**Intermediate Python for Data Science**

**#1 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.

*This course touches on a lot of concepts you may have forgotten, so if you ever need a quick refresher, download the*[***Python for data science Cheat Sheet***](https://datacamp-community-prod.s3.amazonaws.com/e30fbcd9-f595-4a9f-803d-05ca5bf84612)*and keep it handy!*

**Instructions**

**100 XP**

**Instructions**

**100 XP**

* [**print()**](https://docs.python.org/3/library/functions.html#print) the last item from both the year and the pop list to see what the predicted population for the year 2100 is. Use two print() functions.
* Before you can start, you should import matplotlib.pyplot as plt. pyplot is a sub-package of matplotlib, hence the dot.
* Use [**plt.plot()**](http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.plot) to build a line plot. year should be mapped on the horizontal axis, pop on the vertical axis. Don't forget to finish off with the [**show()**](http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.show) function to actually display the plot.

Script.py  
01 # Print the last item from year and pop

02 print(year[-1])

03 print(pop[-1])

04

05 # Import matplotlib.pyplot as plt

06 import matplotlib.pyplot as plt

07

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

09 plt.plot(year,pop)

10

11 # Display the plot with plt.show()

12 plt.show()

IPython Shell  
# 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?

**Instructions**

**50 XP**

**Possible Answers**

* 

2040

* 

2060

* 

2085

* 

2095

**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.

**Instructions**

**100 XP**

* Print the last item from both the list gdp\_cap, and the list life\_exp; it is information about Zimbabwe.
* Build a line chart, with gdp\_cap on the x-axis, and life\_exp on the y-axis. Does it make sense to plot this data on a line plot?
* Don't forget to finish off with a plt.show() command, to actually display the plot.

Script.py  
01 # Print the last item of gdp\_cap and life\_exp

02 print(gdp\_cap[-1])

03 print(life\_exp[-1])

04

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

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

07

08 # Display the plot

09 plt.show()

IPython Shell  
In [1]: # Print the last item of gdp\_cap and life\_exp

print(gdp\_cap[-1])

print(life\_exp[-1])

# 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()

469.70929810000007

43.487

<script.py> output:

469.70929810000007

43.487

In [2]:

**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.

**Instructions**

**100 XP**

* Change the line plot that's coded in the script to a scatter plot.
* A correlation will become clear when you display the GDP per capita on a logarithmic scale. Add the line plt.xscale('log').
* Finish off your script with plt.show() to display the plot.

Script.py  
01 # Change the line plot below to a scatter plot

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

03

04 # Put the x-axis on a logarithmic scale

05 plt.xscale('log')

06

07 # Show plot

08 plt.show()

IPython Shell  
In [1]: # 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()

In [2]:

**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.

**Instructions**

**100 XP**

* Start from scratch: import matplotlib.pyplot as plt.
* Build a scatter plot, where pop is mapped on the horizontal axis, and life\_exp is mapped on the vertical axis.
* Finish the script with plt.show() to actually display the plot. Do you see a correlation?

Script.py  
01 # Import package

02 import matplotlib.pyplot as plt

03

04 # Build Scatter plot

05 plt.scatter(pop, life\_exp)

06

07 # Show plot

08 plt.show()

IPython Shell  
In [1]: # Import package

import matplotlib.pyplot as plt

# Build Scatter plot

plt.scatter(pop, life\_exp)

# Show plot

plt.show()

In [2]:

**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.

**Instructions**

**100 XP**

* Use [**plt.hist()**](http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.hist) to create a histogram of the values in life\_exp. Do not specify the number of bins; Python will set the number of bins to 10 by default for you.
* Add plt.show() to actually display the histogram. Can you tell which bin contains the most observations?

Script.py  
1 # Create histogram of life\_exp data

2 plt.hist(life\_exp)

3

4 # Display histogram

5 plt.show()

IPython Shell  
In [1]: # Create histogram of life\_exp data

plt.hist(life\_exp)

# Display histogram

plt.show()

In [2]:

**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() and plt.clf() calls; plt.show() displays a plot; plt.clf() cleans it up again so you can start afresh.

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

**Instructions**

**100 XP**

* Build a histogram of life\_exp, with 5 bins. Can you tell which bin contains the most observations?
* Build another histogram of life\_exp, this time with 20 bins. Is this better?

Script.py  
01 # Build histogram with 5 bins

02 plt.hist(life\_exp, 5)

03

04 # Show and clean up plot

05 plt.show()

06 plt.clf()

07

08 # Build histogram with 20 bins

09 plt.hist(life\_exp, 20)

10

11 # Show and clean up again

12 plt.show()

13 plt.clf()

IPython Shell  
In [1]: # Build histogram with 5 bins

plt.hist(life\_exp, 5)

# Show and clean up plot

plt.show()

plt.clf()

# Build histogram with 20 bins

plt.hist(life\_exp, 20)

# Show and clean up again

plt.show()

plt.clf()

In [2]:

**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() and plt.clf() commands to render everything nicely are already included. Also matplotlib.pyplot is imported for you, as plt.

**Instructions**

**100 XP**

* Build a histogram of life\_exp with 15 bins.
* Build a histogram of life\_exp1950, also with 15bins. Is there a big difference with the histogram for the 2007 data?

Script.py  
01 # Histogram of life\_exp, 15 bins

02 plt.hist(life\_exp, 15)

03

04 # Show and clear plot

05 plt.show()

06 plt.clf()

07

08 # Histogram of life\_exp1950, 15 bins

09 plt.hist(life\_exp1950, 15)

10

11 # Show and clear plot again

12 plt.show()

13 plt.clf()

IPython Shell  
In [1]: # Histogram of life\_exp, 15 bins

plt.hist(life\_exp, 15)

# Show and clear plot

plt.show()

plt.clf()

# Histogram of life\_exp1950, 15 bins

plt.hist(life\_exp1950, 15)

# Show and clear plot again

plt.show()

plt.clf()

In [2]:

**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?

**Instructions**

**50 XP**

**Instructions**

**50 XP**

**Possible Answers**

* 

Line plot

* 

Scatter plot

* 

Histogram

**Choose the right plot (2)**

You're a professor in Data Analytics with Python, and you want to visually assess if longer answers on exam questions lead to higher grades. Which plot do you use?

**Instructions**

**50 XP**

**Possible Answers**

* 

Line plot

* 

Scatter plot

* 

Histogram

**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#matplotlib.pyplot.xlabel), [**ylabel()**](http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.ylabel) 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.

**Instructions**

**100 XP**

* 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() to actually display the plot.

Script.py  
01 # Basic scatter plot, log scale

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

03 plt.xscale('log')

04

05 # Strings

06 xlab = 'GDP per Capita [in USD]'

07 ylab = 'Life Expectancy [in years]'

08 title = 'World Development in 2007'

09

10 # Add axis labels

11 plt.xlabel(xlab)

12 plt.ylabel(ylab)

13

14 # Add title

15 plt.title(title)

16

17 # After customizing, display the plot

18 plt.show()

IPython Shell  
In [1]: # 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()

In [2]:

**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#matplotlib.pyplot.xticks) 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.

**Instructions**

**100 XP**

* Use tick\_val and tick\_lab as inputs to the [**xticks()**](http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.xticks) function to make the the plot more readable.
* As usual, display the plot with plt.show() after you've added the customizations.

Script.py  
01 # Scatter plot

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

03

04 # Previous customizations

05 plt.xscale('log')

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

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

08 plt.title('World Development in 2007')

09

10 # Definition of tick\_val and tick\_lab

11 tick\_val = [1000, 10000, 100000]

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

13

14 # Adapt the ticks on the x-axis

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

16

17 # After customizing, display the plot

18 plt.show()

IPython Shell  
In [1]: # 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.

**Instructions**

**100 XP**

**Instructions**

**100 XP**

* 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\_pop again. Because np\_popis a Numpy array, each array element will be doubled.
  + Change the s argument inside [**plt.scatter()**](http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.scatter) to be np\_pop instead of pop.

Script.py  
01 # Import numpy as np

02 import numpy as np

03

04 # Store pop as a numpy array: np\_pop

05 np\_pop = np.array(pop)

06

07 # Double np\_pop

08 np\_pop = np\_pop \* 2

09

10 # Update: set s argument to np\_pop

11 plt.scatter(gdp\_cap, life\_exp, s = np\_pop)

12

13 # Previous customizations

14 plt.xscale('log')

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

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

17 plt.title('World Development in 2007')

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

19

20 # Display the plot

21 plt.show()

IPython  
In [1]: # 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 = np\_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()

In [2]:

**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.

**Instructions**

**100 XP**

**Instructions**

**100 XP**

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

Script.py  
01 # Specify c and alpha inside plt.scatter()

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

03

04 # Previous customizations

05 plt.xscale('log')

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

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

08 plt.title('World Development in 2007')

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

10

11 # Show the plot

12 plt.show()

In [1]: # 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()

In [2]:

**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#matplotlib.pyplot.text) functions now. They add the words "India" and "China" in the plot.

**Instructions**

**100 XP**

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

Script.py,  
# 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()

TPython Shell  
In [1]: # 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()

In [2]:

**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?

**Instructions**

**50 XP**

**Possible Answers**

* 

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

* 

There is a negative correlation between GDP per capita and life expectancy.

* 

China has both a lower GDP per capita and low expectancy compared to India.

Script.py