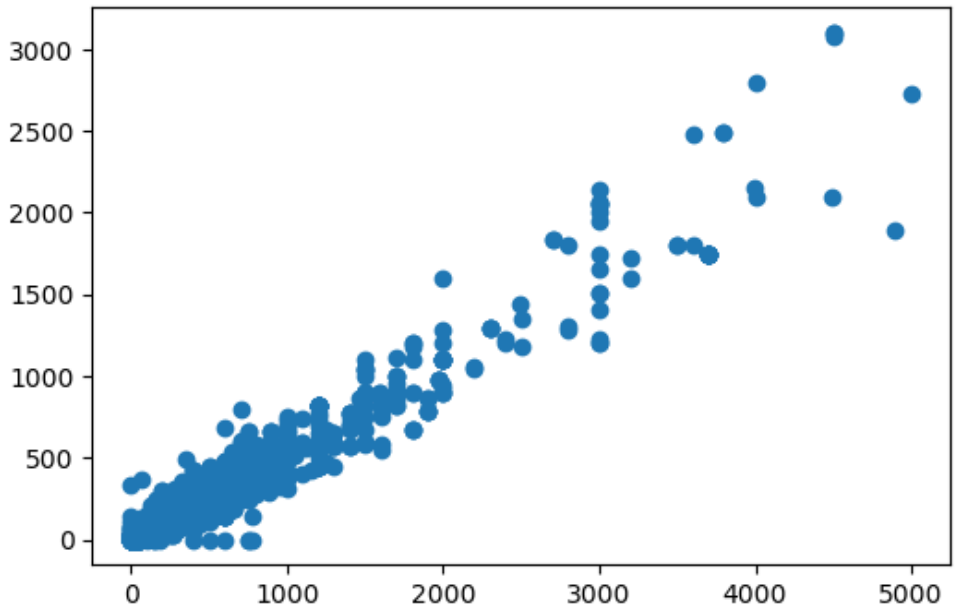


1 We switch to `plt.plot()`, since `plt.scatter()` offers less options

```
plt.plot(myData["PurchAmount"], myData["Cost"],"o")
plt.show()
```

2 You do **not** explicitly need to specify "x=" and "y=“. However, their order must fit.

Step 5: Improve aesthetic features of the plot
The standard plot output



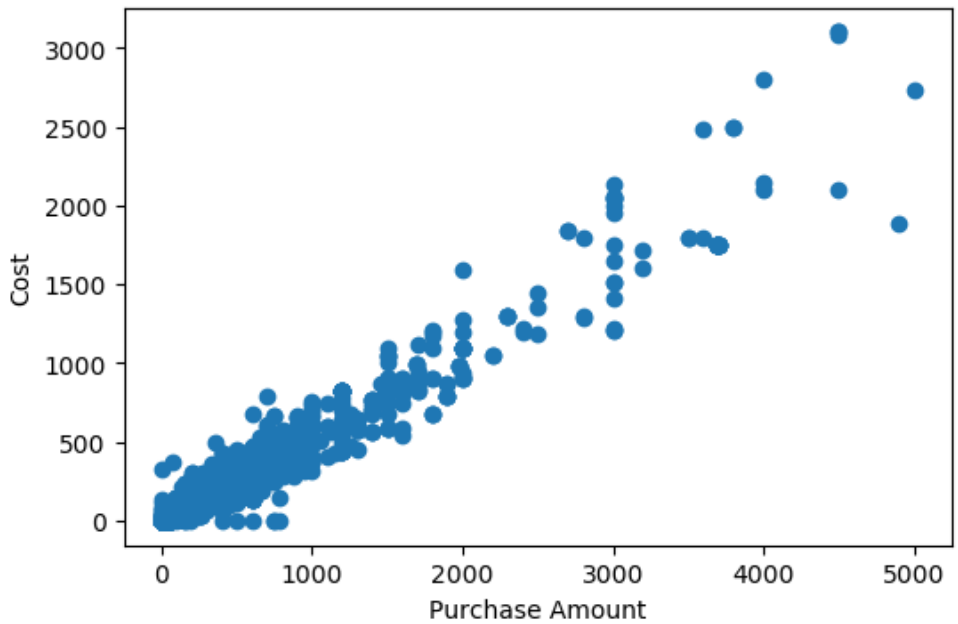
1 We switch to `plt.plot()`, since `plt.scatter()` offers less options

3 Type of symbol used for the plotting

```
plt.plot(myData["PurchAmount"], myData["Cost"],"o")
plt.show()
```

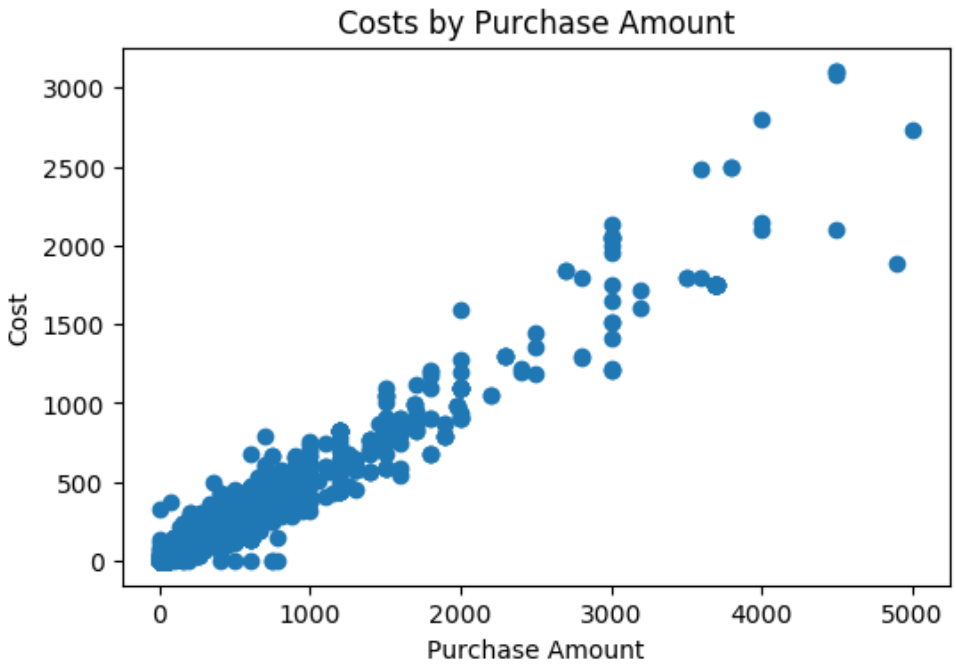
2 Do **not** explicitly specify "x=" and "y="

Step 5: Improve aesthetic features of the plot
Change the axis labels




```
plt.plot(x=myData["PurchAmount"], y=myData["Cost"])
plt.xlabel("Purchase Amount")
plt.ylabel("Cost")
plt.show()
```

Step 5: Improve aesthetic features of the plot
Add a fitting and descriptive title




```
...
plt.title("Costs by Purchase Amount")
plt.show()
```

Step 5: Improve aesthetic features of the plot
Adjust the axes limits


```
...
plt.xlim(-100,6000)
plt.ylim(-100,3500)
plt.show()
```

Step 5: Improve aesthetic features of the plot
Change the marker type to squares

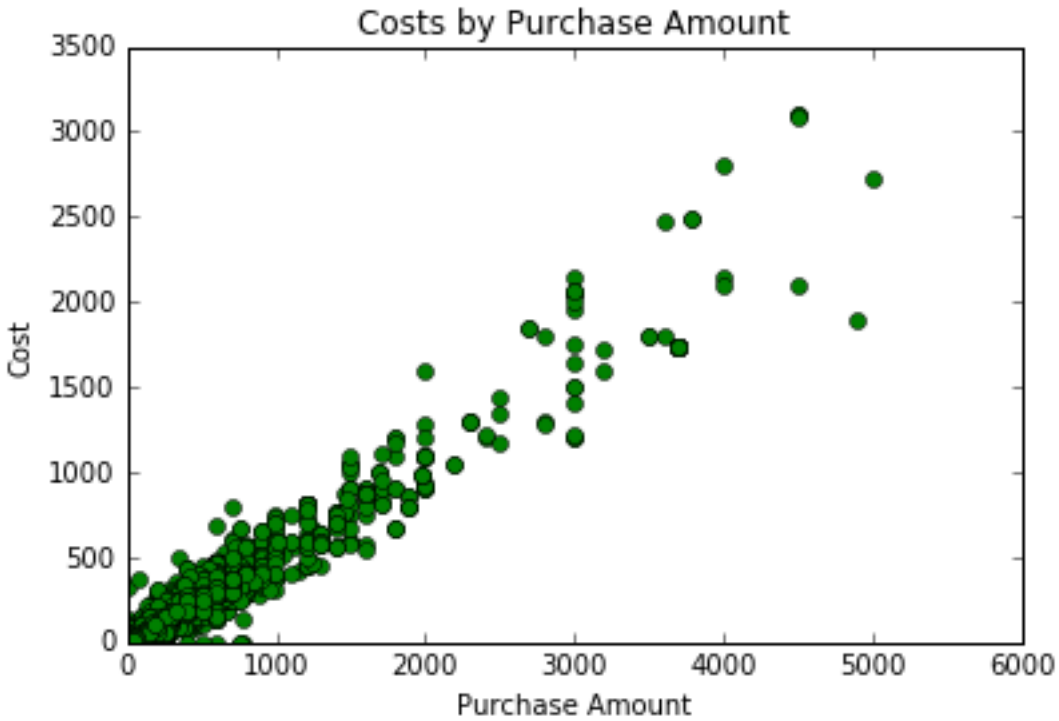
Overview of format strings to control the marker:

Character	Description
's'	square marker
'p'	Pentagon marker
'o'	circle marker
'v'	triangle marker
'x'	x marker

```
...
plt.plot(myData["PurchAmount"], myData["Cost"], "s")
plt.show()
```

plt.scatter()
does *not* work here!

Step 5: Improve aesthetic features of the plot
Choose a nice color



You can pass the color (g) and the marker (o) in one argument ❶

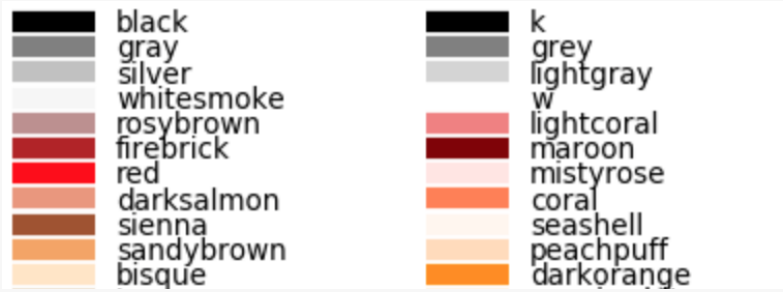
```
1) plt.plot(myData["PurchAmount"], myData["Cost"], "go")
                    plt.show()
                    …
2) plt.plot(myData["PurchAmount"], myData["Cost"], "o", color="green")
                    plt.show()
```

Alternatively, specify the color in a separate argument ❷

Sidenote: `Matplotlib`'s color repertoire is huge

- `Matplotlib` colors can be referenced as strings in certain functions:
- Overview over format characters to control the color:

Character	Description
`'b'`	blue
`'g'`	green
`'r'`	red
`'c'`	cyan
...	and many more



- See https://stackoverflow.com/questions/22408237/named-colors-in-matplotlib

Step 5: Improve aesthetic features of the plot
Change the size of the points


```
plt.plot(myData["PurchAmount"], myData["Cost"],"o", markersize=10)
plt.show()
```

... and the text size


```
plt.plot(myData["PurchAmount"], myData["Cost"], "o")
plt.xlabel("Purchase Amount", size=20)
plt.ylabel("Cost", size=20)
```

Step 5: Improve aesthetic features of the plot
Make the title italic and the axis label bold


```
plt.plot(myData["PurchAmount"], myData["Cost"], "o")
plt.xlabel("Purchase Amount", size=20, fontweight='bold')
plt.title("Costs by Purchase Amount", style='italic')
```

Step 5: Improve aesthetic features of the plot
Remove the box

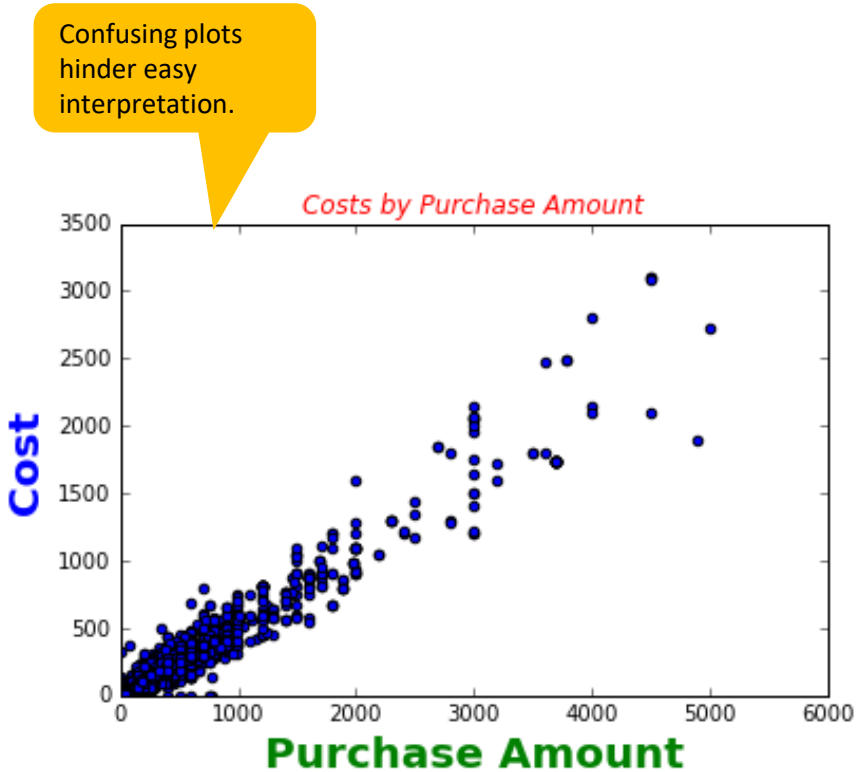
`Pylab` combines the `pyplot` with the `numpy` functionality

```
from pylab import *
axes(frameon = 0)
plt.plot(myData["PurchAmount"], myData["Cost"], "o")
…
```

Step 5: Improve aesthetic features of the plot
Change text color


```
plt.plot(myData["PurchAmount"], myData["Cost"], "o")
plt.xlabel("Purchase Amount", size=20, fontweight='bold', color="green")
plt.ylabel("Cost", size=20, fontweight='bold', color="blue")
plt.title("Costs by Purchase Amount", style='italic', color="red")
```

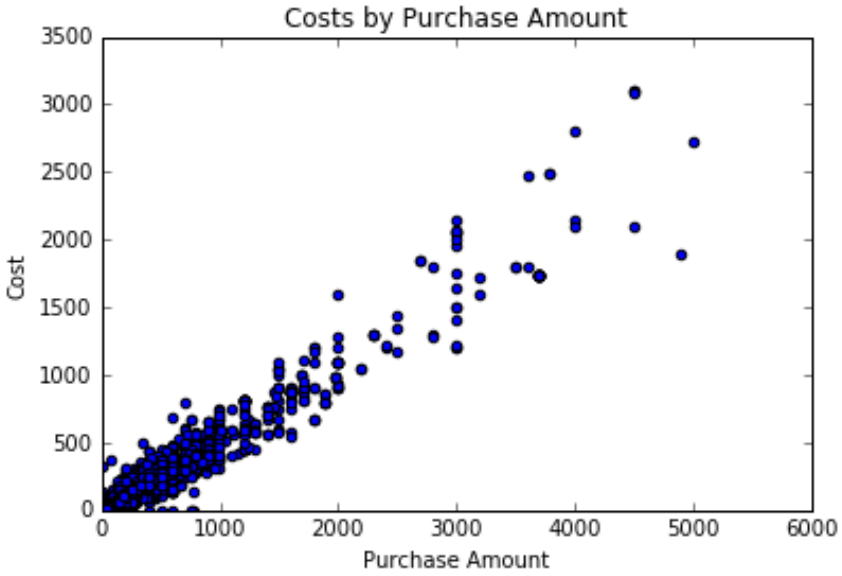
Why not include everything?




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

plt.xlabel("Purchase Amount",
    size=20,fontweight="bold",
    color="green")

plt.ylabel("Cost", size=20,
    fontweight="bold",
    color="blue")

plt.title("Costs by Purchase
    Amount", style="italic",
    color="red")

plt.xlim(0,6000)

plt.ylim(0,3500)

plt.show()
```

But simple, coordinated plots are nicer!
Less is sometimes more.

Even if you **can** change everything, that does not mean that you **should**: Less is sometimes more. ❶




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

plt.xlabel("Purchase Amount")

plt.ylabel("Cost")

plt.title("Costs by Purchase
    Amount")

plt.xlim(0,6000)

plt.ylim(0,3500)

plt.show()
```

The simple plot requires less code which saves you time. ❷

Sidenote: Checklist for good graphics

This list is not exhaustive. Check the original source for the complete list.

- ❑ Does the chart clearly convey the intended message?
- ❑ Are both coordinate axes shown and labelled? Are they self-explanatory and concise?
- ❑ Is there a title for the chart? Is the title self-explanatory and concise?
- ❑ Are the scales and divisions clearly shown on both axes?
- ❑ Are the minimum and maximum of the ranges shown on the axes as appropriate to present the maximum information?
- ❑ Are the curves on a line chart individually labelled? The cells of a bar chart?
- ❑ Are all symbols properly explained? Are the units of measurement indicated?
- ❑ If the curves cross, are the line patterns different to avoid confusion?

Adapted from: Jain, R. (1991). Art of Computer Systems Performance Analysis Techniques For Experimental Design Measurements Simulation And Modeling, p. 156

Formatting plots