Pandas DataFrames

Optional Lesson 7

Slides By Samantha Klasfeld

(Please Sign in)

Previously on GCB Bootcamp

- Defining custom functions
- Hopefully you have already installed Pandas to your Desktop…

Today's Schedule

- Creating a Pandas DataFrame
- Selecting an Index or Column From a DataFrame
- Writing DataFrames

A DataFrame is a Virtual Table

	Column A	Column B	Column C
Row 1	value or NaN	value or NaN	value or NaN
Row 2	value or NaN	value or NaN	value or NaN
Row 3	value or NaN	value or NaN	value or NaN
Row 4	value or NaN	value or NaN	value or NaN

DataFrame Syntax

indexes
(AKA row names)

	gene_id	gene_name
0	AT5G61850	LFY
1	AT1G69120	AP1

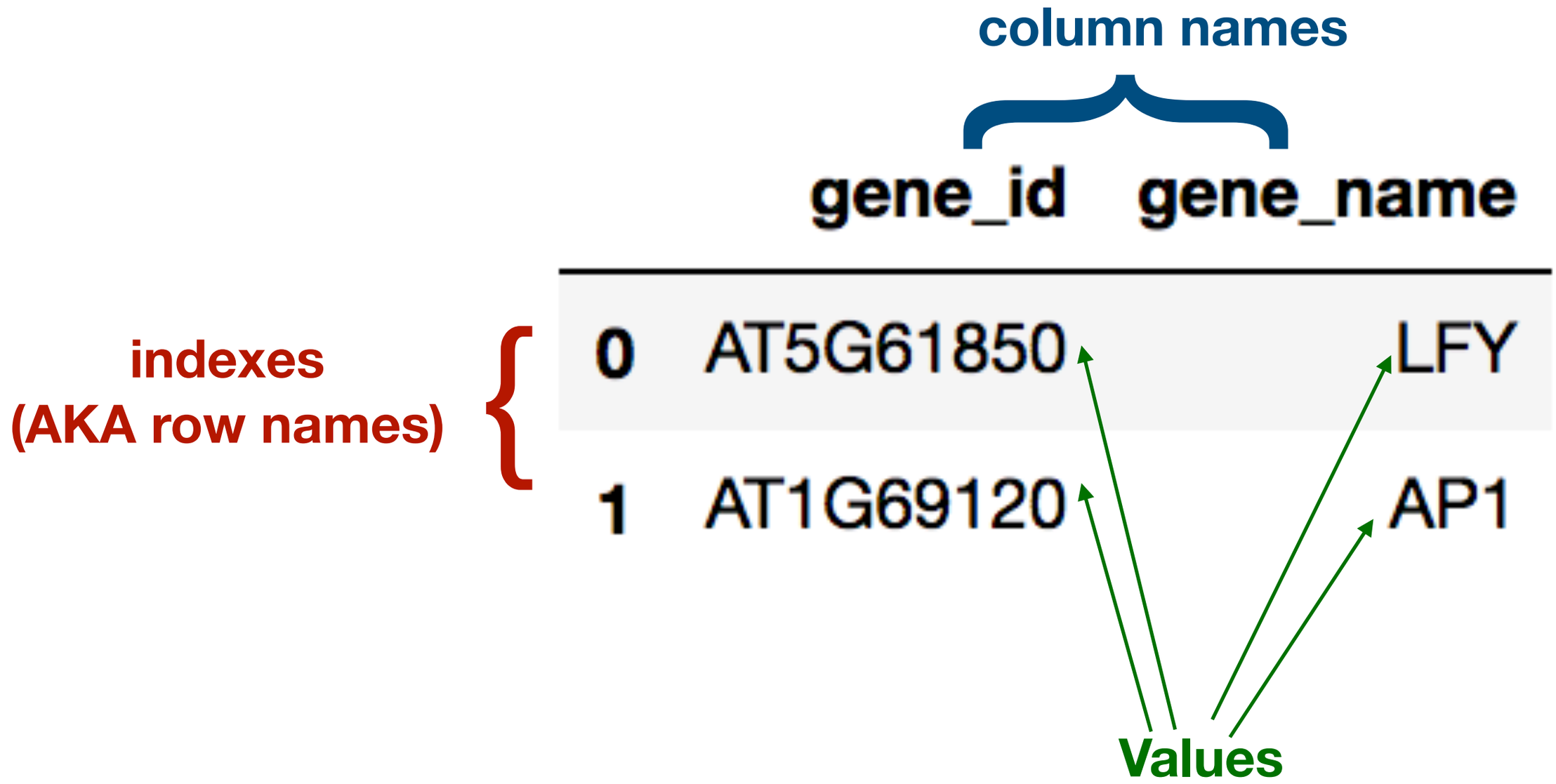
DataFrame Syntax

column names

	gene_id	gene_name
0	AT5G61850	LFY
1	AT1G69120	AP1

indexes (AKA row names)

DataFrame Syntax



There are several ways to create a pandas DataFrame

We can create a DataFrame from a List of Lists

```
1  import pandas as pd # import pandas module
2
3  # create list of lits
4  exp_listOflist = [
5      [2,4],
6      [4,2]
7  ]
8
9  # initialize the DataFrame
10 gene_frame = pd.DataFrame(exp_listOflist,
11                            columns = ["treatmentA","treatmentB"],
12                            index = ["Billy","Bob"]) # row names
13
14
15
16 gene_frame
```

	treatmentA	treatmentB
Billy	2	4
Bob	4	2

Or we can create a DataFrame from a List of Dictionaries

```
1  import pandas as pd # import pandas module
2
3  # create list of dictionaries
4  exp_listOfDict = [
5      {"treatmentA":2,"treatmentB":4},
6      {"treatmentA":4,"treatmentB":2}
7  ]
8
9  # initialize the DataFrame
10 gene_frame = pd.DataFrame(exp_listOfDict,
11                            index=["Billy","Bob"]) # row names
12
13
14
15 gene_frame
```

	treatmentA	treatmentB
Billy	2	4
Bob	4	2

Or we can create a DataFrame from a Dictionary of Lists

```python
import pandas as pd # import pandas module

# create dictionary of lists
exp_dictOfList = {
    "treatmentA":[2,4],
    "treatmentB":[4,2]
}

# initialize the DataFrame
gene_frame = pd.DataFrame(exp_dictOfList,
                          index=["Billy","Bob"]) # row names

gene_frame
```

	treatmentA	treatmentB
Billy	2	4
Bob	4	2

Or we can create a DataFrame from a Dictionary of Dictionaries

```
1  import pandas as pd # import pandas module
2
3  # create dictionary of dictionary
4  exp_dictOfDict = {
5      "treatmentA":{"Billy":2, "Bob":4},
6      "treatmentB":{"Billy":4, "Bob":2}
7  }
8
9  # initialize the DataFrame
10 gene_frame = pd.DataFrame(exp_dictOfDict)
11
12 gene_frame
```

	treatmentA	treatmentB
Billy	2	4
Bob	4	2

To custom orient a dataFrame using a dictionary use the .from_dict() function

```
1  import pandas as pd # import pandas module
2
3  # create dictionary of dictionary
4  exp_dictOfDict = {
5      "treatmentA":{"Billy":2, "Bob":4},
6      "treatmentB":{"Billy":4, "Bob":2}
7  }
8
9  gene_frame = pd.DataFrame.from_dict(exp_dictOfDict,
10                                      orient='index')
11
12 gene_frame
```

	Billy	Bob
treatmentA	2	4
treatmentB	4	2

Import pandas DataFrame from files

For example, lets say we have this bed file “example.bed” filled with genes.
Note that a bed file is tab-delimited. In this case the columns are ordered as such:

1. Chromosome Location
2. Start Location
3. Stop Location
4. Name of Feature
5. Score (in this case this column is uninformative)
6. Feature Strand

examble.bed

Chr1	1100	1600	GeneA	255	+
Chr2	15230	15342	GeneB	255	-
Chr2	15352	15455	GeneC	255	+
Chr2	17006	18110	GeneD	255	+
Chr3	10704	11671	GeneE	255	-

Step 1: Import Modules

```
1  # Step 1
2  import pandas as pd # import pandas module
3  import numpy as np # import numpy module
4                     # this module can classify
5                      # python objects that make
6                      # math easier later...
```

Step 1: Import Modules

```
1  # Step 1
2  import pandas as pd # import pandas module
3  import numpy as np # import numpy module
4                     # this module can classify
5                      # python objects that make
6                      # math easier later...
```

Note that pd is an alias for pandas and np is an alias for numpy now

Step 2: Make a list of the column names

```
# Step 1
import pandas as pd # import pandas module
import numpy as np # import numpy module
                   # this module can classify
                    # python objects that make
                    # math easier later...
```

```
# Step 2
bed_columns = ["chr", # chromosome location
               "start", # start location
               "stop", # stop location
               "name", # name of feature
               "score", # score of feature
               "strand"] # feature strand
```

Step 3: Create a dictionary containing the data-type of each columns

(Otherwise python automatically classifies them and its NOT always right!)

```
# Step 1
import pandas as pd # import pandas module
import numpy as np # import numpy module
                   # this module can classify
                    # python objects that make
                    # math easier later...
```

```
# Step 2
bed_columns = ["chr", # chromosome location
               "start", # start location
               "stop", # stop location
               "name", # name of feature
               "score", # score of feature
               "strand"] # feature strand
```

```
# Step 3
bedCol_types = {"chr" : str,
                "start" : np.int64,
                "stop" : np.int64,
                "name" : str,
                "score" : np.float64,
                "strand":str}
```

Step 4: Read the table into Pandas!

```
# Step 4
bed_df = pd.read_table("example.bed",
                       sep = "\t", # out file is tab-delimited
                       header = None, # the column names are
                                      # not in the file
                       names = bedCols, # sets column names
                       dtype = bedCol_types) # sets column types
bed_df
```

	chr	start	stop	name	score	strand
0	Chr1	1100	1600	GeneA	255.0	+
1	Chr2	15230	15342	GeneB	255.0	-
2	Chr2	15352	15455	GeneC	255.0	+
3	Chr2	17006	18110	GeneD	255.0	+
4	Chr3	10704	11671	GeneE	255.0	-

There are many more parameters one can set for the function `pandas.read_table`, but I do not have time to go through them all. If you google "pandas read_table" you find in-depth explanations at http://pandas.pydata.org/pandas-docs/stable/.

Basic DataFrame Functions

- bed_df**.head(n=*X*)** : gets the first ***X*** rows (default: n=5 if not stated)

- bed_df**.tail(n=*X*)** : gets the last ***X*** rows (default: n=5 if not stated)

- bed_df**.columns** : gets column names

- bed_df**.rows** : gets row names

- bed_df**.dtypes** : gets datatypes of each column

- bed_df**.shape** : gets height and width of the DataFrame in a list format

- **len(**bed_df**)** : gets the height of the DataFrame

Selecting an Index or Column From a DataFrame using .iloc & .loc

- .iloc - command to get values in DataFrame based on row and column locations

- .loc - command to get values in DataFrame based on row values and column values

Which rows will print after the following command (from which row indexes)?

```
1 import pandas as pd
2 peoples_df = pd.DataFrame(
3     {"name" : ["Bob", "Sue", "Tom", “Pam”],
4      "age" : [56, 21, 22, 35],
5      "inches" : [71, 63.5, 68.5, 62]})
6 peoples_df
```

	age	inches	name
0	56	71.0	Bob
1	21	63.5	Sue
2	22	68.5	Tom
3	35	62.0	Pam

```
1 peoples_df.iloc[range(1,3),:]
```

Which rows will print after the following command (from which row indexes)?

```
1 import pandas as pd
2 peoples_df = pd.DataFrame(
3     {"name" : ["Bob", "Sue", "Tom", “Pam”],
4      "age" : [56, 21, 22, 35],
5      "inches" : [71, 63.5, 68.5, 62]})
6 peoples_df
```

	age	inches	name
0	56	71.0	Bob
1	21	63.5	Sue
2	22	68.5	Tom
3	35	62.0	Pam

variable name for DataFrame

```
1 peoples_df.iloc[range(1,3),:]
```

Which rows will print after the following command (from which row indexes)?

```
1 import pandas as pd
2 peoples_df = pd.DataFrame(
3     {"name" : ["Bob", "Sue", "Tom", “Pam”],
4      "age" : [56, 21, 22, 35],
5      "inches" : [71, 63.5, 68.5, 62]})
6 peoples_df
```

	age	inches	name
0	56	71.0	Bob
1	21	63.5	Sue
2	22	68.5	Tom
3	35	62.0	Pam

variable name for DataFrame

```
1 peoples_df.iloc[range(1,3),:]
```

get values in DataFrame based on row and column locations

Which rows will print after the following command (from which row indexes)?

```
1 import pandas as pd
2 peoples_df = pd.DataFrame(
3     {"name" : ["Bob", "Sue", "Tom", “Pam”],
4      "age" : [56, 21, 22, 35],
5      "inches" : [71, 63.5, 68.5, 62]})
6 peoples_df
```

	age	inches	name
0	56	71.0	Bob
1	21	63.5	Sue
2	22	68.5	Tom
3	35	62.0	Pam

variable name for DataFrame

```
1 peoples_df.iloc[range(1,3),:]
```

get values in DataFrame based on row and column locations

This gives a range of numbers from 1-2 for row locations

Which rows will print after the following command (from which row indexes)?

```
1 import pandas as pd
2 peoples_df = pd.DataFrame(
3     {"name" : ["Bob", "Sue", "Tom", “Pam”],
4      "age" : [56, 21, 22, 35],
5      "inches" : [71, 63.5, 68.5, 62]})
6 peoples_df
```

	age	inches	name
0	56	71.0	Bob
1	21	63.5	Sue
2	22	68.5	Tom
3	35	62.0	Pam

variable name for DataFrame

```
1 peoples_df.iloc[range(1,3),:]
```

get values in DataFrame based on row and column locations

This gives a range of numbers from 1-2 for row locations

This symbol represents all of the columns

Which rows will print after the following command (from which row indexes)?

```
1 import pandas as pd
2 peoples_df = pd.DataFrame(
3     {"name" : ["Bob", "Sue", "Tom", “Pam”],
4      "age" : [56, 21, 22, 35],
5      "inches" : [71, 63.5, 68.5, 62]})
6 peoples_df
```

	age	inches	name
0	56	71.0	Bob
1	21	63.5	Sue
2	22	68.5	Tom
3	35	62.0	Pam

```
1 peoples_df.iloc[range(1,3),:]
```

	age	inches	name
1	21	63.5	Sue
2	22	68.5	Tom

ANSWER: This prints rows with indexes 1 & 2 since they are the second and third row

Which rows will print after the following command?

```
1  peoples_df
```

	age	inches	name
0	56	71.0	Bob
1	21	63.5	Sue
2	22	68.5	Tom
3	35	62.0	Pam

```
1 peoples_df.loc[range(1,3),:]
```

Which rows will print after the following command?

```
1 peoples_df
```

	age	inches	name
0	56	71.0	Bob
1	21	63.5	Sue
2	22	68.5	Tom
3	35	62.0	Pam

```
1 peoples_df.loc[range(1,3),:]
```

	age	inches	name
1	21	63.5	Sue
2	22	68.5	Tom

ANSWER: This also prints rows with indexes 1 & 2 since the value of these row indexes are 1 & 2 respectively since we do NOT have row names

Setting Row Names with **.set_index**

```
peoples_df
```

	age	inches	name
0	56	71.0	Bob
1	21	63.5	Sue
2	22	68.5	Tom
3	35	62.0	Pam

```
# set the row names to the "name" column
peoples_df = peoples_df.set_index("name")
peoples_df
```

	age	inches
name		
Bob	56	71.0
Sue	21	63.5
Tom	22	68.5
Pam	35	62.0

Which rows will print after the following command?

```
1  peoples_df
```

name	age	inches
Bob	56	71.0
Sue	21	63.5
Tom	22	68.5
Pam	35	62.0

```
1  peoples_df.loc[1,:]
```

Which rows will print after the following command?

```
1  peoples_df
```

name	age	inches
Bob	56	71.0
Sue	21	63.5
Tom	22	68.5
Pam	35	62.0

```
1  peoples_df.loc[1,:]
```

ANSWER: This prints an ERROR since there is no row name “1”

Which rows will print after the following command?

```
1  peoples_df
```

name	age	inches
Bob	56	71.0
Sue	21	63.5
Tom	22	68.5
Pam	35	62.0

```
1  peoples_df.loc[["Bob","Tom"],:]
```

Which rows will print after the following command?

```
1 peoples_df
```

name	age	inches
Bob	56	71.0
Sue	21	63.5
Tom	22	68.5
Pam	35	62.0

```
1 peoples_df.loc[["Bob","Tom"],:]
```

name	age	inches
Bob	56	71.0
Tom	22	68.5

ANSWER: This prints the rows with row names “Bob” and “Tom”

What value would this print?

```
1  peoples_df
```

name	age	inches
Bob	56	71.0
Sue	21	63.5
Tom	22	68.5
Pam	35	62.0

```
1  peoples_df.iloc[2,1]
```

What value would this print?

```
1  peoples_df
```

name	age	inches
Bob	56	71.0
Sue	21	63.5
Tom	22	68.5
Pam	35	62.0

```
1  peoples_df.iloc[2,1]
```

ANSWER: 68.5

What value would this print?

```
1  peoples_df
```

name	age	inches
Bob	56	71.0
Sue	21	63.5
Tom	22	68.5
Pam	35	62.0

```
1  peoples_df.loc["Sue","age"]
```

What value would this print?

```
1  peoples_df
```

name	age	inches
Bob	56	71.0
Sue	21	63.5
Tom	22	68.5
Pam	35	62.0

```
1  peoples_df.loc["Sue","age"]
```

ANSWER: 21

Which column will print after the following command?

```
1  peoples_df
```

name	age	inches
Bob	56	71.0
Sue	21	63.5
Tom	22	68.5
Pam	35	62.0

```
1  peoples_df.loc[:,"age"]
```

Which column will print after the following command?

```
1  peoples_df
```

name	age	inches
Bob	56	71.0
Sue	21	63.5
Tom	22	68.5
Pam	35	62.0

```
1  peoples_df.loc[:,"age"]
```

```
name
Bob     56
Sue     21
Tom     22
Pam     35
Name: age, dtype: int64
```

ANSWER: The “age” column

But why is the output formatted differently?

Pandas Series data object

- When we subset a DataFrame into a 1-dimensional array we DO NOT get *list* or *dictionary* objects.
- **Series** are 1-dimensional arrays with labels. They can originate from single rows or columns.

```
1 age_series= peoples_df.loc[:,"age"]
2 age_series
```

```
name
Bob     56
Sue     21
Tom     22
Pam     35
Name: age, dtype: int64
```

- We can check the values inside a series using the `.values` function

```
1 print 21 in age_series.values
```

```
True
```

```
1 print 100 in age_series.values
```

```
False
```

DataFrame Conditionals

- Occasionally you may want to subset a DataFrame based on specific columns. For example, we may want to subset the “peoples_df” to people under the age of 50.

- Using the .loc command we can use conditionals on rows (the first parameter) and/or columns (the second parameter)

```
1 peoples_df.loc[peoples_df.loc[:,"age"]<50,:]
```

name	age	inches
Sue	21	63.5
Tom	22	68.5
Pam	35	62.0

What value would this print?

```
1  peoples_df
```

name	age	inches
Bob	56	71.0
Sue	21	63.5
Tom	22	68.5
Pam	35	62.0

```
1  min(peoples_df.loc[peoples_df.loc[:,"age"]<50,"inches"])
```

What value would this print?

```
1  peoples_df
```

name	age	inches
Bob	56	71.0
Sue	21	63.5
Tom	22	68.5
Pam	35	62.0

```
1  min(peoples_df.loc[peoples_df.loc[:,"age"]<50,"inches"])
```

ANSWER: 62.0

What row(s) would this print?

```
1  peoples_df
```

name	age	inches
Bob	56	71.0
Sue	21	63.5
Tom	22	68.5
Pam	35	62.0

Note: for multiple conditions inside .loc or .iloc we only use a single ampersand

```
1  peoples_df.loc[(peoples_df.loc[:,"age"]<50) &
2                 (peoples_df.loc[:,"inches"]>65),:]
```

What row(s) would this print?

```
1  peoples_df
```

name	age	inches
Bob	56	71.0
Sue	21	63.5
Tom	22	68.5
Pam	35	62.0

```
1  peoples_df.loc[(peoples_df.loc[:,"age"]<50) &
2                 (peoples_df.loc[:,"inches"]>65),:]
```

ANSWER:

name		
Tom	22	68.5

Creating New Columns

You can also add columns using the .loc and .iloc commands.

For example, we can add a row for height in “centimeters” in addition to inches

```
1 peoples_df
```

name		
Bob	56	71.0
Sue	21	63.5
Tom	22	68.5
Pam	35	62.0

```
1 peoples_df.loc[:,"centimeters"] = peoples_df.loc[:,"inches"] * 2.54
2 peoples_df
```

name	age	inches	centrimeters
Bob	56	71.0	180.34
Sue	21	63.5	161.29
Tom	22	68.5	173.99
Pam	35	62.0	157.48

Exporting a DataFrame to a File

Similar to importing a text file to a *pandas* DataFrame, there are also several parameters for exporting a pandas DataFrame.

The syntax for exporting a DataFrame is :

```
dataFrame.to_csv(path_to_outputFile, arguments)
```

For example, to export a tab-delimited file from DataFrame `test_df` with a header and no row-names to the file “test.txt” we use the command:

```
test_df.to_csv(“test.txt”, sep=“\t”, header=True, index=False)
```

Summary

- We can build pandas DataFrames in python using dictionaries or files
- We can subset these DataFrames using .iloc or .loc
- We can export these DataFrames to a File

What else can we do with Pandas?

- Sort tables by specific columns
- Merge two DataFrames together using the .join() function
- Group series together using .groupby() function
- Create plots and visualizations using the matplotlib() library
- and more! Just google it! (eg. http://pandas.pydata.org/pandas-docs/stable/)

The End?

Next Lecture

- Merge two DataFrames together using the .join() function
- Group series together using .groupby() function