Python and Pandas

What is Pandas

- An open-source Python package that is most widely used for data science/data analysis and machine learning tasks.
- Built on top of NumPy which provides support for multi-dimensional arrays.
- References both “Panel Data” and “Python Data Analysis”
- Created by Wes McKinney in 2008

Pandas vs NumPy

- NumPy provides objects for multi-dimensional arrays
- Pandas is capable of offering an in-memory 2d table object called a **DataFrame**.
- NumPy consumes less memory compared to Pandas

What can Pandas do?

- Data Cleansing
- Data normalization
- Data visualization
- Data inspection
- Data fill
- Merges and joins
- Statistical analysis
- Loading and saving data

Remember:

Pandas is a Module

You have to install it first:

pip install pandas

You have to import it at the beginning of every code file:

import pandas as pd

DataFrames & Series

DataFrames

- 2-dimensional data structure
- 2-dimensional array
- Table with rows & columns

Series

- Column within a table
- 1 dimensional array holding data of any types.

DataFrame Example

(create a new Python file)

```
import pandas as pd
Report = {"Classes": ["Math", "Science",
"Spanish", "History", "Health"],
    "Grades": [75, 80, 95, 60, 100]}

results = pd.DataFrame(Report)
print(results)
```

You should have

	Classes	Grades
0	Math	75
1	Science	80
2	Spanish	95
3	History	60
4	Health	100

Finding the location

```
import pandas as pd
Report = {"Classes": ["Math", "Science",
"Spanish", "History", "Health"],
    "Grades": [75, 80, 95, 60, 100]}

results = pd.DataFrame(Report)
print(results.loc[3])
```

You should have

```
Classes    History
Grades          60
Name: 3, dtype: object
```

Location of More than 1 line

```
import pandas as pd
Report = {"Classes": ["Math", "Science",
"Spanish", "History", "Health"],
    "Grades": [75, 80, 95, 60, 100]}

results = pd.DataFrame(Report)
print(results.loc[[2, 3]])
```

You should have

	Classes	Grades
2	Spanish	95
3	History	60

Naming the Rows

```
import pandas as pd
Report = {"Classes": ["Math", "Science",
"Spanish", "History", "Health"],
    "Grades": [75, 80, 95, 60, 100]}

results = pd.DataFrame(Report, index =
["week1", "week2", "week3", "week4", "week5"])
print(results)
```

You should have

	Classes	Grades
week1	Math	75
week2	Science	80
week3	Spanish	95
week4	History	60
week5	Health	100

Locating a specific row

```
import pandas as pd
Report = {"Classes": ["Math", "Science",
"Spanish", "History", "Health"],
    "Grades": [75, 80, 95, 60, 100]}

results = pd.DataFrame(Report, index =
["week1", "week2", "week3", "week4", "week5"])
print(results.loc[“week3”])
```

You should have

```
Classes    Spanish
Grades          95
Name: week3, dtype: object
```

Series Examples

(create a new Python file)

```
import pandas as pd

age = [20, 40, 60]
years = pd.Series(age)

print(years)
```

Finding the location

```
import pandas as pd

age = [20, 40, 60]
years = pd.Series(age)

print(age[0])
```

You should have

```
20
```

Naming the Rows

```
import pandas as pd

age = [20, 40, 60]
years = pd.Series(age, index = ["Me", "My
Brother", "My Sister"])

print(years)
```

You should have

```
Me              20
My Brother      40
My Sister       60
   dtype: int64
```

Locating a specific row

```
import pandas as pd

age = [20, 40, 60]
years = pd.Series(age, index = ["Me", "My Brother", "My Sister"])

print(years["My Sister"])
```

You should have

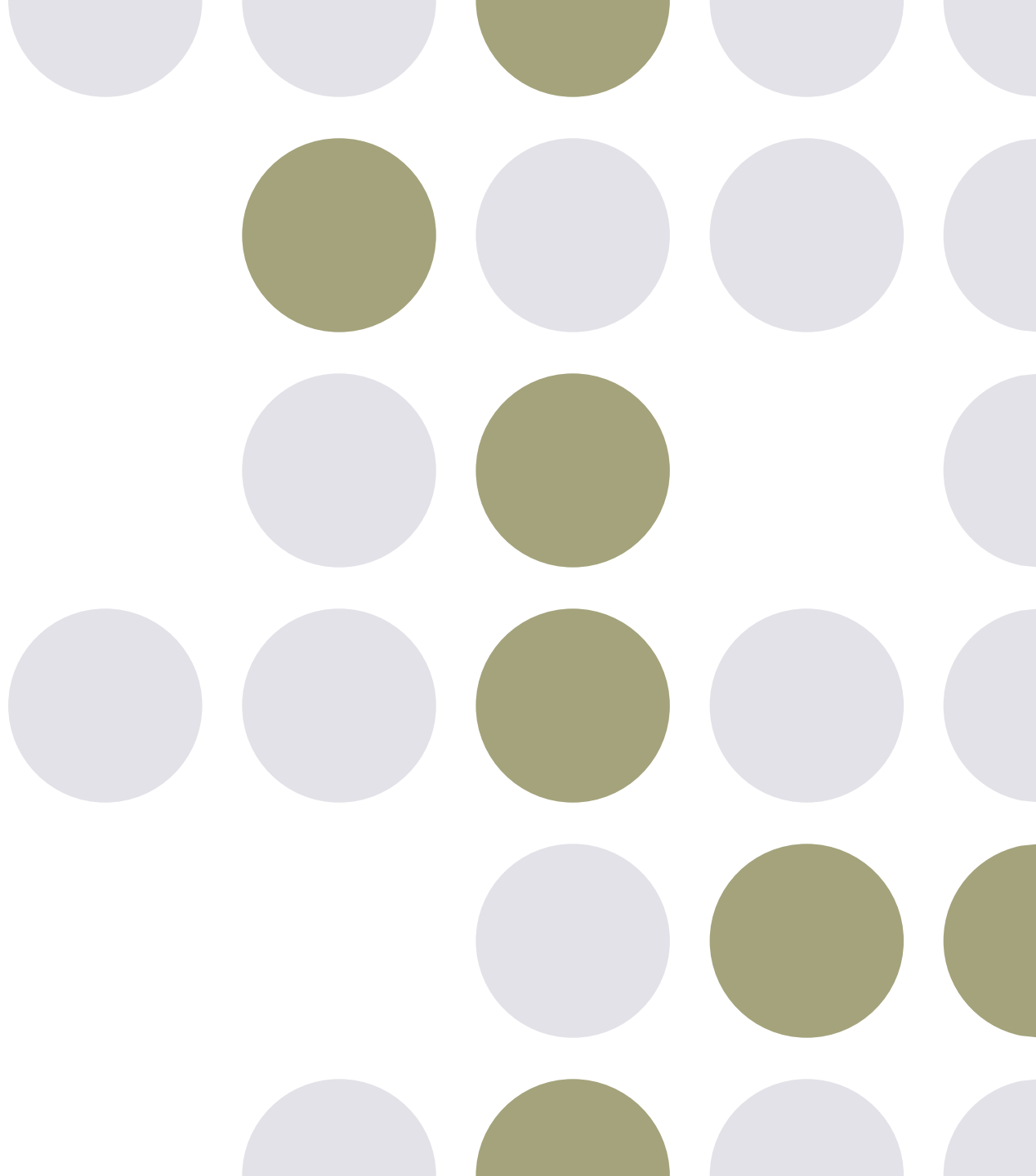
```
60
```

Now,
let’s try working in Jupyter Notebook

Sorting dataframes

- df.sort_values(by=[‘x’], inplace=True)
- df.sort_values(by=[‘x’], inplace=True, ascending=False)
- df.sort_values(by=[‘first column’, ‘second column’,...], inplace=True)

Reading .csv files with Pandas



What is a .CSV file???

- A file that contains plain text, with values that are commas separated
- CSV == “Commas Separated Values”
- Common file extension for data sets
- You can open it in Notepad but the format will be off; **Use VS Code instead**

Pandas -- reading data from files

(create a new Python file)

```
import pandas as pd


df = pd.read_csv('data.csv')
print(df.to_string())
```

With this print statement, you get the WHOLE dataframe

Print Summary of a DataFrame

```
import pandas as pd

df = pd.read_csv('data.csv')
print(df)
```

With this print statement,
you get the first 5 lines & the last 5 lines
With the row & column count

max_rows command

```
import pandas as pd

df = pd.read_csv('data.csv')

print(pd.options.display.max_rows)
```

You will get the max rows set on your system

Setting max_rows

```
import pandas as pd
pd.options.display.max_rows = 9999
df = pd.read_csv('data.csv')
print(df)
```

With this setting,
you should get the whole dataframe

Viewing the FIRST 10 rows

```
import pandas as pd

df = pd.read_csv('data.csv')
print(df.head(10))
```

*If you have an empty **head** command, you get the first 5*

Viewing the LAST 10 rows

```
import pandas as pd

df = pd.read_csv('data.csv')
print(df.tail(10))
```

*If you have an empty **tail** command, you get the last 5*

Information about the DataFrame

```
import pandas as pd

df = pd.read_csv('data.csv')
print(df.info())
```

*If you print an empty **info** command,*
you get the DataFrame Summary

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 169 entries, 0 to 168
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Duration  169 non-null    int64
 1   Pulse     169 non-null    int64
 2   Maxpulse  169 non-null    int64
 3   Calories  164 non-null    float64
dtypes: float64(1), int64(3)
memory usage: 5.4 KB
```

Explanation of type of object

How many rows and columns

Name of each column with data type

Total number of data types

Memory used for the data frame

A closer look at the data ...

#	Column	Non-Null Count	Dtype
0	Duration	169 non-null	int64
1	Pulse	169 non-null	int64
2	Maxpulse	169 non-null	int64
3	Calories	164 non-null	float64

There are 5 rows in the Calories column without data.

Nulls are bad!

Nulls = the wrong result when you analyze data

Let’s find the Nulls

```
import pandas as pd

df = pd.read_csv(‘data.csv’)

print(df.to_string())
```

Output: 17 27 91 118 141

Dropping the nulls #1

```
import pandas as pd

df = pd.read_csv('data.csv')
new_df = df.dropna()

print(new_df.to_string())
```

This DOES NOT change the original dataframe BECAUSE we are using **new_df**

Dropping the nulls #2

```
import pandas as pd

df = pd.read_csv('data.csv')
df = df.dropna(inplace = True)

print(df.to_string())
```

This changes the ORIGINAL

Replacing Nulls -- fillna

```
import pandas as pd

df = pd.read_csv('data.csv')

df = df.fillna(130, inplace = True)

print(df.to_string())
```

Replace NULL with 130

Replacing Nulls in Specific Columns

```
import pandas as pd

df = pd.read_csv('data.csv')

df["Calories"].fillna(130, inplace = True)

print(df.to_string())
```

Replace NULL in Calories

Replace the Nulls using mean, median, mode

- **Mean** = the average value
- **Median** = the value in the middle, after you have sorted all the values ascending
- **Mode** = the value that appears most frequently

Replace NULLS with MEAN

```
import pandas as pd
df = pd.read_csv('data.csv')

x = df["Calories"].mean()

df["Calories"].fillna(x, inplace = True)


Print(df.to_string())
```

Replace NULLS with MEDIAN

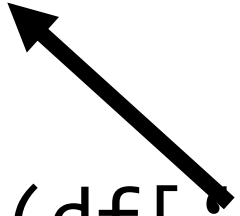
```
import pandas as pd
df = pd.read_csv('data.csv')

x = df["Calories"].median()

df["Calories"].fillna(x, inplace = True)


Print(df.to_string())
```

Replace NULLS with MODE

```
import pandas as pd
df = pd.read_csv('data.csv')

x = df["Calories"].mode()[0]

df["Calories"].fillna(x, inplace = True)


print(df.to_string())
```

Fixing dates

```
import pandas as pd
df = pd.read_csv('data1.csv')
print(df.to_string())

df[‘Date’] = pd.to_datetime(df[‘Date’])
df.dropna(subset=[‘Date’], inplace = True)
print(df.to_string())
```



Fixing wrong info

```
import pandas as pd
df = pd.read_csv('data1.csv')

print(df.to_string())
df.loc[9, ‘Duration’] = 45
```

We want to change line 9 Duration to be 45

Fixing wrong info in LARGE sets

```
import pandas as pd
df = pd.read_csv('data.csv')
for x in df.index:
    if df.loc[x, "Duration"] > 120:
        df.loc[x, "Duration"] = 120

print(df.to_string())
```

Removing rows in LARGE sets

```
import pandas as pd
df = pd.read_csv('data.csv')
for x in df.index:
    if df.loc[x, "Duration"] > 120:
        df.drop(x, inplace = True)

print(df.to_string())
```

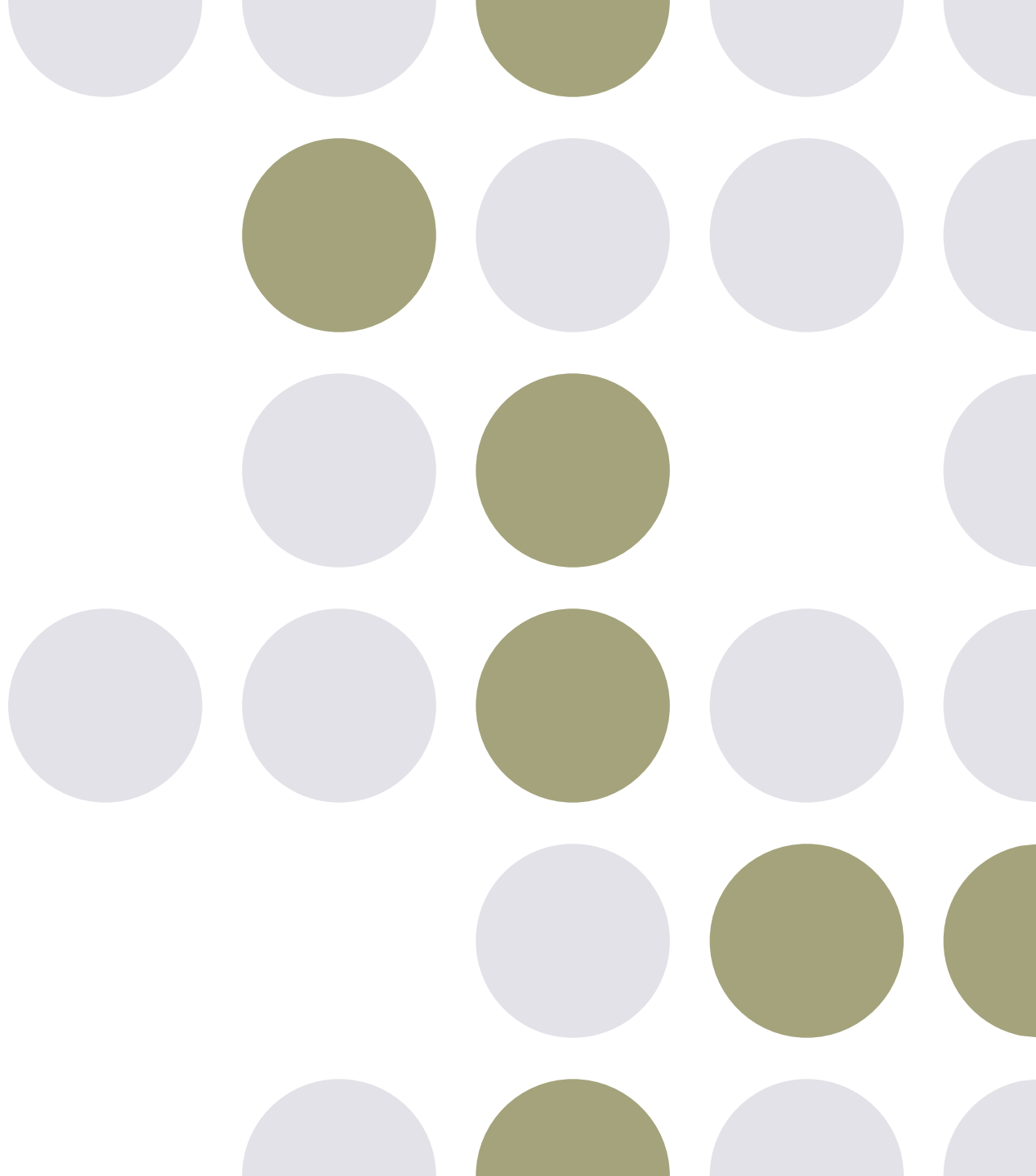
Finding all Duplicates

```
import pandas as pd

df = pd.read_csv('data1.csv')

print(df.duplicated())
```

Removing all Duplicates

```
import pandas as pd
df = pd.read_csv('data1.csv’)

df.drop_duplicates(inplace = True)

print(df.duplicated())
```

Plotting & Practice with Pandas



Plotting directly from a .csv

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv(“data.csv”)

df.plot()
plt.show()
```

Creating a scatter plot

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('data.csv')

df.plot(kind = "scatter", x = "Duration", y
= "Calories")
plt.show()
```

Creating a Histogram plot

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("data.csv")

df["Duration"].plot(kind ='hist')
plt.show()
```

Practice with .csv files

First, we need to gather our data

```
import pandas as pd
titanic_data = pd.read_csv(“titanic.csv”)

print(titanic_data.head())
```

Practice with .csv files

If you wanted to use the data straight from the web

```
import pandas as pd
titanic_data =
pd.read_csv("https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv")

print(titanic_data.head())
```

Customizing Headers

```
import pandas as pd

col_names = ["Id", "Survived",
        "Passenger Class", "Full Name",
        "Gender", "Age", "SibSp", "Parch",
        "Ticket Number", "Price", "Cabin", "Station"]

titanic_data =
pd.read_csv(r"C:\Users\User\Desktop\DAP2022\titanic.csv",
names = col_names)

print(titanic_data.head())
```

Skipping Rows

```
import pandas as pd


col_names = ["Id", "Survived",
        "Passenger Class", "Full Name",
        "Gender", "Age", "SibSp", "Parch",
        "Ticket Number", "Price", "Cabin", "Station"]


titanic_data =
pd.read_csv(r"C:\Users\User\Desktop\DAP2022\titanic.csv"
, names=col_names, skiprows=[0])

print(titanic_data.head())
```

```
import pandas as pd
col_names = ["Id", "Survived",
        "Passenger Class", "Full Name",
        "Gender", "Age", "SibSp",
        "Parch", "Ticket Number", "Price",
        "Cabin", "Station"]
titanic_data =
pd.read_csv(r"C:\Users\User\Desktop\DAP2022\titanic.csv"
, names=col_names, skiprows=[0])
titanic_data.to_csv('use_titanic.csv', index=False)
```

Saving to a new .csv file

Creating a .csv from scratch

```
import pandas as pd

cities = pd.DataFrame([["St. Louis",
"Missouri"], ["Atlanta", "Georgia"]],
columns=["City", "State"])

cities.to_csv('cities.csv')
```

Viewing the .csv file

```
import pandas as pd

df = pd.read_csv('cities.csv’)

print(df)
```

Saving the file without indexes

```
import pandas as pd
cities = pd.DataFrame([["St. Louis",
"Missouri"], ["Atlanta", "Georgia"]],
columns=["City", "State"])
cities.to_csv('cities.csv’, index=False)
df = pd.read_csv('cities.csv')
print(df)
```

Individual Practice

- Create your own .csv file with data and save it
- Make a change to the file and save a copy with a different name.
- Send both these files to your assigned TA in a Slack message