KEY_Lesson17_Matplotlib_Intro

July 16, 2019

1 Matplotlib Introduction

Data visualization is one of the best ways to communicate your data with others as well as to get insights into data. Today we are going to learn how to use Python's `matplotlib` package to visualize and interpret data.

In this lesson, we want to see how the the the number of passengers of a flight changes over time.

pyplot odule is an interface for matplotlib that let's you work on your plot interactively. To import this module we would reference it using `matplotlib.pyplot`.

[0]:
```
# load matplotlib.pyplot
import matplotlib.pyplot
```

As we've seen before, it is a pain to use long prefixes whenever we want to use a function from a package with a long name, and when adding a reference to a submodule it gets even longer. So, we'll want to create another **nickname** for this package. For `matplotlib.pyplot`, it is common to import using the nickname `plt`, so that's what we'll do from now on.

[0]:
```
# load matplotlib.pyplot as plt
import matplotlib.pyplot as plt
```

There is another popular plotting package in python called **seaborn**. Though most of our plotting lessons will focus on **matplotlib**, the **seaborn** package has a nice feature that procides a number of built-in datasets as pandas dataframes, which can be useful when practicing with new skills like we're about do to with plotting. We will use the nickname `sns` to reference the seaborn package.

For this lesson we are going to use a **seaborn** built-in dataset called "flights". This dataset includes the number of passangers of flights during each month from years 1949 to 1960.

We'll use the function `load_dataset` to load "flights" dataset from **seaborn** and save it to a variable called `flights`.

[0]:
```
# import seaborn and name it sns
# load flights dataset from seaborn and assign it to a variable called flights
import seaborn as sns
flights = sns.load_dataset("flights")
```

Based on the description above, what type do you think the `flights` object is?

[4]:
```
print(type(flights))
```

```
<class 'pandas.core.frame.DataFrame'>
```

Let's look at the data to get a better understanding of it.

[5]:
```
# return the first 15 rows of flights dataframe
flights.head(15)
```

[5]:

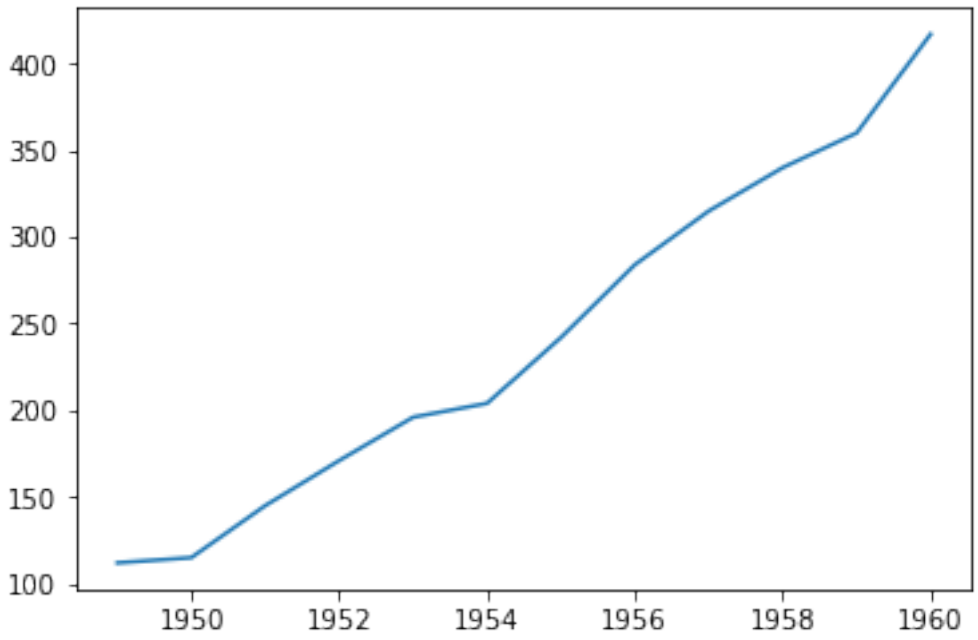
	year	month	passengers
0	1949	January	112
1	1949	February	118
2	1949	March	132
3	1949	April	129
4	1949	May	121
5	1949	June	135
6	1949	July	148
7	1949	August	148
8	1949	September	136
9	1949	October	119
10	1949	November	104
11	1949	December	118
12	1950	January	115
13	1950	February	126
14	1950	March	141

How do you think the number of passengers in January has changed from year 1949 to 1960? It is difficult to tell buy just looking at the table above. An easier way to get a sense of the trend over time is to **plot our data**, which is super easy with the **matplotlib** package.

For this specific example, we will use the number of passengers on flights at the start of the year to measure change over time. To do this, we will first use our pandas skills to select the rows in our dataframe `flights` where the `month` column has the value `'January'` and save this to a new dataframe called `flights_January`. Then we will plot these data using the `plt.plot` function.

[6]:
```
# select all samples where month is January and assign it to a new dataframe␣
 ↪named "flights_January"
# use plt.plot function to plot the number of passenger for each year.
# we want to show year on a horizonal axis (also called x-axis) and
# show the number of passenger on the vertical axis (also called y-axis)
flights_January = flights[flights['month'] == "January"]
plt.plot(flights_January["year"], flights_January["passengers"])
```

[6]:
```
[<matplotlib.lines.Line2D at 0x7fe42268b630>]
```



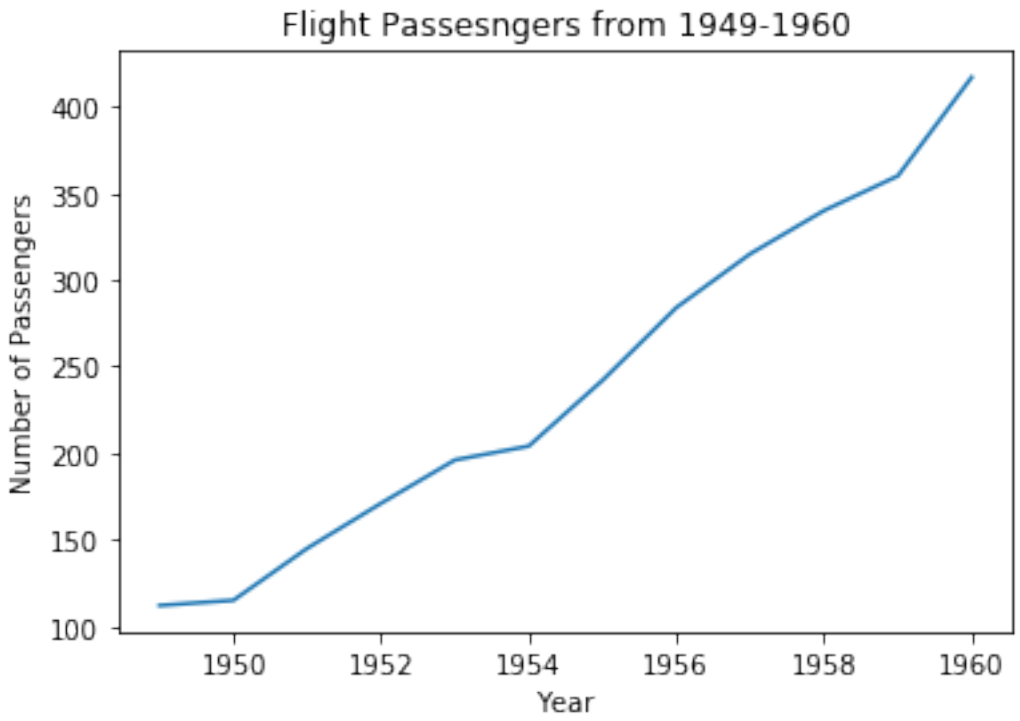
Yay! You just made your first plot! This type of plot is called **line graph**. Line graph is a simple and at the same time powerful tool to show changes of a variable over time!

But is there anything missing from the plot? Is this plot self explanotory?

Not really right? No worries - we can fix that! To help people better understand a plot, we should add a proper title and labels to the plot. For that, let's explore the `plt.title`, `plt.xlabel`, and `plt.ylabel` functions.

[7]:
```
# reproduce the above plot
# but this time, add the title "Flight Passesnger from 1949-1960"
# label the horizontal and verrtical axes "Year" and "Number of Passengers"␣
 ↪respectively
plt.plot(flights_January["year"], flights_January["passengers"])
plt.title("Flight Passesngers from 1949-1960")
plt.xlabel("Year")
plt.ylabel("Number of Passengers")
# Remember that labels/title are string variables.
```

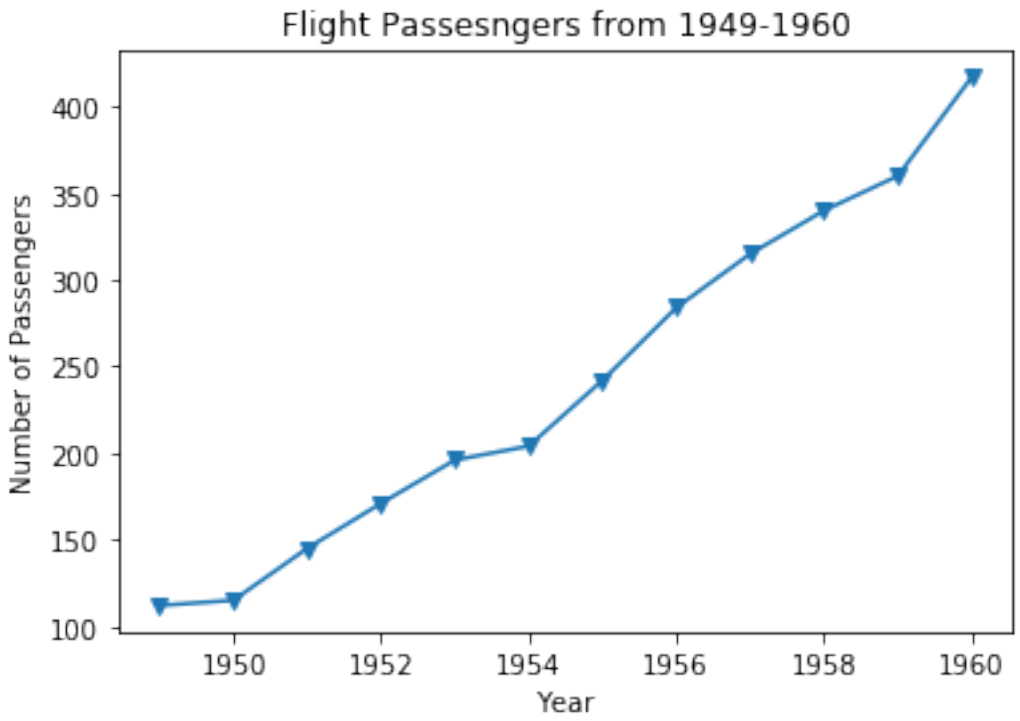
[7]:
```
Text(0, 0.5, 'Number of Passengers')
```



The line graph seems to be continous, however, it is actually measured at fixed intervals rather than continuously. In this plot, we are showing yearly intervals. Adding markers is a way to show at which points exactly along the x-axis the data are measured, and which parts correspond to the trends between the measurement intervals.

[8]:
```
# reproduce the above plot
# this time, pass "v" to markers to add a triangle shaped markers
# don't forget to add labels/title
plt.plot(flights_January["year"], flights_January["passengers"], marker = "v")
plt.title("Flight Passesngers from 1949-1960")
plt.xlabel("Year")
plt.ylabel("Number of Passengers")
```

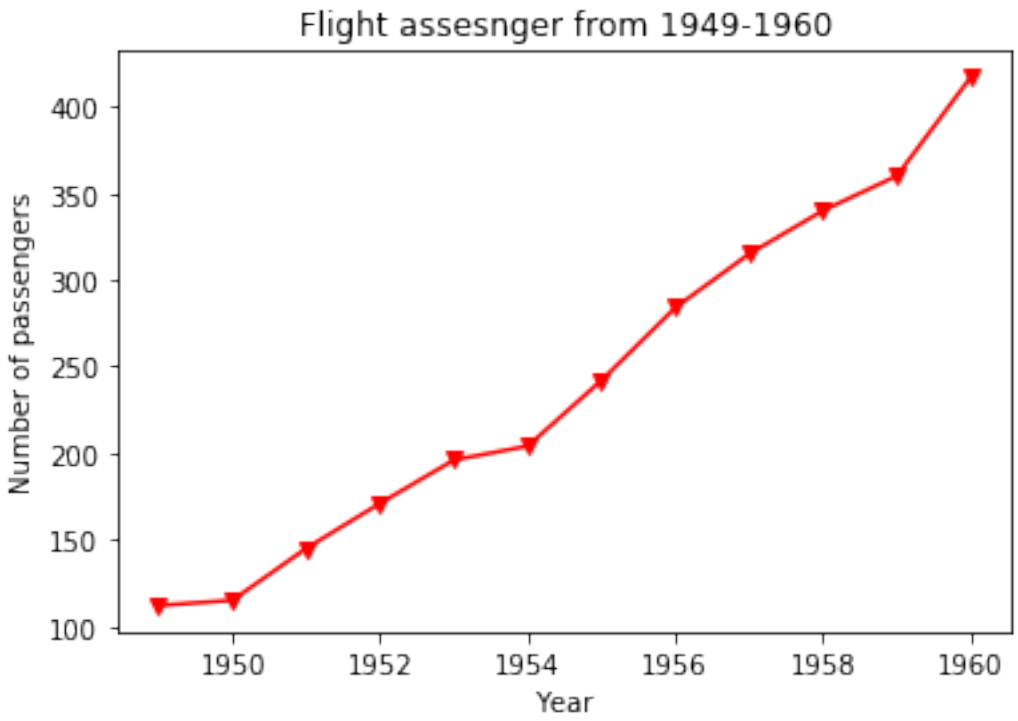
[8]:
```
Text(0, 0.5, 'Number of Passengers')
```



Finally, you can change the color of a line graph.

[9]:
```
# recreate the above plot
# pass "r" to color oprtion to show the plot in red
# don't forget to add labels/title/markers
plt.plot(flights_January["year"], flights_January["passengers"], marker = "v",␣
 ↪color = 'r')
plt.title("Flight assesnger from 1949-1960")
plt.xlabel("Year")
plt.ylabel("Number of passengers")
```

[9]:
```
Text(0, 0.5, 'Number of passengers')
```



Don't forget that Google is your best friend! Use it to explore all other options for color and marker style.

In this lesson you learned about: * using **matplotlib** package of Python to communicate your data using **line graph** plots. * aading labels and a title to a plt * adding markers to a line graph * changing the color of a line graph