Bar Chart Guide

For SMASH CS Fundamentals II: Data Analysis
This guide teaches you how you create basic bar charts with **matplotlib**.

matplotlib.pyplot API:
https://matplotlib.org/api/pyplot_api.html (https://matplotlib.org/api/pyplot_api.html)

Import matplotlib.pyplot as plt

As always, we will be using pyplot for all of our visualization needs.
Be sure to import it at the top of your file. Otherwise you won't be able to do anything.
The *"as plt"* part is just so we don't need to say *matplotlib.pyplot* each time we want to use it.

In [ ]:
```
import matplotlib.pyplot as plt
```

Create your data

Later in the course we'll be using very large amounts of data from a database.
But for now let's just use some small data.
Here's some: http://www.usclimatedata.com (http://www.usclimatedata.com)
Take a look at the monthly temperatures in Washington, DC from 1961-1990.
Let's use **tuples** to store this data.

In [2]:
```
# the data, taken from http://www.usclimatedata.com
low_avg_temps = (27, 28, 35, 44, 54, 63, 68, 66, 59, 48, 38, 29)
high_avg_temps = (42, 44, 53, 64, 75, 83, 87, 84, 78, 67, 55, 45)
month_labels = ("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug",
```

The *month_labels* tuple is there for later.
By the way, why did I decide to use tuples and not lists to store this data? Think about it:

- There are exactly 12 months.
- The order in which the numbers are arranged has meaning (i.e., the order of the months).
- Since these months already happened, so I the values won't change.

First, let's just make a bar chart of the **low average temperatures**.

Setup your graphic

I'll use comments to annotate each part of this bit.

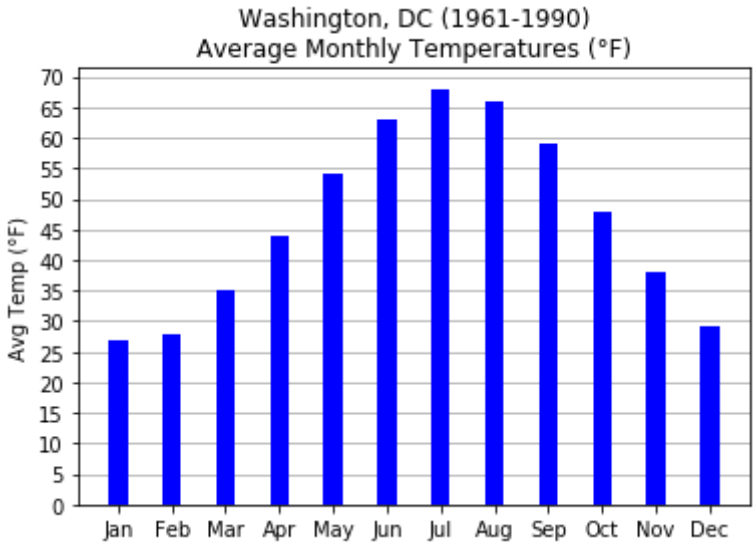
In [3]:
```
# Width of bars in pixels. I'll use this in a bit.
bar_width = 0.35

# This list corresponds with the number of months, which will be my x-ax
x_range = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]

# Create y_range by specifiying values from 0 to a max value that makes
# For degrees farenheit, a good max value might be 95.
# matplotlib will not actually draw a tick for 95 unless a temp gets tha
# still making sure I'm giving it a realistic maximum.

# We'll start at 0 and increment 5 each tick until we hit 95.
# Just like you would with a for-loop.

# y_range = range(0, 95, 5)

# You should transform that range into list by wrapping list() around it

# y_range = list(range(0, 95, 5))

# As a general rule, the baseline for a bar chart should be 0.
# Otherwise you risk mischaracterizing your data.
# Starting at 0 ensures that the length of each bar represents the tempe
# See here for more: https://flowingdata.com/2015/08/31/bar-chart-baseli

y_range = list(range(0, 95, 5))

#Place this above all of your figures.
#It just ensures gridlines appear behind everything else.
plt.rc("axes", axisbelow=True)

# Give this figure a unique number.
plt.figure(1)

# Create x-axis tickmarks according to the x_range.
# The "month_labels" here just apply the labels to those tickmarks.
plt.xticks(x_range,  month_labels)

# Create the y-axis tickmarks according to the y_range.
plt.yticks(y_range)

# Create horizontal gridlines so it's easier to measure the bars.
plt.grid(axis="y")

# What's the name of your graphic?
# I'm using \N{DEGREE SIGN} ensure the degree sign goes before F.
# And of course, \n just puts everything after it on the next line.
plt.title('Washington, DC (1961-1990)\nAverage Monthly Temperatures (\N{

# Don't need to label the x-axis since the months are already labeled.
# So let's label the y-axis.
plt.ylabel('Avg Temp (\N{DEGREE SIGN}F)')
```

Out[3]:
```
<matplotlib.text.Text at 0x7f7c2ebaf978>
```

Draw the bars

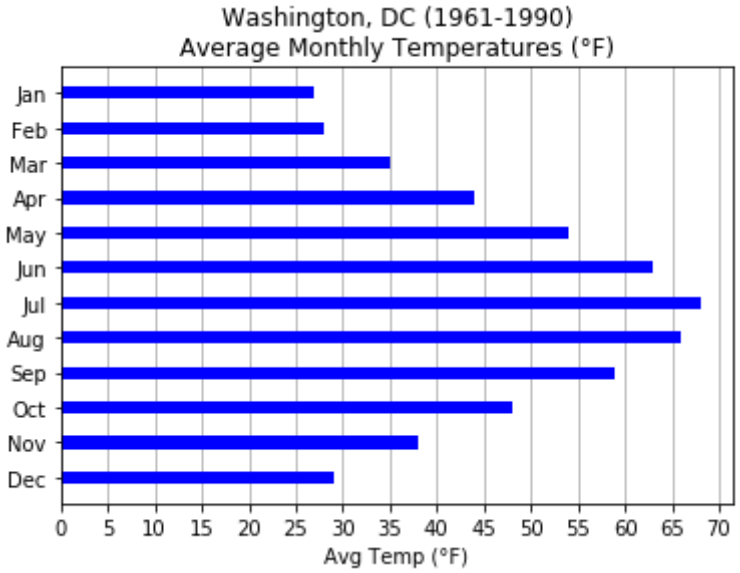
This is the easy part. It's just a few lines.

In [4]:
```
# Draw the bar chart.
plt.bar(x_range, low_avg_temps, width=bar_width, color="blue")

# Show the figure.
plt.show()
```



that looks cool. In PyCharm we can edit it to make it prettier from here.
But for now this'll do.
Why not make a **horizontal** bar chart?
It might provide a more intuitive way to view this data.

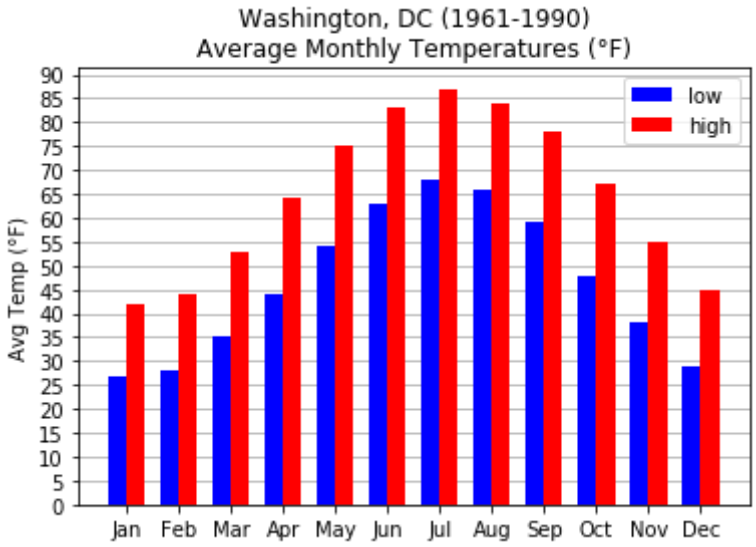
In [5]:
```
# Give the figure a unique number.
plt.figure(2)

# Setup your x and y tickmarks just like last time, but in reverse.
plt.xticks(y_range)
plt.yticks(x_range,  month_labels)

# This nifty little function here just reverses the y-axis, so January a
plt.gca().invert_yaxis()

# This time we'll need vertical gridlines.
plt.grid(axis="x")

# This part doesn't change.
plt.title('Washington, DC (1961-1990)\nAverage Monthly Temperatures (\N{

# Now we need to label our x-axis.
plt.xlabel('Avg Temp (\N{DEGREE SIGN}F)')

# To draw the bars, we use barh and specify height, not width.
plt.barh(x_range, low_avg_temps, height=bar_width, color="blue")

# Show the figure.
plt.show()
```



Draw two datasets side-by-side for comparison

This is both easier and harder than it sounds.
The easy part is drawing both datasets on the same chart.
The hard part is drawing them *neatly*.
You'll need some aesthetic sense and a bit of math for this.

In [6]:
```
# Give the figure a unique number.
plt.figure(3)

# This stuff doesn't change
plt.xticks(x_range,  month_labels)
plt.yticks(y_range)

# Horizontal gridlines again.
plt.grid(axis="y")

# This part doesn't change.
plt.title('Washington, DC (1961-1990)\nAverage Monthly Temperatures (\N{

# Label the y-axis
plt.ylabel('Avg Temp (\N{DEGREE SIGN}F)')

# Draw the the two sets of bars.

# Our bars now need to be setup so that they appear side-by-side.

# For the low_avg_temps bars, we'll use x - (bar_width / 2). This will m
# bars to the left of the tickmark by half of the width of the bar itsel

# Using this same logic, we'll use x + (bar_width / 2) for the high_avg_

# There are two ways to do either. Here's the longer way:

# low_range = []
# for x in x_range:
#     low_range.append(x - bar_width / 2)

# A slicker, more "Pythonistic" way of doing it:

# low_range = [x - bar_width / 2 for x in x_range]

# Choose whichever way you like. I'm using the shorter way to save some
low_range = [x - bar_width / 2 for x in x_range]
high_range = [x + bar_width / 2 for x in x_range]
plt.bar(low_range, low_avg_temps, bar_width, color='blue', label='low')
plt.bar(high_range, high_avg_temps, bar_width, color='red', label='high'

# Don't forget to display the legend! That's why I included the "labels"
# The function "draggable()" just makes it so you can drag around the le
plt.legend(loc="upper right").draggable()

# Show the figure.
plt.show()
```



That's all!

There is more to bar charts, but this is all you'll need for this course.
I suggest playing around with this more until you feel more comfortable.
And you can learn a lot by utilizing the excellent documentation I linked at the top.