1. **Labels**

It's time to customize your own plot. This is the fun part, you will see your plot come to life!

You're going to work on the scatter plot with world development data: GDP per capita on the x-axis (logarithmic scale), life expectancy on the y-axis. The code for this plot is available in the script.

As a first step, let's add axis labels and a title to the plot. You can do this with the **[xlabel()](http://matplotlib.org/api/pyplot_api.html" \l "matplotlib.pyplot.xlabel" \t "_blank)**, **[ylabel()](http://matplotlib.org/api/pyplot_api.html" \l "matplotlib.pyplot.ylabel" \t "_blank)** and [**title()**](http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.title) functions, available in matplotlib.pyplot. This sub-package is already imported as plt.

* The strings xlab and ylab are already set for you. Use these variables to set the label of the x- and y-axis.
* The string title is also coded for you. Use it to add a title to the plot.
* After these customizations, finish the script with **[plt.show()](http://matplotlib.org/api/pyplot_api.html" \l "matplotlib.pyplot.show" \t "_blank)** to actually display the plot.

# Ticks

The customizations you've coded up to now are available in the script, in a more concise form.

In the video, Filip has demonstrated how you could control the y-ticks by specifying two arguments:

plt.yticks([0,1,2], ["one","two","three"])

In this example, the ticks corresponding to the numbers 0, 1 and 2 will be replaced by one, two and three, respectively.

Let's do a similar thing for the x-axis of your world development chart, with the **[xticks()](http://matplotlib.org/api/pyplot_api.html" \l "matplotlib.pyplot.xticks" \t "_blank)** function. The tick values 1000, 10000 and 100000 should be replaced by 1k, 10k and 100k. To this end, two lists have already been created for you: tick\_val and tick\_lab.

* Use tick\_val and tick\_lab as inputs to the **[xticks()](http://matplotlib.org/api/pyplot_api.html" \l "matplotlib.pyplot.xticks" \t "_blank)** function to make the the plot more readable.
* As usual, display the plot with **[plt.show()](http://matplotlib.org/api/pyplot_api.html" \l "matplotlib.pyplot.show" \t "_blank)** after you've added the customizations.

# Sizes

Right now, the scatter plot is just a cloud of blue dots, indistinguishable from each other. Let's change this. Wouldn't it be nice if the size of the dots corresponds to the population?

To accomplish this, there is a list pop loaded in your workspace. It contains population numbers for each country expressed in millions. You can see that this list is added to the scatter method, as the argument s, for size.

* Run the script to see how the plot changes.
* Looks good, but increasing the size of the bubbles will make things stand out more.
  + Import the numpy package as np.
  + Use np.array() to create a numpy array from the list pop. Call this Numpy array np\_pop.
  + Double the values in np\_pop by assigning np\_pop \* 2 to np\_popagain. Because np\_pop is a Numpy array, each array element will be doubled.
  + Change the s argument inside **[plt.scatter()](http://matplotlib.org/api/pyplot_api.html" \l "matplotlib.pyplot.scatter" \t "_blank)** to be np\_popinstead of pop.

# Colors

The code you've written up to now is available in the script on the right.

The next step is making the plot more colorful! To do this, a list col has been created for you. It's a list with a color for each corresponding country, depending on the continent the country is part of.

How did we make the list col you ask? The Gapminder data contains a list continent with the continent each country belongs to. A dictionary is constructed that maps continents onto colors:

dict = {

'Asia':'red',

'Europe':'green',

'Africa':'blue',

'Americas':'yellow',

'Oceania':'black'

}

Nothing to worry about now; you will learn about dictionaries in the next chapter.

* Add c = col to the arguments of the **[plt.scatter()](http://matplotlib.org/api/pyplot_api.html" \l "matplotlib.pyplot.scatter" \t "_blank)** function.
* Change the opacity of the bubbles by setting the alpha argument to 0.8inside **[plt.scatter()](http://matplotlib.org/api/pyplot_api.html" \l "matplotlib.pyplot.scatter" \t "_blank)**. Alpha can be set from zero to one, where zero is totally transparent, and one is not at all transparent.

# Additional Customizations

If you have another look at the script, under # Additional Customizations, you'll see that there are two **[plt.text()](http://matplotlib.org/api/pyplot_api.html" \l "matplotlib.pyplot.text" \t "_blank)** functions now. They add the words "India" and "China"in the plot.

* Add **[plt.grid(True)](http://matplotlib.org/api/pyplot_api.html" \l "matplotlib.pyplot.grid" \t "_blank)** after the **[plt.text()](http://matplotlib.org/api/pyplot_api.html" \l "matplotlib.pyplot.text" \t "_blank)** calls so that gridlines are drawn on the plot.

# Interpretation

If you have a look at your colorful plot, it's clear that people live longer in countries with a higher GDP per capita. No high income countries have really short life expectancy, and no low income countries have very long life expectancy. Still, there is a huge difference in life expectancy between countries on the same income level. Most people live in middle income countries where difference in lifespan is huge between countries; depending on how income is distributed and how it is used.

What can you say about the plot?

Answer is The countries in blue, corresponding to Africa, have both low life expectancy and a low GDP per capita.