

Basics

%matplotlib inline	plots into notebook
df = pd.read_csv(path, index_col='name')	loads dataframe
df.head()	
df.tail()	
df.values	
df.plot()	
df.plot(style='.')	
df.index	returns row indexes
df[col].loc[index]	returns value with given column and index
df.loc[:, 'col_n+1'] = x	referring to a col that doesn't exist creates a new one

STOPPED FILLING AT LECTURE 6 LINE 122

Basic Dataframe Analysis

df.isnull()	returns bool
df.isnull().sum	returns sum of trues
df.isnull().any	checks whether there is a true
df[col].max()	
df[col].min()	
df[col].idxmax()	
df[col].idxmin()	
df[col].median()	
df[col].mean()	
df[col].describe()	gives statistic analysis
df[col].quantile(.5)	50% quantile
df.boxplot(by='col')	boxplot grouped by column
df.hist(bins=20)	histogram in 20 bars
df.plot.scatter(x='name', y='name')	scatterplot

Basic Dataframe Analysis (cont)

pd.plotting.scatter_matrix(df)	multiple scatterplots
pd.plotting.parallel_coordinates(df, 'name')	lines drawn connecting dimensions of an entry
df['col_name'].unique	returns list of singled entries
pd.get_dummies(df, columns=['Name'])	dummie column (0 or 1) that indicates whether the entry in another column is a certain entry
np.random.choice(n, x, replace=False)	selects random set
np.setdiff1d(set_1, set_2)	New set with only the differing entries
df.to_numpy()	gives array of entries

Working with a Dataframe

df['col1'] == x	bool if entry is x
df[df == x] = y	replace all values of a kind

Label-based indexing with .loc / .iloc

df.loc[rowindex, columnname]	
df.loc[3, col1]	3rd entry of 1st column
df.loc[3:6, [col1, col2]]	
df.loc[:, 'col1'] == 'name'	column with t/f whether entry in col1 is name
df.iloc[3:-1, 2:]	[rows, columns]

Label-based indexing with .loc / .iloc (cont)

df.iloc[:, [3, 1]]	columns with index 3 & 1
.loc is label based, .iloc is integer index based	

Series

s1 = pd.Series([1, 2, 3], index=['a', 'b', 'c'])	creates a pandas series
s1.add(s2, fill_value=0)	
s.isnull() ; s.notnull()	
s.dropna()	drops all rows with missing values
s.fillna(x)	
s = pd.DataFrame({'Size':s1, 'Weight':s2})	Best way to define dataframe out of series: Give dict out of columns
'e' in s1	returns bool
s.name = 'str'	names series
s.index.name = 'str'	names index If s doesn't exist, this creates a df
s.columns['Red', 'Green']	
s.columns.name = 'Color'	
s.reindex(['m', 'n', 'o'], method='ffill')	ffill = forward fill

