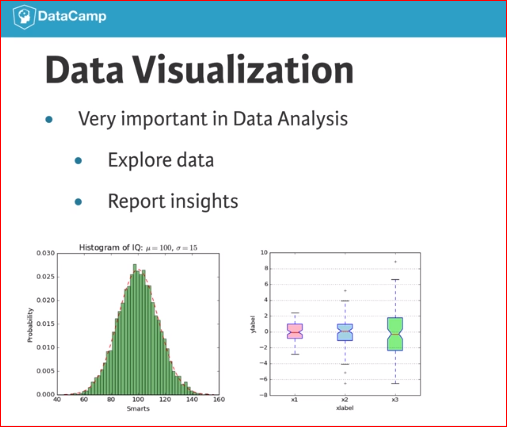
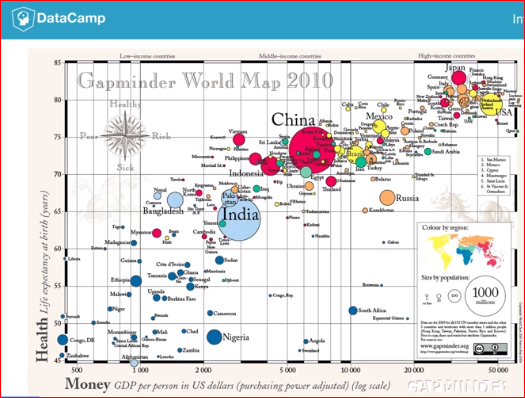
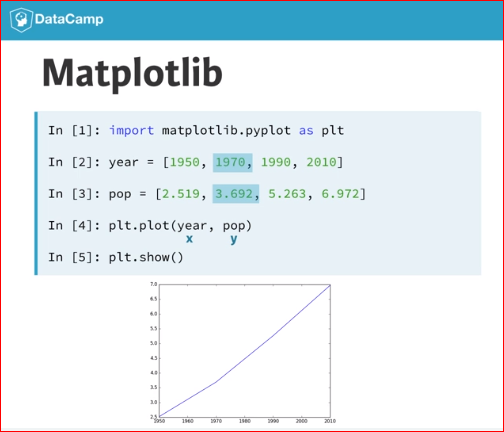
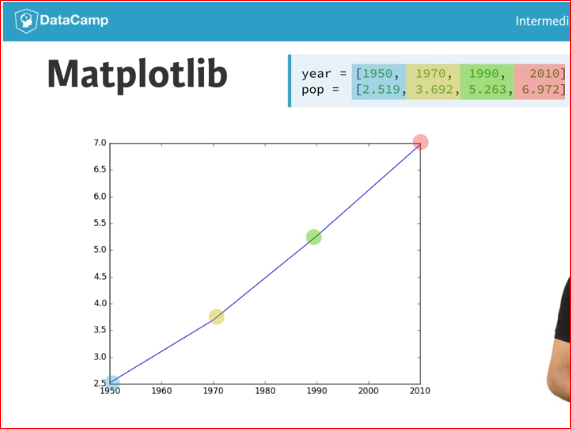
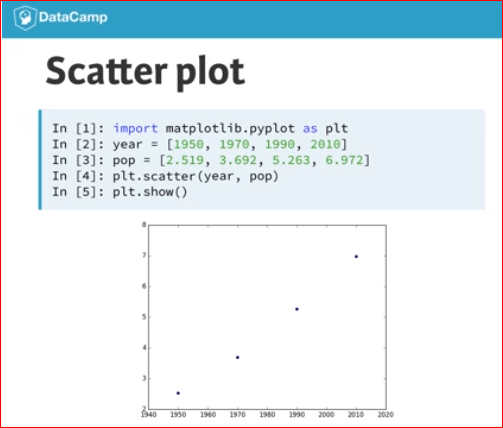
**Basic plots with matplotlib**

Instalar **matplotlib**

C:\WINDOWS\system32>pip install **matplotlib**

Line plot (1)

With matplotlib, you can create a bunch of different plots in Python. The most basic plot is the line plot. A general recipe is given here.

import matplotlib.pyplot as plt

plt.plot(x,y)

plt.show()

In the video, you already saw how much the world population has grown over the past years. Will it continue to do so? The world bank has estimates of the world population for the years 1950 up to 2100. The years are loaded in your workspace as a list called year, and the corresponding populations as a list called pop.

# Print the last item from year and pop

print (year[-1])

print (pop[-1])

# Import matplotlib.pyplot as plt

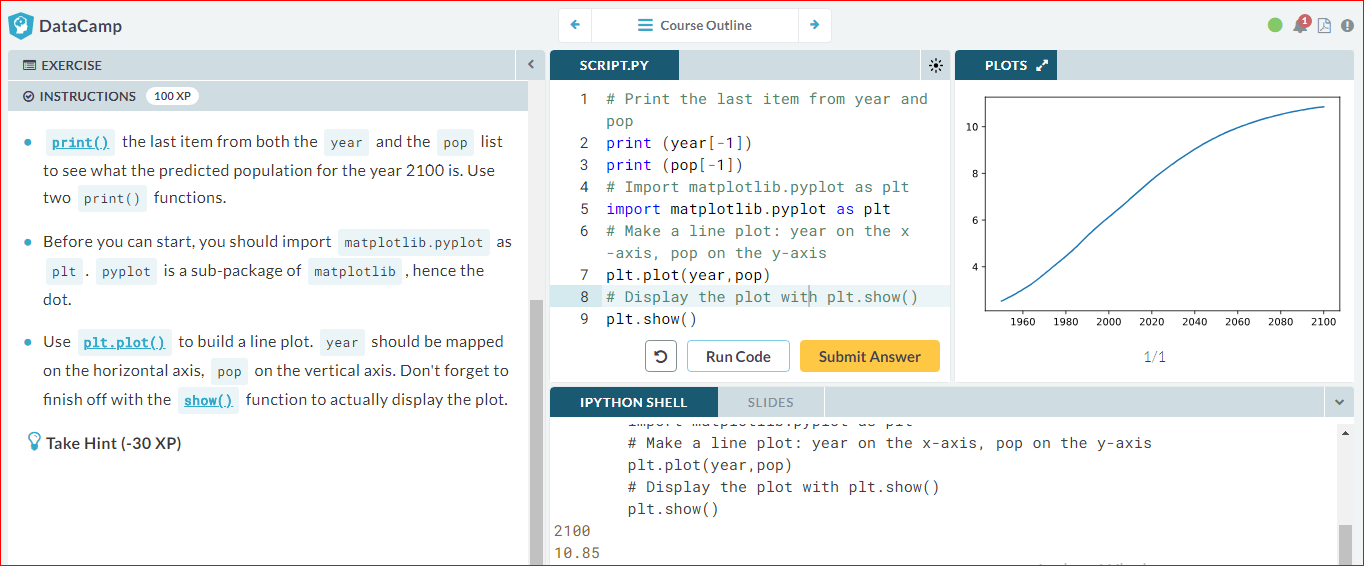
import matplotlib.pyplot as plt

# Make a line plot: year on the x-axis, pop on the y-axis

plt.plot(year,pop)

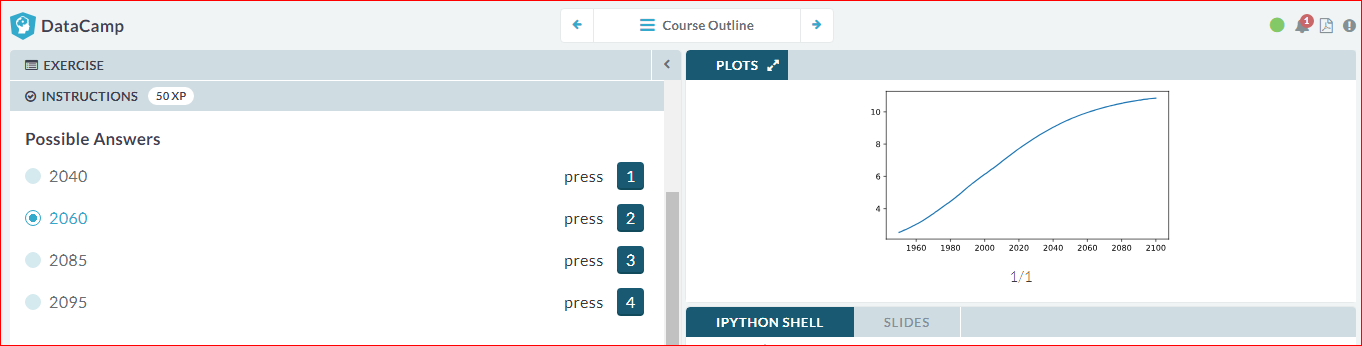
# Display the plot with plt.show()

plt.show()



# Line Plot (2): Interpretation

Have another look at the plot you created in the previous exercise; it's shown on the right. Based on the plot, in **approximately** what year will there be more than ten billion human beings on this planet?



Line plot (3)

Now that you've built your first line plot, let's start working on the data that professor Hans Rosling used to build his beautiful bubble chart. It was collected in 2007. Two lists are available for you:

* life\_exp which contains the life expectancy for each country and
* gdp\_cap, which contains the GDP per capita (i.e. per person) for each country expressed in US Dollars.

GDP stands for Gross Domestic Product. It basically represents the size of the economy of a country. Divide this by the population and you get the GDP per capita.

matplotlib.pyplot is already imported as plt, so you can get started straight away.

# Print the last item of gdp\_cap and life\_exp

**print(gdp\_cap[-1])**

**print(life\_exp[-1])**

# Import matplotlib.pyplot as plt

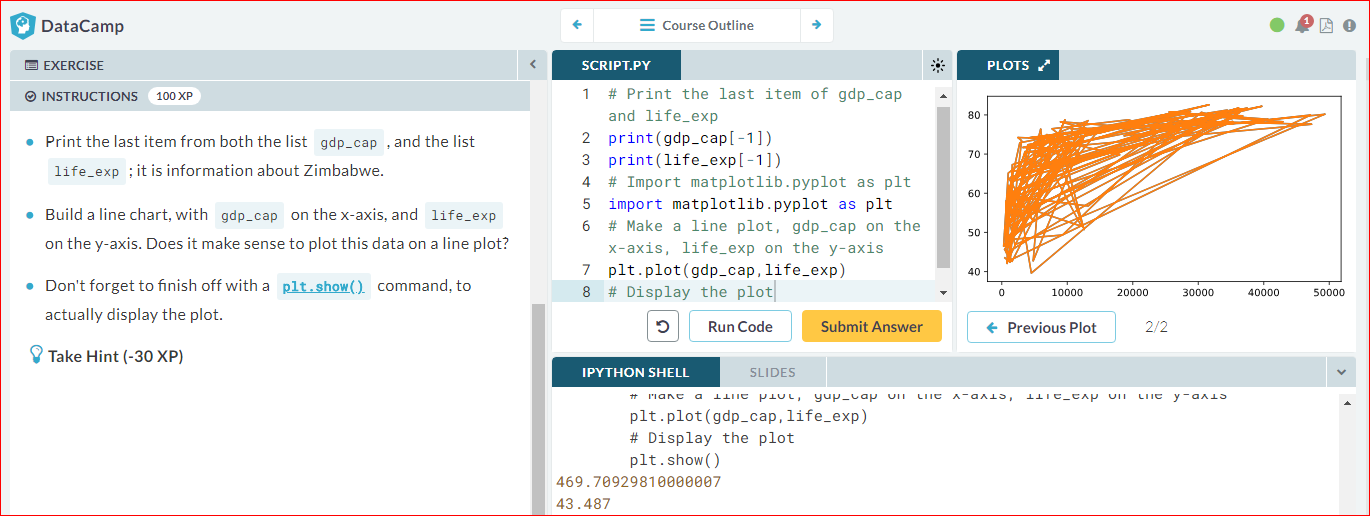
import matplotlib.pyplot as plt

# Make a line plot, gdp\_cap on the x-axis, life\_exp on the y-axis

plt.plot(gdp\_cap,life\_exp)

# Display the plot

plt.show()



# Scatter Plot (1)

When you have a time scale along the horizontal axis, the line plot is your friend. But in many other cases, when you're trying to assess if there's a correlation between two variables, for example, the scatter plot is the better choice. Below is an example of how to build a scatter plot.

import matplotlib.pyplot as plt

plt.scatter(x,y)

plt.show()

Let's continue with the gdp\_cap versus life\_exp plot, the GDP and life expectancy data for different countries in 2007. Maybe a scatter plot will be a better alternative?

Again, the matplotlib.pyplot package is available as plt.

# Change the line plot below to a scatter plot

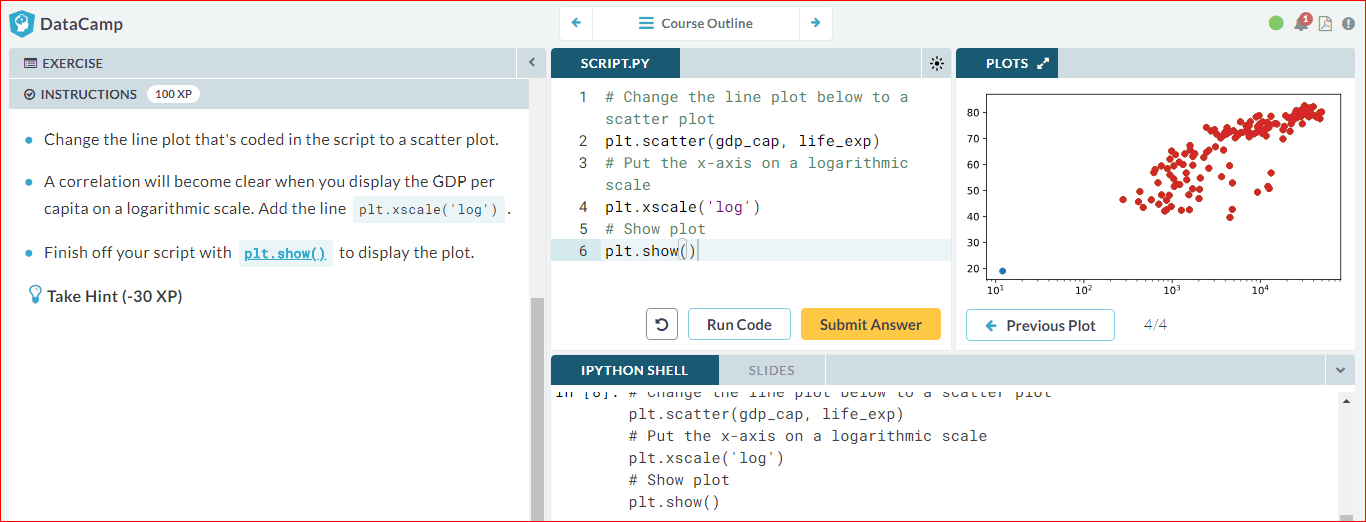
plt.scatter(gdp\_cap, life\_exp)

# Put the x-axis on a logarithmic scale

**plt.xscale('log')**

# Show plot

plt.show()



# Scatter plot (2)

In the previous exercise, you saw that that the higher GDP usually corresponds to a higher life expectancy. In other words, there is a positive correlation.

Do you think there's a relationship between population and life expectancy of a country? The list life\_exp from the previous exercise is already available. In addition, now also pop is available, listing the corresponding populations for the countries in 2007. The populations are in millions of people.

# Import package

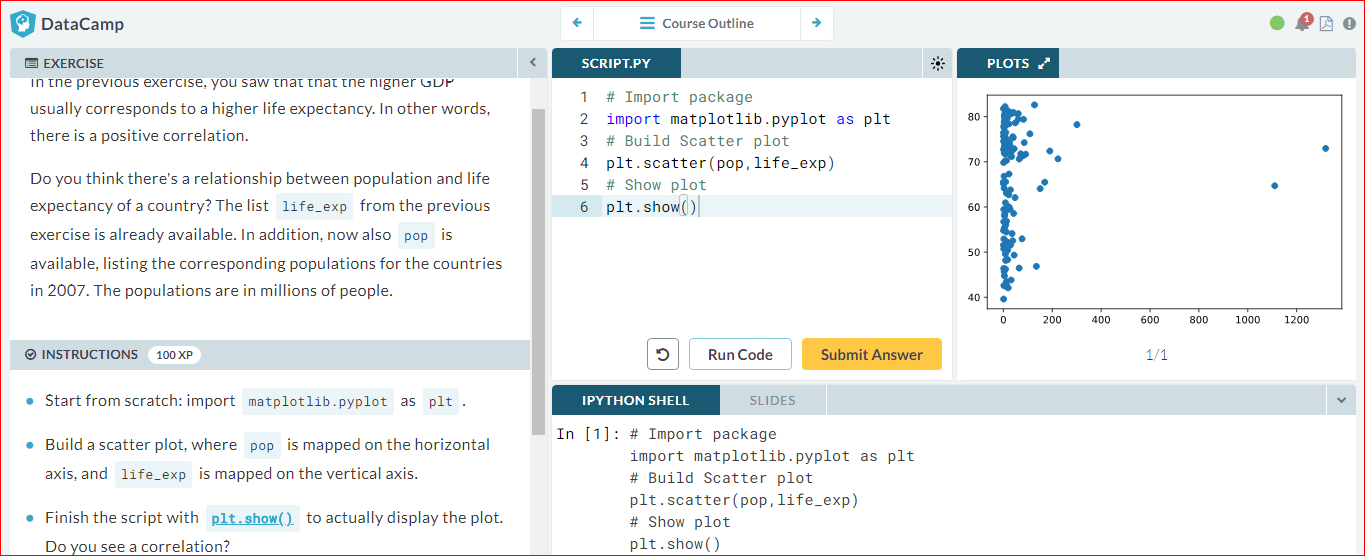
import matplotlib.pyplot as plt

# Build Scatter plot

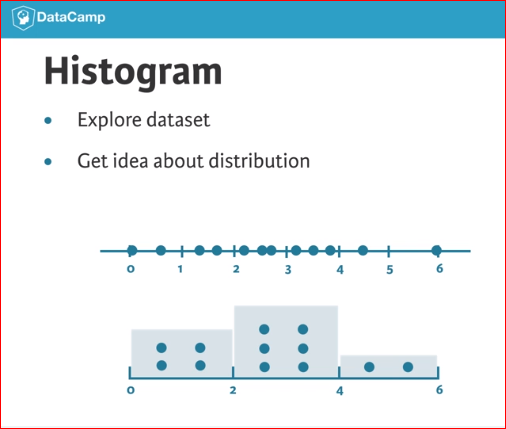
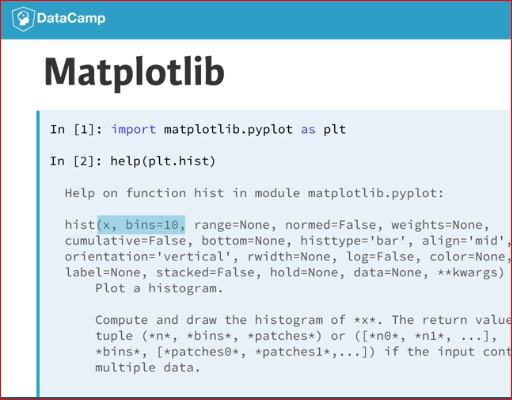
plt.scatter(**pop,life\_exp**)

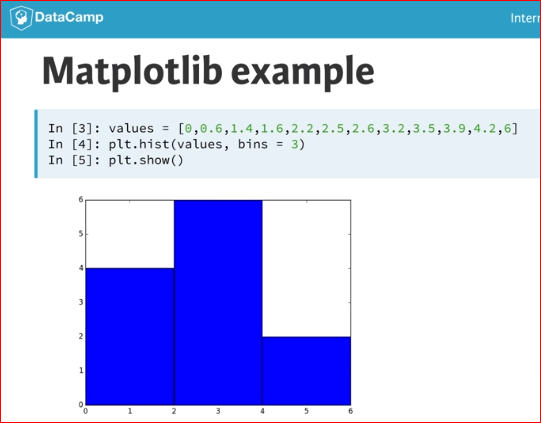
# Show plot

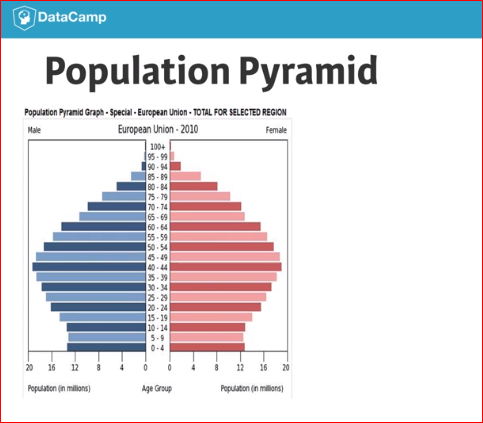
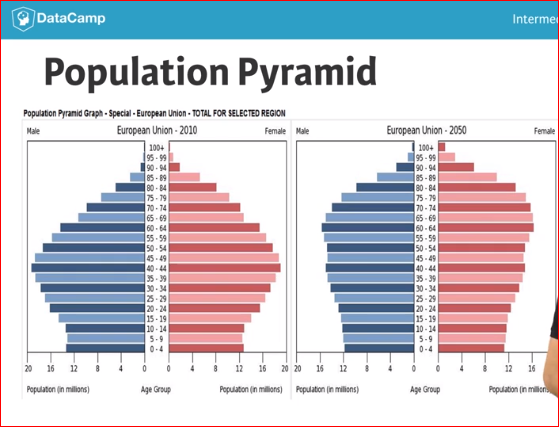
plt.show()



# Histograms



# Build a histogram (1)

life\_exp, the list containing data on the life expectancy for different countries in 2007, is available in your Python shell.

To see how life expectancy in different countries is distributed, let's create a histogram of life\_exp.

matplotlib.pyplot is already available as plt.

# Create histogram of life\_exp data

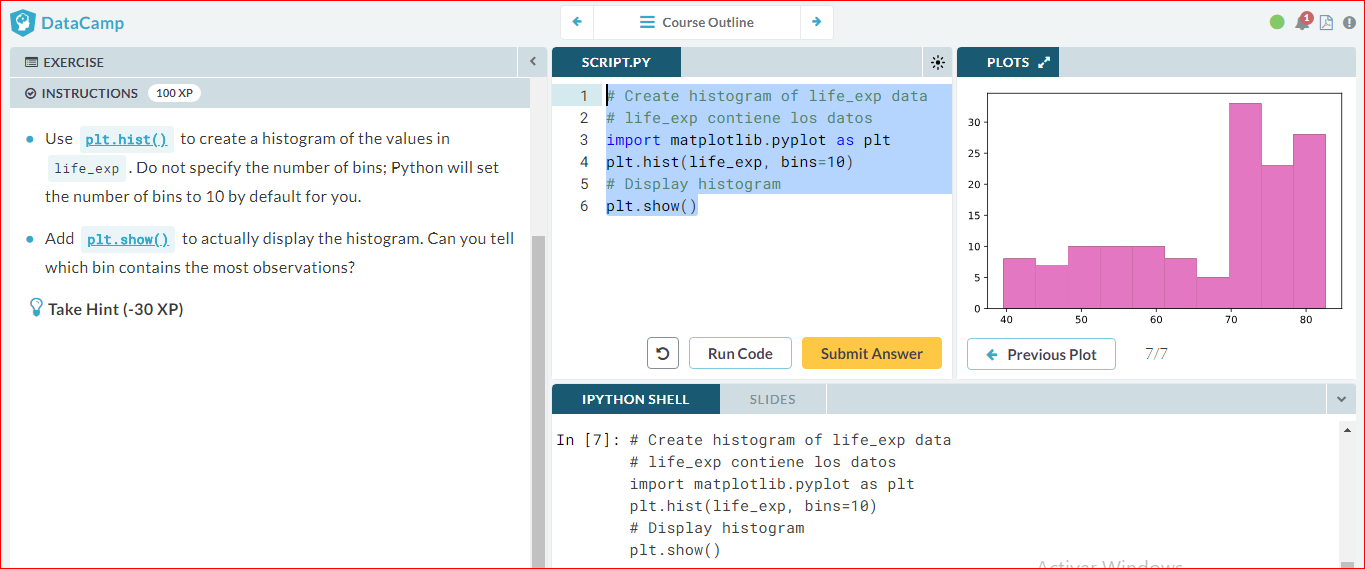
# life\_exp contiene los datos

import matplotlib.pyplot as plt

plt.hist(**life\_exp,** bins=10)

# Display histogram

plt.show()



# Build a histogram (2): bins

In the previous exercise, you didn't specify the number of bins. By default, Python sets the number of bins to 10 in that case. The number of bins is pretty important. Too few bins will oversimplify reality and won't show you the details. Too many bins will overcomplicate reality and won't show the bigger picture.

To control the number of bins to divide your data in, you can set the bins argument.

That's exactly what you'll do in this exercise. You'll be making two plots here. The code in the script already includes [plt.show()](http://matplotlib.org/api/pyplot_api.html" \l "matplotlib.pyplot.show" \t "_blank)and [plt.clf()](http://matplotlib.org/api/pyplot_api.html" \l "matplotlib.pyplot.clf" \t "_blank) calls; [plt.show()](http://matplotlib.org/api/pyplot_api.html" \l "matplotlib.pyplot.show" \t "_blank) displays a plot; [plt.clf()](http://matplotlib.org/api/pyplot_api.html" \l "matplotlib.pyplot.clf" \t "_blank)cleans it up again so you can start afresh.

As before, life\_exp is available and matplotlib.pyplot is imported as plt.

# Build histogram with 5 bins

# life\_exp contiene los datos

import matplotlib.pyplot as plt

plt.hist(life\_exp, **bins=5**)

# Show and clean up plot

plt.show()

plt.clf()

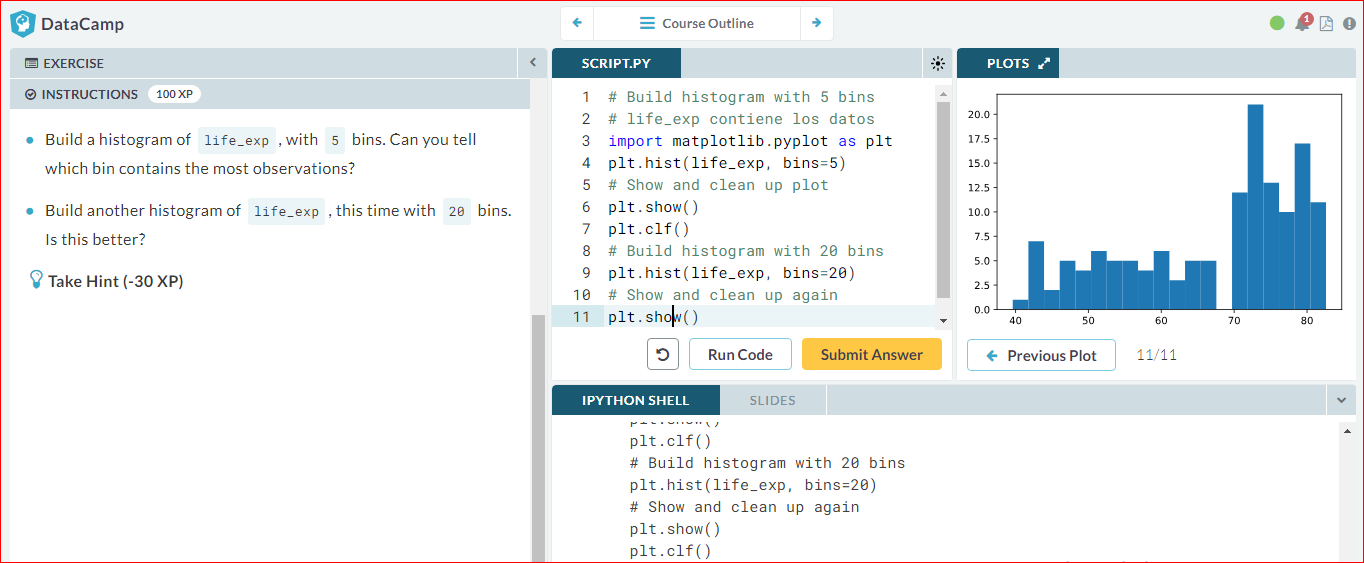
# Build histogram with 20 bins

plt.hist(life\_exp, **bins=20**)

# Show and clean up again

plt.show()

plt.clf()



# Build a histogram (3): compare

In the video, you saw population pyramids for the present day and for the future. Because we were using a histogram, it was very easy to make a comparison.

Let's do a similar comparison. life\_exp contains life expectancy data for different countries in 2007. You also have access to a second list now, life\_exp1950, containing similar data for 1950. Can you make a histogram for both datasets?

You'll again be making two plots. The [plt.show()](http://matplotlib.org/api/pyplot_api.html" \l "matplotlib.pyplot.show" \t "_blank) and [plt.clf()](http://matplotlib.org/api/pyplot_api.html" \l "matplotlib.pyplot.clf" \t "_blank) commands to render everything nicely are already included. Also matplotlib.pyplot is imported for you, as plt.

# Histogram of life\_exp, 15 bins

# life\_exp y life\_exp1950 contienen datos

import matplotlib.pyplot as plt

plt.hist(**life\_exp, bins=15**)

# Show and clear plot

plt.show()

plt.clf()

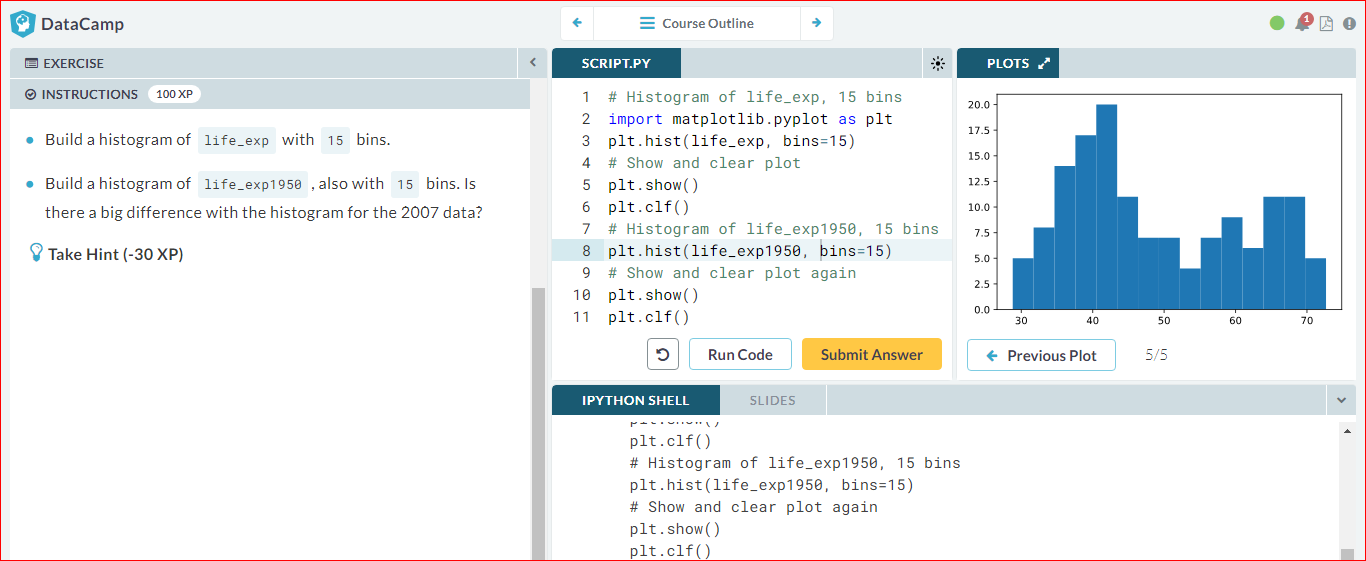
# Histogram of life\_exp1950, 15 bins

plt.hist(**life\_exp1950, bins=15**)

# Show and clear plot again

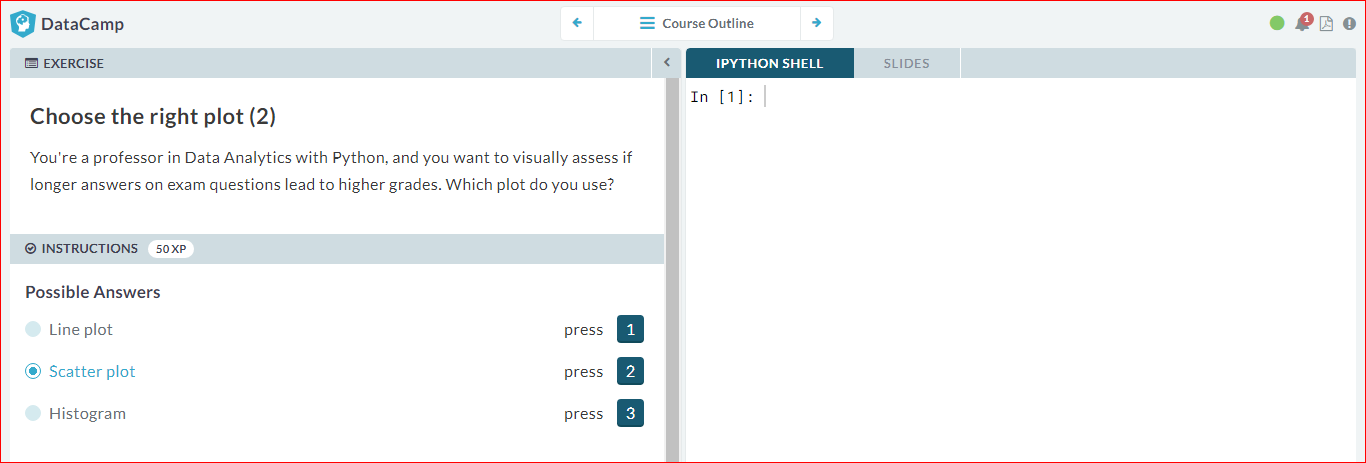
plt.show()

plt.clf()

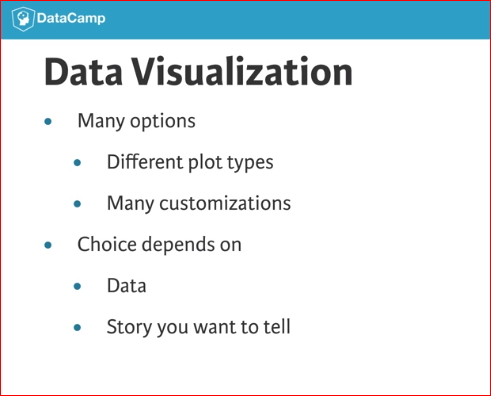


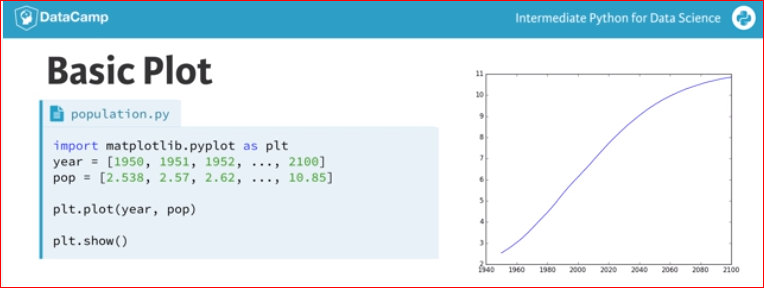
# Choose the right plot (1)

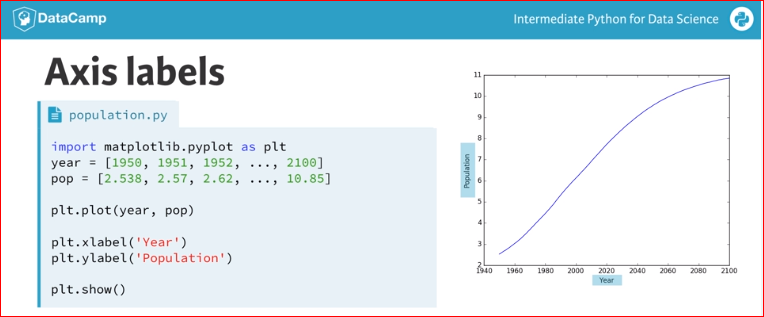
You're a professor teaching Data Science with Python, and you want to visually assess if the grades on your exam follow a particular distribution. Which plot do you use?

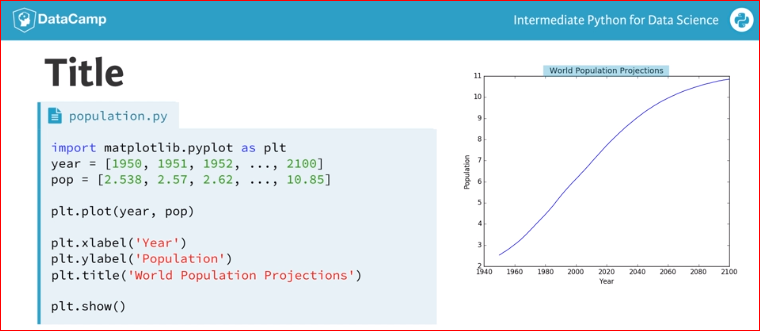


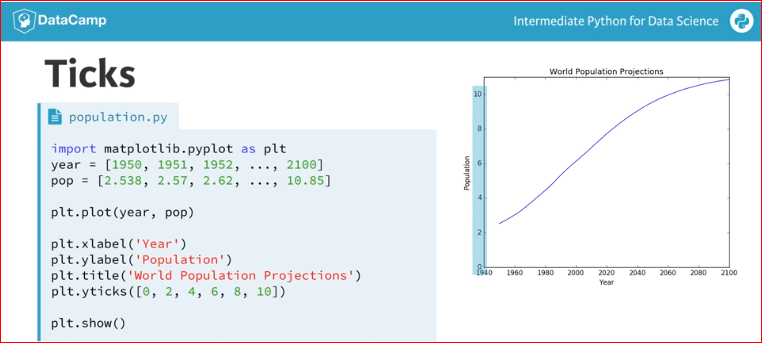
# Customization

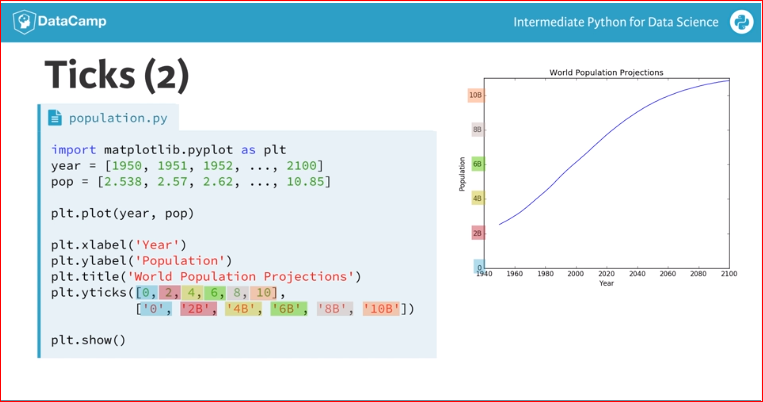


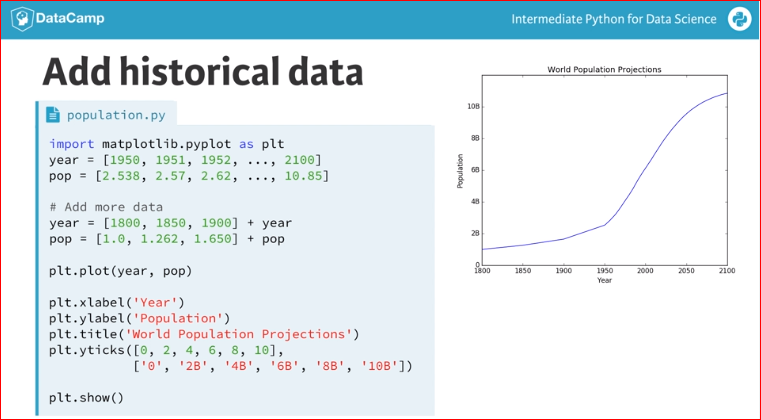


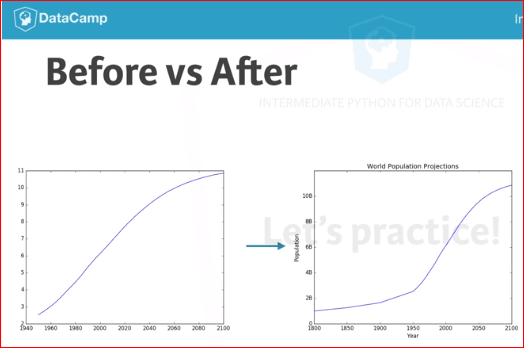












# Labels

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.

import matplotlib.pyplot as plt

# Basic scatter plot, log scale

plt.scatter(gdp\_cap, life\_exp)

plt.xscale('log')

# Strings

xlab = 'GDP per Capita [in USD]'

ylab = 'Life Expectancy [in years]'

title = 'World Development in 2007'

**# Add axis labels**

**plt.xlabel(xlab)**

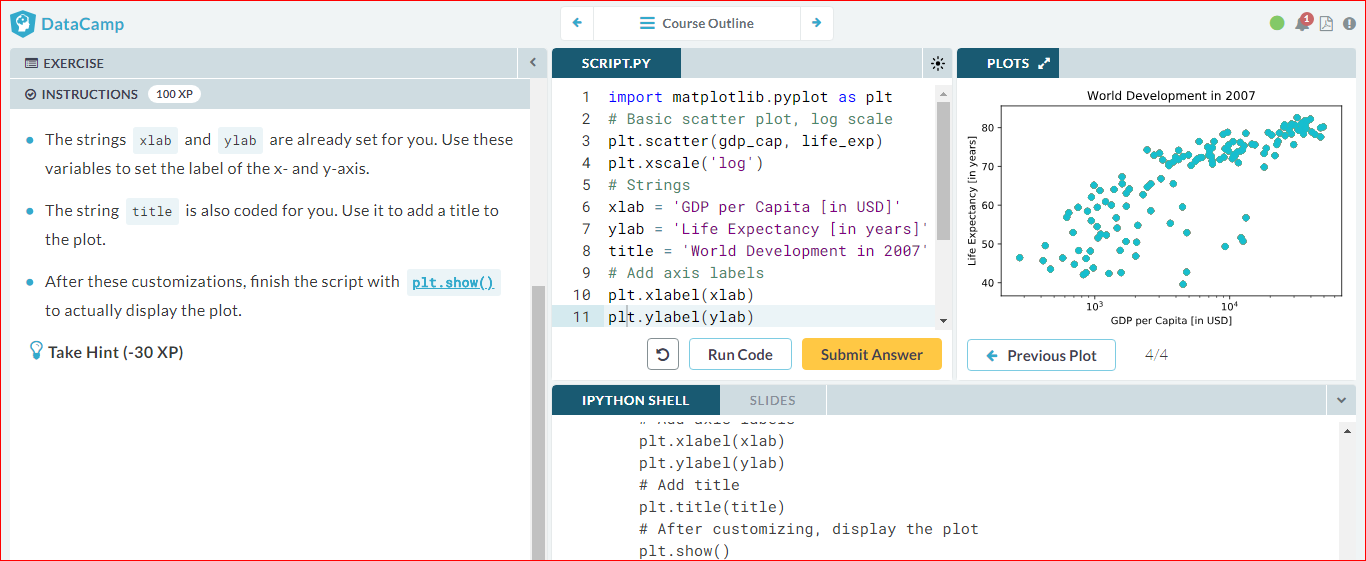
**plt.ylabel(ylab)**

**# Add title**

**plt.title(title)**

# After customizing, display the plot

plt.show()



# 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.

import matplotlib.pyplot as plt

# Scatter plot

plt.scatter(gdp\_cap, life\_exp)

# Previous customizations

plt.xscale('log')

plt.xlabel('GDP per Capita [in USD]')

plt.ylabel('Life Expectancy [in years]')

plt.title('World Development in 2007')

# Definition of tick\_val and tick\_lab

tick\_val = [1000,10000,100000]

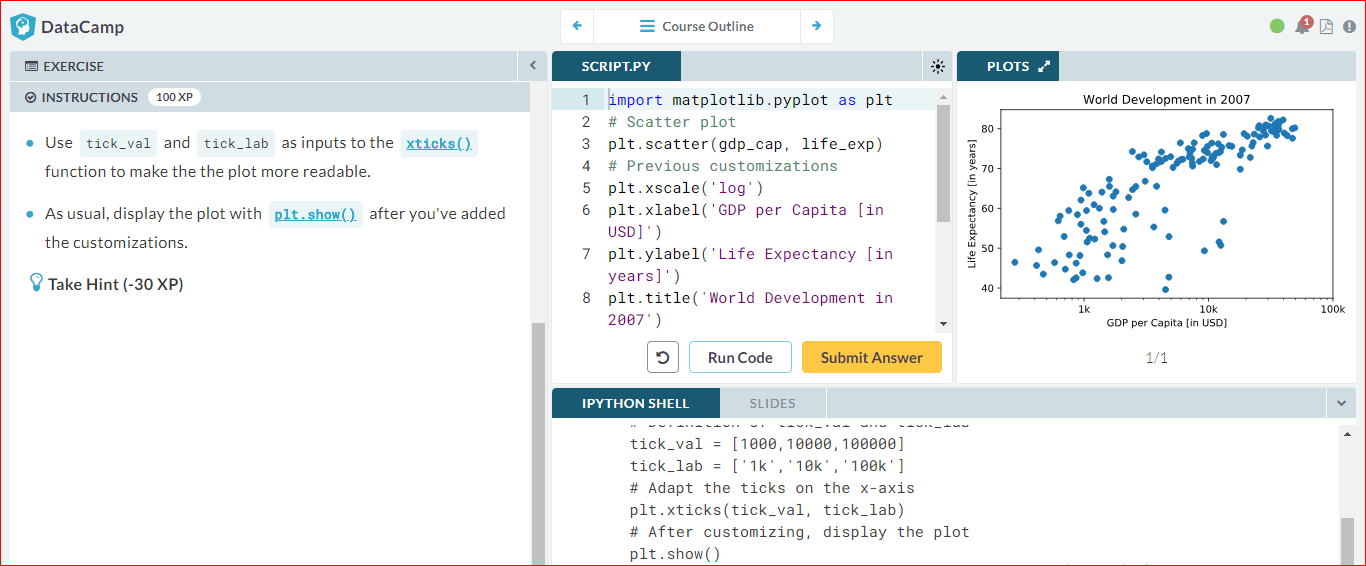
tick\_lab = ['1k','10k','100k']

**# Adapt the ticks on the x-axis**

**plt.xticks(tick\_val, tick\_lab)**

# After customizing, display the plot

plt.show()



# 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.

# Import numpy as np

import numpy as np

# Store pop as a numpy array: np\_pop

np\_pop = np.array(pop)

# Double np\_pop

np\_pop = np\_pop \* 2

**# Update: set s argument to np\_pop**

plt.scatter(gdp\_cap, life\_exp, **s = pop**)

# Previous customizations

plt.xscale('log')

plt.xlabel('GDP per Capita [in USD]')

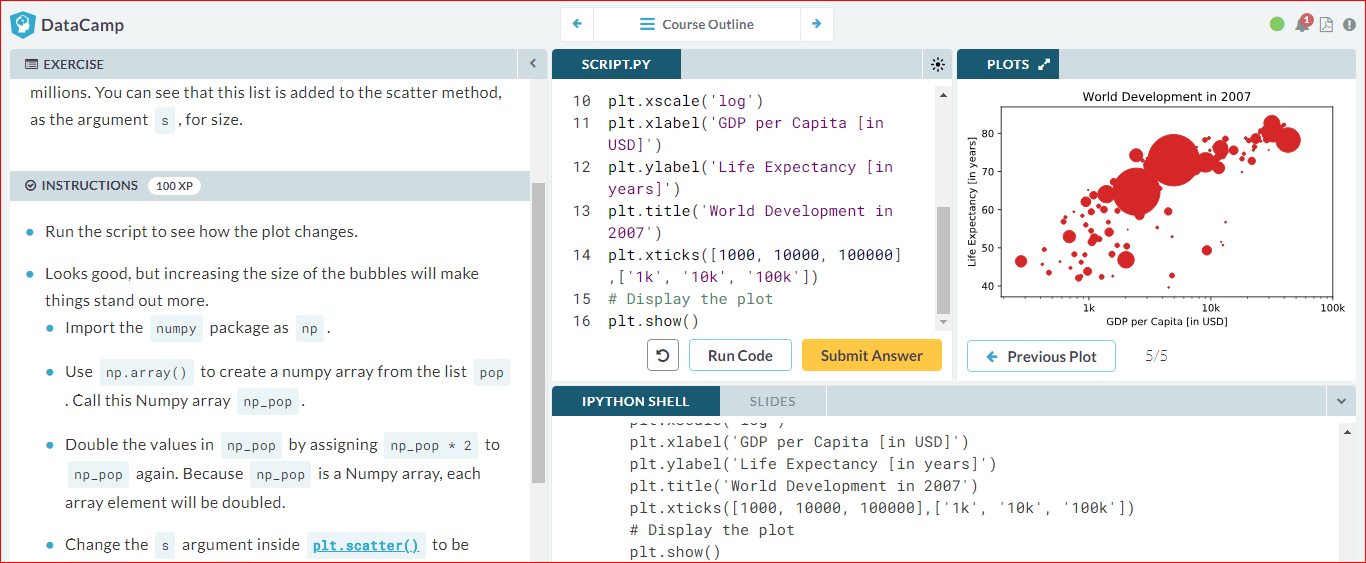
plt.ylabel('Life Expectancy [in years]')

plt.title('World Development in 2007')

plt.xticks([1000, 10000, 100000],['1k', '10k', '100k'])

# Display the plot

plt.show()



# 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.

# Import numpy as np

import numpy as np

dict = {

'Asia':'red',

'Europe':'green',

'Africa':'blue',

'Americas':'yellow',

'Oceania':'black'

}

**# Specify c and alpha inside plt.scatter()**

plt.scatter(x = gdp\_cap, y = life\_exp, s = np.array(pop) \* 2, **c = col, alpha = 0.8**)

# Previous customizations

plt.xscale('log')

plt.xlabel('GDP per Capita [in USD]')

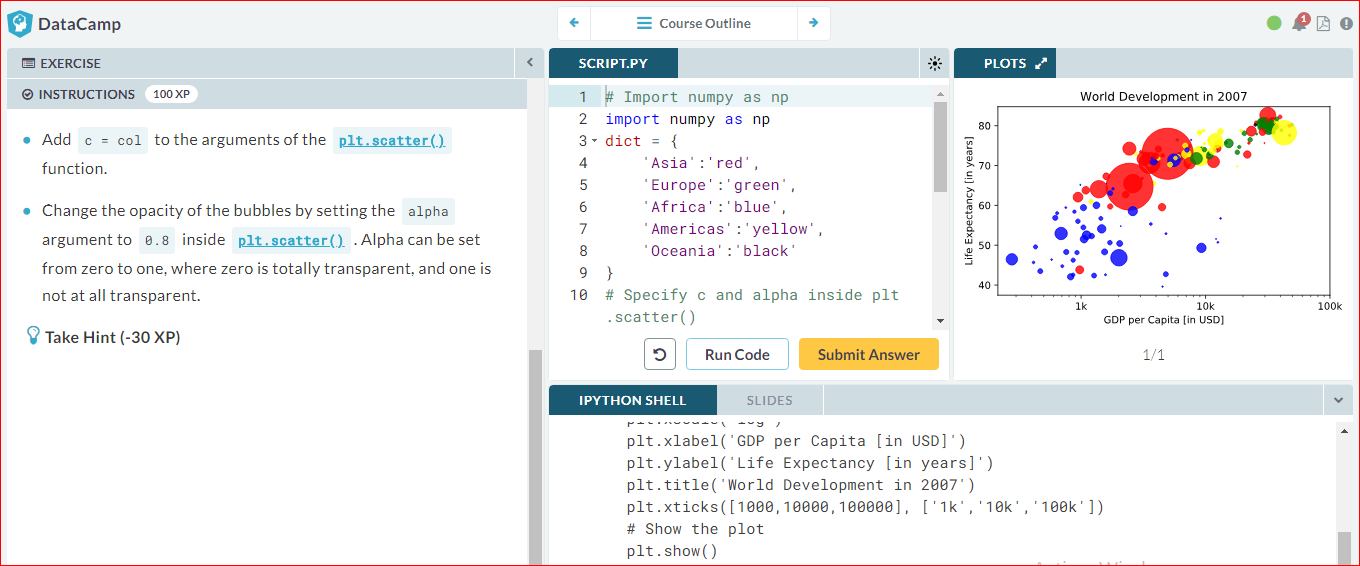
plt.ylabel('Life Expectancy [in years]')

plt.title('World Development in 2007')

plt.xticks([1000,10000,100000], ['1k','10k','100k'])

# Show the plot

plt.show()



# 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.

# Import numpy as np

import numpy as np

dict = {

'Asia':'red',

'Europe':'green',

'Africa':'blue',

'Americas':'yellow',

'Oceania':'black'

}

# Scatter plot

plt.scatter(x = gdp\_cap, y = life\_exp, s = np.array(pop) \* 2, c = col, alpha = 0.8)

# Previous customizations

plt.xscale('log')

plt.xlabel('GDP per Capita [in USD]')

plt.ylabel('Life Expectancy [in years]')

plt.title('World Development in 2007')

plt.xticks([1000,10000,100000], ['1k','10k','100k'])

# Additional customizations

plt.text(1550, 71, 'India')

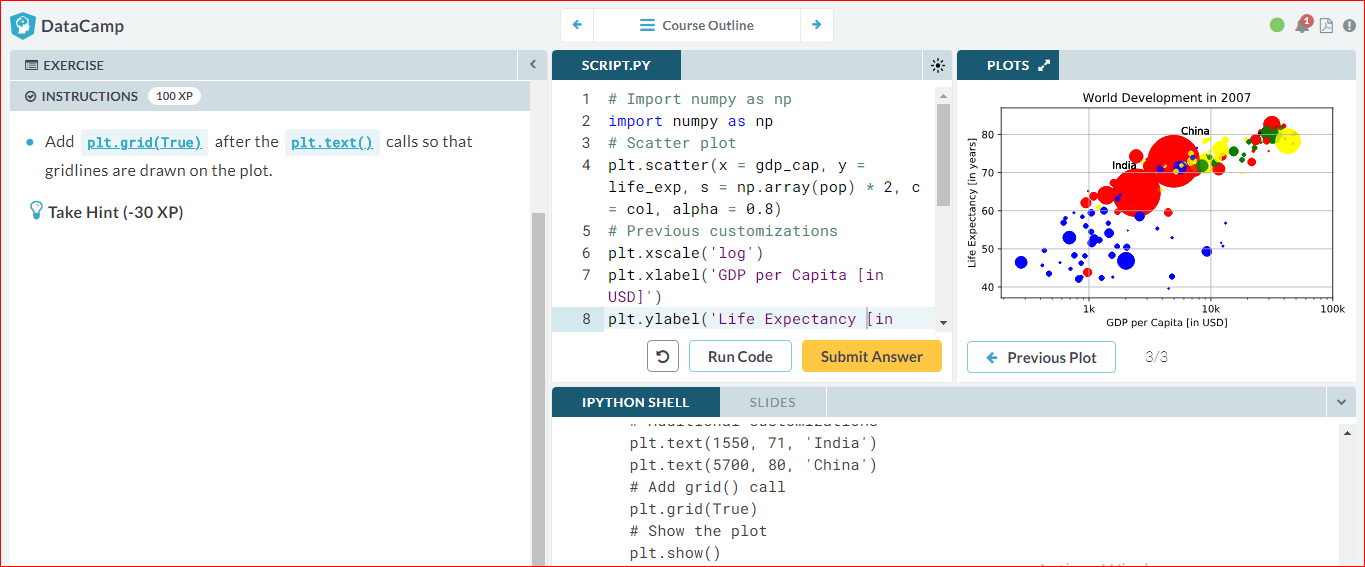
plt.text(5700, 80, 'China')

**# Add grid() call**

**plt.grid(True)**

# Show the plot

plt.show()



# 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?

