HACKWAGON

• ACADEMY •

DS102

Data Analysis with Python
The Cheat Sheet

All the imports - Data Manipulation

pandas	`import pandas as pd`
numpy	`import numpy as np`
matplotlib	`import matplotlib.pyplot as plt`
seaborn	`import seaborn as sns`
sklearn	`import sklearn`
statsmodels	`import statsmodels`
scipy	`import scipy`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

All the imports - Text Mining, Web Scraping

wordcloud	`import wordcloud`
requests	`import requests`
bs4	`from bs4 import BeautifulSoup`
nltk	`import nltk` `nltk.download('corpus')`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Instantiating a DataFrame

Instantiate a df from a CSV file	`df = pd.read_csv('x.csv')`
	
	Variations:
	`df = pd.read_csv('arrivals.csv', sep='|', index_col=0)`
	`df = pd.read_csv('arrivals.csv', index_col=0)`

Instantiate a df from a list of dict	`students = ['name': 'Ben', 'age': 22, 'gender': 'M'},`
	`{'name': 'John', 'age': 32, 'gender': 'M'},`
	`{'name': 'Charlotte', 'age': 21, 'gender': 'F'}]`
	
	`df = pd.DataFrame(students)`

Instantiate a df from a dict	`student = {'name': ['Ben','John'], 'age': [22,32], 'gender': ['M','M']}`
	
	`df = pd.DataFrame.from_dict(student)`
	
	name | age | gender
	Ben | 22 | M
	John | 32 | M

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Properties of a DataFrame

Gets the number of columns and rows in the DataFrame and returns it in a tuple	`df.shape`
Gets all the data types of each column	`df.dtypes`
Gets all the column labels of the DataFrame (header names)	`df.columns`
Gets the column count and the data type for each column	`df.info()`
Gets statistical data of the DataFrame such as count, mean, standard deviation, min value, max value	`df.describe() / s.describe()`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Peeking into a DataFrame

Obtains the first 5 (default) data in the DataFrame unless specified.
 	
Eg. df.head(2) obtains the first 2 data
	`df.head() / s.head()`
	
	`Variations: df.head(2), s.head(2)`

Obtains 1 (default) random data from the DataFrame without replacement unless specified
 	
Eg. df.sample(2, replace=True) obtains 2 random data with replacement
	`df.sample() / s.sample()`
	
	`Variations: df.sample(3), s.sample(3), df.sample(4, replace=True)`

Gets an array / list of unique values	`df.unique() / s.unique()`
Gets the number of unique values	`df.nunique() / s.nunique()`
Gets the count of non-NA values for each column	`df.count() / s.count()`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Peeking into a DataFrame

Get the minimum value for each column of the df	`df.min() / s.min()`
Get the maximum value for each column of the df	`df.max() / s.max()`
Get the summation of all the values for each column	`df.sum() / s.sum()`
Get the cumulative sum of all the values for each column	`df.cumsum() / s.cumsum()`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Sorting

Sorts the Dataframe by the column name specified	`df.sort_values('col_x')`
Sorts the Dataframe by the column name specified	`df.sort_values(['col_x'])`
Sorts the Dataframe by the column name specified in the ascending order	`df.sort_values('col_x', ascending=True)` Variations: `df.sort_values('col_x', ascending=False)`
Sorts the dataframe by column: ‘col_x’ followed by column: ‘col_y	`df.sort_values(['col_x', 'col_y'])` Variations: `df.sort_values(['col_x', 'col_y', ascending=[True, False])`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Retrieval from a DataFrame

Locates a selection based on the position index	`df.iloc[0]` Variations: `df.iloc[[1, 2, 3]]`, `df.iloc[1:10]`
Access a group of row and column	`df.loc[0]` Variations: `df.loc[[1, 2, 3]]`, `df.loc[range(0, 10)]`
Retrieves all values for the specified column in the dataframe	`df['col_x']`
Retrieves all values for the specified columns in the dataframe	`df[['col_x', 'col_y']]`
The value for the specified column in the series	`s['col_x']`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Simple Statistics & Moments of a Distribution

Returns the mean of each numeric column in the dataframe / series	`df.mean() / s.mean()`
Returns the standard deviation of each numeric column in the dataframe / series	`df.std() / s.std()` `Variations: df.sample(t), s.sample(t), df.sample(t, replace=True)`
Returns the variance of each numeric column in the dataframe / series	`df.var() / s.var()`
Returns the median of each numeric column in the dataframe / series	`df.median() / s.median()`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Simple Statistics & Moments of a Distribution

Returns the mode of each numeric column in the dataframe / series	`df.mode() / s.mode()`
Returns a unbiased skew of the dataframe / series	`df.skew() / s.skew()`
Returns a unbiased kurtosis of the dataframe / series	`df.kurt() / s.kurt()`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Filtering

Filter for values greater than *i*	`df[df['col_1'] > i]`
	
	`Variations: df[df['col_1'] >= i],`
	`df[df['col_1'] < i],`
	`df[df['col_1'] <= i]`

Filter for values equal to *i*
(Applies to strings as well)
	`df[df['col_1'] == i]`
	
	`Variations: df[df['col_1'] == 'string_i']`

Filter for having a list of values
(Applies to strings as well)
	`df[df['col_1'].isin([i, j, k]]`
	
	`Variations: df[df['col_1'].isin(['string_i', 'string_j',`
	`'string_k'])]`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Filtering (AND, OR, NOT)

Using NOT on a condition	`df[~(df['col_1'] > i)]`

Using AND on two conditions (record must fulfill condition_1 AND condition_2)	`df[(condition_1) & (condition_2)]`
	`df[(df['col_1'] > lo) & (df['col_1'] < hi)]`
	
	Variations: `df[(condition_1) & (condition_2) & ...]` **(nsc)**

Using OR on two conditions (record can fulfill condition_1 OR condition_2)	`df[(condition_1) | (condition_2)]`
	`df[(df['col_1'] < lo) | (df['col_1'] > hi)]`
	
	Variations: `df[(condition_1) | (condition_2) | ...]` **(nsc)**

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Handling Empty Values

Replaces all NA / NaN values with the specified value	`df.fillna()`
Drops rows with any NA / NaN values	`df.dropna()` `Variations: df[(condition_1) & (condition_2) & ...]` **(nsc)**
Returns a same sized object that shows if the value is a NA / NaN. If value is null, it will be shown as True else, False	`df.isnull()` `Variations: df[(condition_1) | (condition_2) | ...]` **(nsc)**

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Substitutions

Renames the column's name or index name	`df.rename()` `Variations: df.rename(columns={}), df.rename({}, axis='index')`
Replaces the specified values with another value and returns the Dataframe.	`df.replace()` `Variations: df[(condition_1) & (condition_2) & ...]` **(nsc)**

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Removing Items

Drops the row by index	`df.drop(axis=0)` Variations: `df.drop('column name', axis=0)`
Drops the column of the specified value	`df.drop(axis=1)` Variations: `df.drop('column name', axis=1, inplace=True)`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Creating New Columns

Using simple arithmetic	`df['x'] = df['a'] + df['b'] + df['c'] + ...`

Using apply and a function (1 column)	`def minus_5(x):`
	`  return x-5`
	
	`df['col_name'].apply(minus_5)`

Using apply and a function (2 columns)	`cols_name = ['col1','col2']`
	`for col in cols_name:`
	`  df['col'].apply(minus_5)`

Using apply and a function (named params)	`def add(x, y=4):`
	`  return x+y`
	
	`df['col_name'].apply(add, y=5)`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Creating New Columns

Using apply and a function (lambda)	`D = {True: 'Yes', False: 'No'}`
	`df['yn'] = df['Legendary'].apply(lambda x :`
	`return D[x])`

Using apply and a function (numpy functions)	`df['yn'] = df['Attack'].apply(np.int)`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Aggregation

groupby() on one column	`df.groupby('column1')`
groupby() on two or more columns	`df.groupby(['column1','column2'])`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Aggregation Functions

Gives the maximum value of the first numerical valued column in the group	`groupby().max()`
Gives the minimum value of the first numerical valued column in the group	`groupby().min()`
Shows the number of occurrence that falls in the group	`groupby().size()`
Aggregates using one or more functions over the groups	`groupby().agg()` `Variations: df.groupby().agg(['min','max'])`
Computes the mean of the groups	`groupby().mean()`
Transforms means combining based on the idmax	`groupby().transform(idmax)`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Pivot

df

	date	game	no_players	score
0	14/12/2018	basketball	14	87
1	15/12/2018	basketball	13	76
2	16/12/2018	basketball	15	54
3	14/12/2018	soccer	14	2
4	15/12/2018	soccer	15	4
5	16/12/2018	soccer	14	1
6	14/12/2018	tennis	4	24
7	15/12/2018	tennis	4	23
8	16/12/2018	tennis	2	17

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Pivot

```
df.pivot(index = 'game',
columns = 'date')
```

	no_players			score		
date	14/12/2018	15/12/2018	16/12/2018	14/12/2018	15/12/2018	16/12/2018
game						
basketball	14	13	15	87	76	54
soccer	14	15	14	2	4	1
tennis	4	4	2	24	23	17

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Pivot

```
df.pivot(index='game',columns='date',values='no_players')
```

date	14/12/2018	15/12/2018	16/12/2018
game			
basketball	14	13	15
soccer	14	15	14
tennis	4	4	2

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Pivot Tables

Returns a pivot table with the resultant index and columns which can aggregate the data based on what you specify in the index.
 	
Values are used to specify which data we want to grab from the pivot table which return them in a column. If there are multiple indexes with the same key, it will take the average. However, if you were to specify aggfunc to be np.sum, it will return the sum of those with the same index.
 	
Aggfunc can take a list of functions to apply to the index. E.g. len, np.mean()
	`pd.pivot_table(df,`
	`index=[],`
	`columns=[],`
	`values=[],`
	`aggfunc=len)`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Pivot Tables

```
df.pivot_table(index='game',columns='no_players')
```

	score				
no_players	**2**	**4**	**13**	**14**	**15**
game					
basketball	NaN	NaN	76.0	87.0	54.0
soccer	NaN	NaN	NaN	1.5	4.0
tennis	17.0	23.5	NaN	NaN	NaN

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Pivot Table (Margins)

```
import numpy as np

df.pivot_table(index='game',columns='date',
margins=True,aggfunc=np.sum)
```

	no_players				score			
date	14/12/2018	15/12/2018	16/12/2018	All	14/12/2018	15/12/2018	16/12/2018	All
game								
basketball	14	13	15	42	87	76	54	217
soccer	14	15	14	43	2	4	1	7
tennis	4	4	2	10	24	23	17	64
All	32	32	31	95	113	103	72	288

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Merge & Join

Returns a merged Dataframe of a and b with the name of ‘c’ as the header	`pd.merge(a, b, on = 'c')`
Returns a joined dataframe with the left dataframe as a and the right as b. The rows which are not filled will be replaced by ‘NaN’	`a.join(lsuffix = 'a', rsuffix = 'b')`
Returns a transposed a with the rows as columns and vice versa	`a.transpose() a.T()`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

More EDA (Quartiles, IQR, Correlation)

Returns the 25th percentile of s	`np.percentile(s, 25)`
Returns the 25th, 50th, and 75th percentile of s	`np.percentile(s, [25, 50, 75])`
Returns the correlation between s1 and s2	`s1.corr(s2)`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Binning

Sort and separate values into bins. Bins are ranges such as ‘50,100’ or ‘0,50’, evenly spreaded out. A bin represents a range of that variable. if an observation falls within that range then it belongs to the bin	`s.cut(bins=4, right= False)`
Sort and separate values into bins but qcut will choose bins such that there will be the same number of records in each bin. q is used to specify the number of bins. Qcut is frequency based.	`s.qcut(q = 5)`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Dropping

Drops all duplicate, only keeping the last occurrence	`df.drop_duplicates(keep = 'last')`
Drops the column ‘A’	`df.drop(columns = ['A'])`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Plotting I

Creates a line chart of the values in the dataframe	`df.plot(kind='line', x='', y='')`
Creates a bar chart of the values in the dataframe	`df.plot(kind='bar', x='', y='')`
Creates a scatter plot of the values in the dataframe	`df.plot(kind='scatter', x='', y='')`
Creates a boxplot of the values in the series	`s.plot(kind='box', x='', y='')`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Plotting II

Creates a distribution plot. Mainly used to look at univariate distribution.	`sns.distplot(kind='line', x='', y='')`
Creates a kdeplot. Mainly used to estimate the probability density function of a variable.	`sns.kdeplot(kind='bar', x='', y='')`
Creates a plot of two variables with bivariate and univariate graphs	`sns.jointplot(kind='scatter', x='', y='')`
Creates a violin plot (combination of boxplot and kernel density estimate). Shows the distribution of quantitative data	`sns.violinplot(kind='box', x='', y='')`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Plotting III

Creates a heatmap using the Seaborn data visualization library	`sns.heatmap()`
Creates a swarmplot using the Seaborn data visualization library. Swarmplot is a form Categorical scatterplot	`sns.swarmplot()`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

`fig, ax` and Multiple Subplots

Creates a figure and one subplot	`fig, ax = plt.subplots()`

Creates a new figure with another figure number
 	
Adds an Axes to the figure.
add_subplot(number of rows, number of columns, index position on the grid)
	`fig = plt.figure()`
	`Variation: fig = plt.figure(figsize=(16, 6))`
	
	`ax1 = fig.add_subplot(1,2,1)`
	
	`Variation: ax1 = fig.add_subplot(1,2,1)`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

ax Settings

Sets the title of the graph with a given string input	`ax.set_title(t)`
Sets the label text of the x and y axis	`ax.set_xlabel(y)` `ax.set_ylabel(y)`
Sets the x-tick labels with a list of labels	`ax.set_xticks(np.arange(3), ['label1', 'label2','label3'])`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Extracting from a website

Sends a request to the server to get the contents of it	`resp = requests.get(<Website URL>)`
Converts the HTML format data to a BeautifulSoup data type for extraction	`soup = BeautifulSoup(resp.text, 'html.parser')`
Allows us to see the HTML data in a readable format with indentations.	`soup.prettify()`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

Extracting from a website

Finds a specified tag in the soup data
Returns a result and if the tag is not found, returns a None
 	
Attrs → if you want to find a specified tag to find
 	
Eg: div tag with class attribute 'about-widget'
	`Sentence = soup.find()`
	
	`Variation: soup.find(<tagname>, attrs={'class' : '<classname>'})`

Similar to .find(), instead this will return a list containing the results and if the tag is not found, it returns an empty list	`soup.find_all()`
Gets the text of sentence	`sentence.text`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct

WordCloud

Generates a word cloud based on the string that you input	`WordCloud(width=800, height=400, background_color='black', max_words=100).generate(<variable name>)`
Shows the generated WordCloud	`plt.imshow()`

HACKWAGON
•ACADEMY•

Key
df: pandas DataFrame
s : pandas Series
nsc: NOT syntactically correct