Import numpy as np

Import pandas as pd

Import matplotlib.pyplot as plt

from pandas import Series

from pandas import

**DAY 1**

* Graphs – Scatter plot

# df is a dataframe

df.plot(kind='scatter',x='num\_bikes',y='num\_pets',color='blue')

plt.show()

* Create dataframe

dictionary\_1 = {

'name':['john','vj','xin','amanda','sungjin','lisa','jose'],

'age':[23,78,22,19,45,33,20],

}

df = pd.DataFrame(dictionary\_1)

**DAY 2**

* Load CSV file

menu = pd.read\_csv(‘menu.csv’)

avocado = pd.read\_csv(‘data/avocado.csv’, delimiter=’;’, decimal=’,’)

* List

Names = [“Charlotte”, “Ingrid”]

Scores = [80, 95, 2]

* Array

Ascores = np.array(scores)

* Series--one dimensional array w axis labels
* Makes a dataframe – sData.to\_frame()
* Find number of rows and cols -- .shape
* Extract list of columns – menu.columns
* Extract specific column –

menu[‘Category’],

Iloc indexes positions ex. Integers

Loc indexes labels

Error: Menu\_i.loc[0]

No error: menu\_i.iloc[0]

menu.category,

menu[[‘Item’, ‘Calories’]]

* Extract row by index -- menu.iloc[0]
* Set index by item not by number

menu\_i = menu.set\_index(‘Item’)

menu\_i.loc[‘Egg McMuffin’]

* Extract row and a slice of its columns -- menu\_i.iloc[0,0:2]
* Extract column and a slice of its rows -- menu\_i.iloc[1:3, :]
* Sorting -- Menu\_sorted\_cal = m
* enu.sort\_values{‘Calories’, ascending=True)
* Filtering

// displays all index and True/False

Menu[‘Trans Fat’] > 0

// displays entire row including item, cal, fat, etc

Menu[menu[‘Trans Fat’] > 0.0]

1. Write some python code to divide all the scores by 2. The results should be saved to a variable called half.
2. Display the four menu items that have the most Saturated Fat (the absolute amount, not the % Daily Value).[¶](http://localhost:8890/lab#-Exercise-4-(2-points):-Display-the-four-menu-items-that-have-the-most-Saturated-Fat-(the-absolute-amount,-not-the-%-Daily-Value).)

menu\_sorted\_by\_fat = menu.sort\_values('Saturated Fat',ascending=False)

1. List the top 3 breakfast items have the most Dietary Fiber.[¶](http://localhost:8890/lab#Exercise-5-(2-points):-List-the-top-3-breakfast-items-have-the-most-Dietary-Fiber.)

menu\_sorted\_by\_fiber = menu.sort\_values('Dietary Fiber',ascending=False)

**DAY 3**

* Find values from specific column

menu.Carbohydrates.head()

* Sort dataframe

menu[(menu['Category']) != 'Beverages'].sort\_values(['Carbohydrates'], ascending = True)

* Count values – value\_counts()

avocado['type'].value\_counts()

* Count number of types in category – number of different regions

len(avocado['region'].unique())

* Read JSON files

pd.read\_json('data/nfl\_football\_profiles.json')

* Replace

nfl\_football\_players['current\_salary'].str.replace(',', '')

* Extract tables from web pages – pd.read\_html

contracts\_scraped = pd.read\_html('https://en.wikipedia.org/wiki/List\_of\_largest\_sports\_contracts',header=0)

* Dropna & fillna
* Vectorized string functions
* CSV file into pandas dataframe

1. Read the data/avocado\_eu.csv file using the correct delimiter and decimal character into a dataframe

pd.read\_csv('data/avocado\_eu.csv', delimiter=';', decimal=',')

1. Count the number of players from each sport in the List of Largest Sports Contracts.

contracts['Sport'].value\_counts()

1. Create a new dataframe that contains all the columns in the nfl\_football\_players dataframe as well as an additional column that contains each player's height in centimeters.

nfl\_football\_players['height']

height = nfl\_football\_players.height.str.split('-', expand = True)

nfl\_football\_players['centimeters'] = (height[0].astype(float)\*12+height[1].astype(float))\*2.54

**Day 4**

* Read HTML (pd.read\_html(link\_url)
* dropna
* Make, sort and construct dataframe

Names = [‘john’, ‘jack’]

url = [‘www.’ , ‘www.’]

udf = pd.DataFrame()

udf[‘name’] = names

udf[‘url’] = urls

* Set index

// get index

df\_nameindexed.index

// set index as name (and race)

df\_nameindex = df.set\_index(‘name’)

df\_racename\_indexed = df.set\_index([‘race’, ‘name’])

// returns info about aragon

df\_nameindex.loc[‘Aragon’]

// returns all hobbits

df\_racename\_indexed = df.set\_index[‘Hobbit’]

// returns all hobbit’s name and magic strength

df\_racename\_indexed[‘magic’].loc[‘Hobbit’]

df\_racename\_indexed[‘magic’].loc[‘Hobbit’, ‘Frodo’]

* .apply() – loops a series/list etc through a function

udf['url'].apply(get\_aliases)

* Concat – joins dataframes together

pd.concat([dataframe1,dataframe2],axis="columns")

* Merge – join databases and can specify index and how (left/right)

df.merge(urls\_wrong\_order,on='name')

pandas.merge(left,right,left\_on,right\_on)

* Stacking – takes wide data and makes it taller

Rather than horizontal, stack vertically

1. Construct a dataframe with 5 columns (names, races, magic, aggression, and stealth) using the lists above.
2. Using .loc find how much aggression Legalos, an Elf, has

agg = df\_racename\_index[‘aggression’].loc[‘Elf, ‘Legolas’]

1. Construct a dataframe with lotr\_wikipedia\_wrong\_order.csv which is in the data folder.

pd.read\_csv('data/lotr\_wikipedia\_wrong\_order.csv')

1. Join the ```udf``` DataFrame (that contains aliases) to the ```df``` DataFrame using an appropriate merge

**Day 5**

* stringIO to create dataframe – paste data straight into jupyter
* .describe() – Basic statistics
* .groupby() (for numbers)

fruit.groupby('State').describe()

fruit.groupby(['State','Fruit']).sum()

fruit.groupby('Retailer').sum().sort\_values(by='Sales').tail(5)

* Pivot

metal.pivot(index='Item',columns='CType',values='USD')

* Pivot tables (if overlap)

p=metal.pivot\_table(index='Item',columns='CType',values='USD',aggfunc=np.mean)

1. Using pivot and the dataframe below make a table with each person as the index, their purchased items as columns, and the price they paid as the values
2. Which state has the maximum total sales?
3. Which state has the maximum total sales for apples?

fruitSalesByState = fruit.groupby('State').sum()

max\_state = fruitSalesByState['Sales'].idxmax()

1. Which race has the lowest average magic value

lotr.groupby('race')['magic'].mean().idxmin()

**DAY 6**

* Algorithm efficiency
* 5 types of big O notation
* Execution times

%timeit

%lprun -f name\_of\_function name\_of\_function(x)

* Profiling output
* Haversine – determines distance between 2 points on a sphere given lat and long

**Day 10**

* Spacy
* Lower case – .lower()
* Removes all punctuation – re.sub(r'[^\w\s]|\_+', ' ', some\_text\_lower)
* Remove line break – some\_clean\_text = some\_text\_lower\_no\_punct.replace(os.linesep, " ")
* Remove numbers – re.sub(r'[0-9]', ' ', some\_clean\_text)
* Remove stop words –

for word in sentence:

if word.text not in STOP\_WORDS:

print(word.text)

words\_nostop = list()

for word in words:

if word not in STOP\_WORDS:

words\_nostop.append(word)

* Most common words – Counter(words).most\_common(10)
* Graph of common words
* Tokenization (split sentence into numbers/tokens)

For i, token in enumerate(doc)

Print(i,token)

1. Hello
2. World
3. !

* Split sentence by punctuation

// index number on each split sentence

sentences = re.split(r"(?<!\w\.\w.)(?<![A-Z][a-z]\.)(?<=\.|\?)\s", doc\_multsent)

for i, sent in enumerate(sentences):

print(i, sent)

**Day 13**

* Matplotlib