data visualization

basic plot

Python

Dictionary

Table

Description automatically generated

Datasets in python

With [**loc**](http://pandas.pydata.org/pandas-docs/stable/indexing.html#different-choices-for-indexing) and **[iloc](http://pandas.pydata.org/pandas-docs/stable/indexing.html" \l "different-choices-for-indexing" \t "_blank)** you can do practically any data selection operation on DataFrames you can think of. [**loc**](http://pandas.pydata.org/pandas-docs/stable/indexing.html#different-choices-for-indexing) is label-based, which means that you have to specify rows and columns based on their row and column labels. **[iloc](http://pandas.pydata.org/pandas-docs/stable/indexing.html" \l "different-choices-for-indexing" \t "_blank)** is integer index based, so you have to specify rows and columns by their integer index like you did in the previous exercise.

**while loop**

In the previous chapter, you've discovered the if-elif-else construct, a way to alter the flow of your scripts. As soon as Python encounters an if statement, it checks the condition.

## if-elif-else

When this condition is True, the corresponding code is executed. If the conditions evaluates to False, and if there are elif and else statements, Python continues the search. Anyways, Python goes through this piece of code only once. After that, Python heads over to the next command in the script. The while loop is somewhat similar to an if statement: it executes the code inside if the condition is True. However, as opposed to the if statement, the while loop will continue to execute this code over and over again as long as the condition is true.

The iterrows method looks at the data frame, and on each iteration generates two pieces of data: the label of the row and then the actual data in the row as a Pandas Series. Let's change the rest of the for loop to reflect this change: we store the row label as lab, and the row data as row. To understand what's happening, let's print lab and row seperately. In the first iteration, lab is BR, and row is this entire Pandas Series. Because this row variable on each iteration is a Series, you can easily select additional information from it using the subsetting techniques you learned about earlier.

Data manipulation

homelessness\_reg\_fam = homelessness.sort\_values(["region", "family\_members"], ascending=[True, False])

# Print the top few rows

print(homelessness\_reg\_fam.head())

data manipulation with pandas