EE393 Python for Engineers

Dr. Orhan Gökçöl

orhan.gokcol@ozyegin.edu.tr

online

14.12.2020

2020-2021 Fall Semester

ICE BREAKER

Image Processing using Python

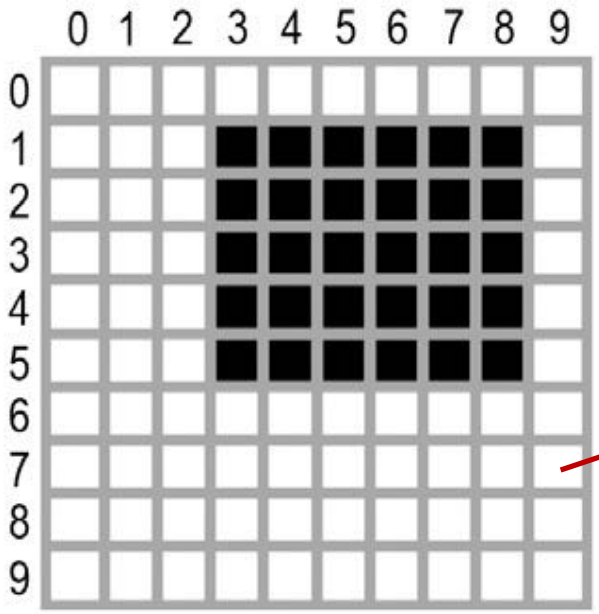
Pillow is a **Python** Imaging Library (PIL), which adds support for opening, manipulating, and saving images.

pillow

RGBA model

Each pixel contains 4 bytes of information for **red**, **green**, **blue** and transparency. Hence, color component has values between 0 and 255

PIXEL



Pixel : (7,9)

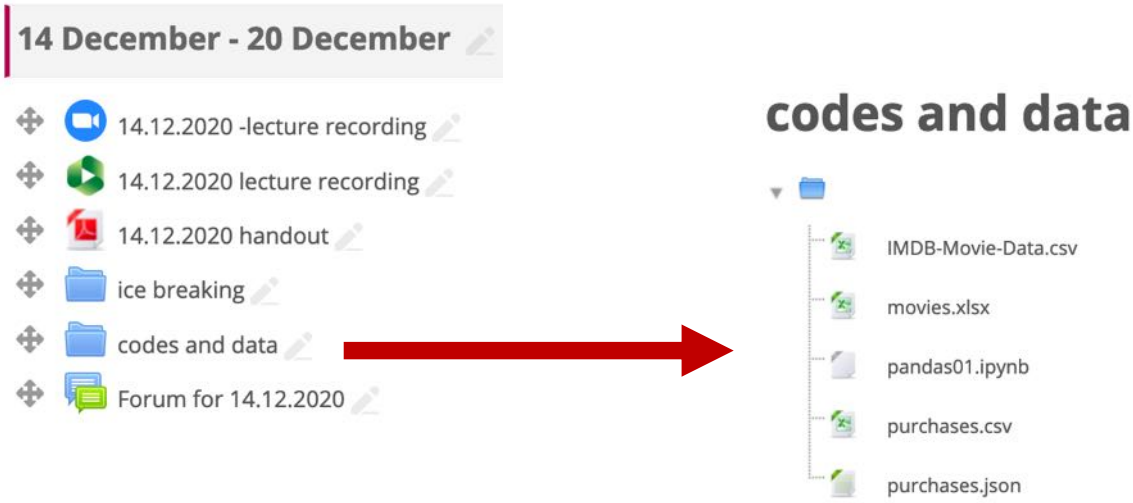
icebreaking/pillow.ipynb

Agenda

- Dealing with data
- Pandas
- Creating datasets
- Working with datasets
- Reading data from external sources: csv, xlsx, json

(Download to your local)

14.12.2020 | LMS resources



Learning objectives for 14.12.2020

- Knows basic principles of data science
- Knows how to deal with data sets using Pandas
- Knows how to import data from external resources to Pandas
- Nakes complex queries on data frame and analyse results

MIDTERM EXAM

DECEMBER 21, 2020; 08:40

- **Take Home | You'll have 4 hours to complete and submit your exam.**
- **Exam file will be in .docx/.pages/.odt formats. You'll pick one of them and answer questions on the file.**
- **Asking questions is not allowed.**

FINAL EXAM

JANUARY 15, 2021; 09:00

Online
(Details will be announced later)

Python Libraries for Data Science

pandas

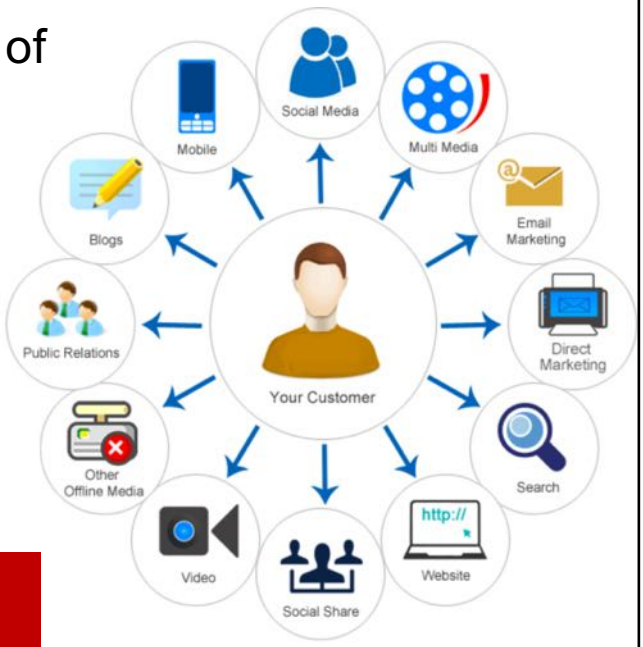
Pandas:

- adds data structures and tools designed to work with table-like data (similar to Series and Data Frames in R and Sheet structure in Excel)
- provides tools for data manipulation: reshaping, merging, sorting, slicing, aggregation etc.
- allows handling missing data

Link: http://pandas.pydata.org/

DATA

- The volume and variety of available data is overwhelming:
  - social media graphs
  - Facebook Likes
  - Tweets
  - auto registration
  - voting records, etc.



plus, we have business data

Example: Customer taxonomy analysis

Consider customer data.

- **Identity:** Can we identify them? Who are they?
- **History:** What's in their past? What have they done or achieved?
- **Proclivities:** What attracts them? What do they like?
- **Possessions:** What do they have, whether purchased, acquired, found, or made?
- **Activities:** Can we catch them in the act? What do they do and how do they do it?
- **Beliefs:** How do they feel and where do they stand on issues?

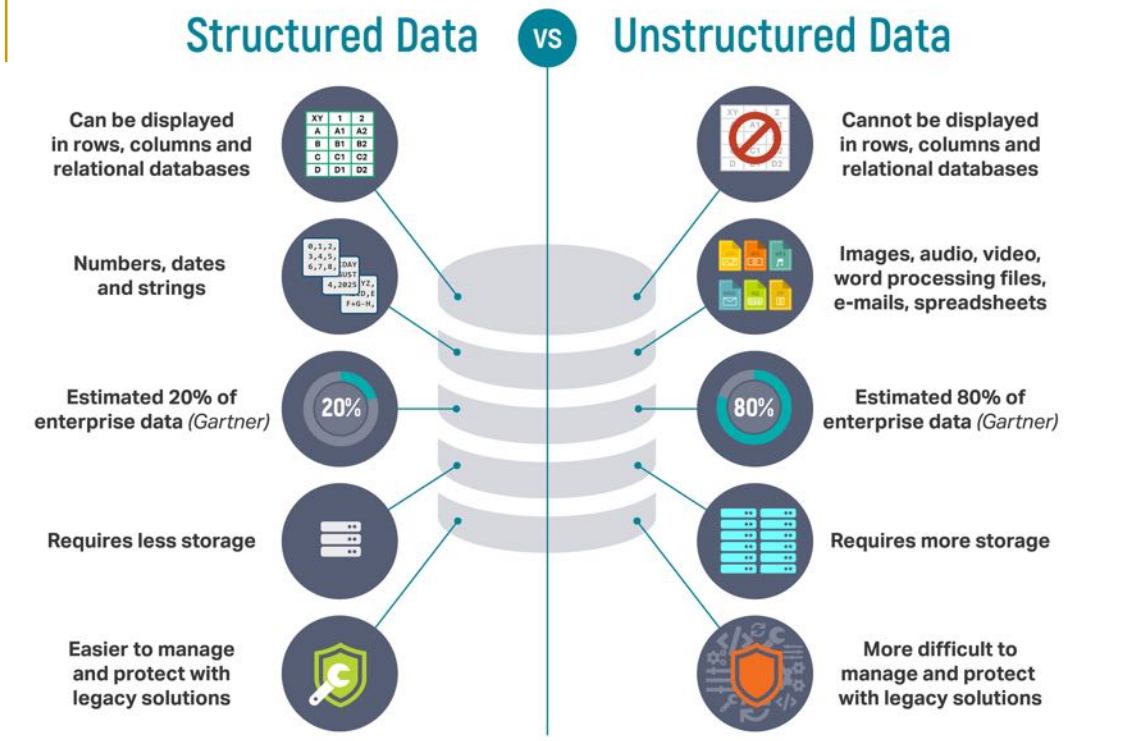
DATA IS EVERYTHING!

Online business

Where do the data come from????????

TRUTH: When asking for a specific page, the browser shares the following with the server:

- Time and date
- IP address
- Domain name
- Operating system and version
- Browser type and version
- **Pages requested => i.e. where did you click on?**
- Errors



Ref: https://lawtomated.com/structured-data-vs-unstructured-data-what-are-they-and-why-care/

Big data and AI

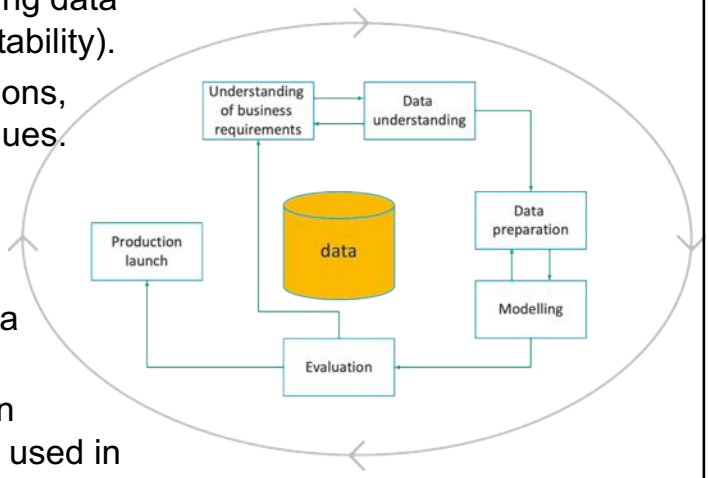
- Big Data will remain a critical part of many AI projects

Characteristics of big data:

- Volume
- Velocity
- Variety

CRISP-DM (Cross-Industry Standard Process for Data Mining) Methodology

- ✓ Understanding business conditions.
- ✓ Understanding the data (including data collecting and assessing its suitability).
- ✓ **Data preparation:** transformations, cleaning, removing extreme values.
- ✓ **Modeling** data for the purpose of developing the model.
- ✓ **Evaluating the model** and a decision whether it is ready for a production implementation.
- ✓ **Implementation** of the model in a production environment to be used in practice.



A group of experts, software developers, consultants, and academics created the CRISP-DM in 1990s

So, why do we use Pandas?

First and most importantly, for data cleaning purposes:	Statistics	Graphing
• Are there any missing data in the set? • Are data values meaningful? • Any extreme values? • Data visualization using tabular output and graphics • Data manipulation – removing values, concentation etc.	• Data summarization • Descriptive statistics • Correlations	• Matplotlib and seaborn integration

Pandas

- Pandas is an open source library built for Python programming language & **Numpy**, which provides high performance data analysis tools.
- In order to work with pandas in Python, you need to import pandas library in your python environment.
- Benefits of using Panda for Data Analysis
  - It can read or write in many different data formats(integer,float,double,etc.)
  - It can calculate in all ways data is organized, i.e., across rows and down columns.
  - It can easily select subsets of data from bulky data sets and even combine multiple datasets together.
  - It has functionality to find and fill missing data.
  - It supports advanced time-series functionality(Time series forecasting is the use of a model to predict future values based on previously observed values)

Importing libraries

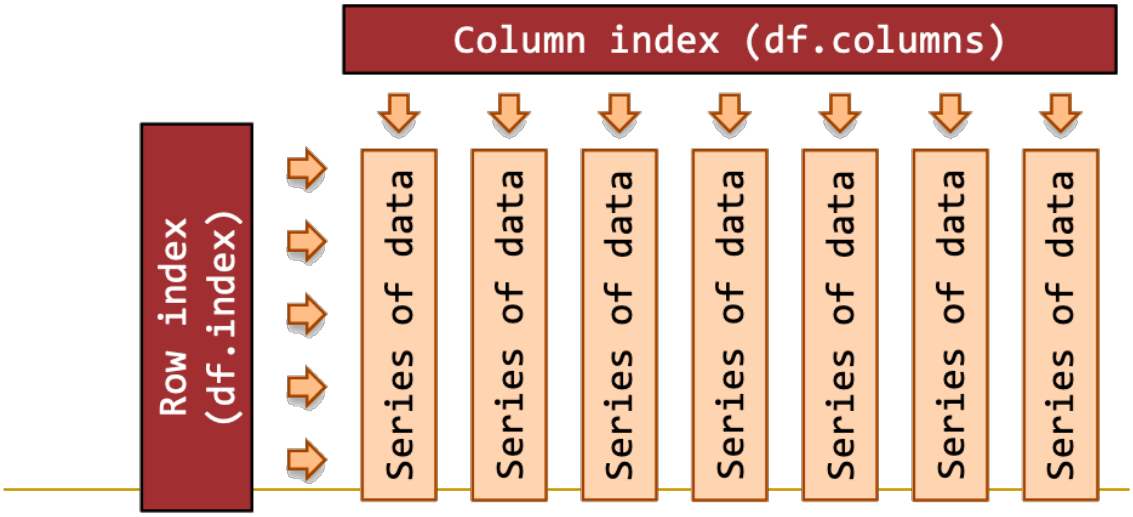
- Programmers generally use libraries as follows:

```
#Import Python Libraries
import numpy as np
import scipy as sp
import pandas as pd
#import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
```

Part of this handout is prepared using tutorial on https://www.learndatasci.com/tutorials and sample data(**IMDB** Database) is taken from Kaggle

The conceptual model

DataFrame object: The pandas DataFrame is a two-dimensional table of data with column and row indexes. The columns are made up of pandas Series objects.



Ref: Version 30 April 2017 - [Draft – Mark Graph – mark dot the dot graph at gmail dot com – @Mark_Graph on twitter]

Pandas – working with tabular data

pandas01.ipynb

```
import pandas as pd

d = {'col1': [1, 2, 3, 4, 7],
     'col2': [4, 5, 6, 9, 5],
     'col3': [7, 8, 12, 1, 11]
    }
df = pd.DataFrame(data=d)
print(df)
```

```
   col1  col2  col3
0     1     4     7
1     2     5     8
2     3     6    12
3     4     9     1
4     7     5    11
```

There are five rows. Each row, by default, will have row no.s starting from 0. It is also possible to give names to rows.

There are three columns. Each column, by default, will have column no.s starting from 0. It is also possible to give names to columns

```
#no of rows and columns
m = df.shape[1]
n = df.shape[0]
print(m, n)
```

```
3 5
```

Core components of pandas: Series and DataFrames

- The primary two components of pandas are the **Series** and **DataFrame**.
- A **Series** is essentially a column, and a **DataFrame** is a multi-dimensional table made up of a collection of Series. Both are modelled using numpy arrays.

You can create "Series" and "DataFrame" from the scratch of read data from some data resources like csv, json, xlsx, database …

Series

	apples
0	3
1	2
2	0
3	1

+

Series

	oranges
0	0
1	3
2	7
3	2

=

DataFrame

	apples	oranges
0	3	0
1	2	3
2	0	7
3	1	2

DataFrame is a 2-dimensional labeled data structure with columns of potentially different types. It is generally the most commonly used pandas object.

Creating data frames from the scratch

```
data = { 'apples':   [3, 2, 0, 1],
         'oranges': [0, 3, 7, 2]  }
```

It is a dictionary

*Then, pass it to pandas **DataFrame** as follows*

```
purchases = pd.DataFrame(data)
```

```
[2]: data = { 'apples':   [3, 2, 0, 1], 'oranges': [0, 3, 7, 2]  }
[3]: purchases = pd.DataFrame(data)
[4]: purchases
```

pandas01.ipynb

[4]:

	apples	oranges
0	3	0
1	2	3
2	0	7
3	1	2

DataFrame

- Each *(key, value)* item in data corresponds to a *column* in the resulting DataFrame.
- The **Index** of this DataFrame was given to us on creation as the numbers 0-3, but we could also create our own when we initialize the DataFrame.

Series

```
#working with series
import pandas as pd
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

print (s)
#retrieve multiple elements
print (s['a':'e':2])

#retrieve using index
print (s[2])
```

```
a    1
b    2
c    3
d    4
e    5
dtype: int64
a    1
c    3
e    5
dtype: int64
3
```

- A **Series** is a single vector of data (like a NumPy array) with an *index* that labels each element in the vector.
- **DataFrames consist of one or more Series**
- **Series are numpy arrays**

```
s1 = Series(range(0,4))  # -> 0, 1, 2, 3
s2 = Series(range(1,5))  # -> 1, 2, 3, 4
s3 = s1 + s2             # -> 1, 3, 5, 7
```


DataFrame

pandas.DataFrame(data, index, columns, dtype, copy)

data

data takes various forms like ndarray, series, map, lists, dict, constants and also another DataFrame.

index

For the row labels, the Index to be used for the resulting frame is Optional Default np.arange(n) if no index is passed.

columns

For column labels, the optional default syntax is - np.arange(n). This is only true if no index is passed.

dtype

Data type of each column.

copy

This command (or whatever it is) is used for copying of data, if the default is False.

DataFrame

- Each *(key, value)* item in data corresponds to a *column* in the resulting DataFrame.
- The **Index** of this DataFrame was given to us on creation as the numbers 0-3, but we could also create our own when we initialize the DataFrame.

```
purchases = pd.DataFrame(data, index=['June', 'Robert', 'Lily', 'David'])
purchases
```

	apples	oranges
June	3	0
Robert	2	3
Lily	0	7
David	1	2

Creating dataframe from a numpy array

```
#Create DataFrame from numpy array
tmp1 = np.array([[1, 2, 3], [2, 4, 6],
                 [3, 0, 4], [1, 1, 1]])
print (tmp1)
#create dataframe
# Create the dataframe
df = pd.DataFrame(tmp1)
print (df)
```

```
[[1 2 3]
 [2 4 6]
 [3 0 4]
 [1 1 1]]
   0  1  2
0  1  2  3
1  2  4  6
2  3  0  4
3  1  1  1
```

Creating a data frame from a properly-formatted python list

```
#creating a data frame from a list which contains properly formatted values
# initialize list of lists
data = [['tom', 10,'m'], ['nick', 15,'m'], ['juli', 14,'f']]

# Create the pandas DataFrame
df2 = pd.DataFrame(data, columns = ['Name', 'Age', 'Gender'])
df2
```

	Name	Age	Gender
0	tom	10	m
1	nick	15	m
2	juli	14	f

Creating data from list of dicts

```
# creating data from list of dicts
import pandas as pd

# Initialise data to lists.
data = [{'a': 1, 'b': 2, 'c':3}, {'a':10, 'b': 20, 'c': 30}]

# Creates DataFrame.
df = pd.DataFrame(data)

# Print the data
df
```

	a	b	c
0	1	2	3
1	10	20	30

```python
# Another df creation example
import pandas as pd

# Intitialise data of lists
data = [{'b': 2, 'c':3}, {'a': 10, 'b': 20, 'c': 30}]

# Creates padas DataFrame by passing lists of dictionaries and row index.
df = pd.DataFrame(data, index =['first', 'second'])

# Print the data - if there is some missing value in df, it becomes NaN
df
```

	b	c	a
first	2	3	NaN
second	20	30	10.0

- Index can explicitly be specified
- Any non-existent value becomes NaN


```python
#row and column indices at the same time
import pandas as pd

# Intitialise lists data.
data = [{'a': 1, 'b': 2}, {'a': 5, 'b': 10, 'c': 20}]

# With two column indices, values same as dictionary keys
df1 = pd.DataFrame(data, index=['first', 'second'], columns=['a', 'b'])

# With two column indices with one index with other name
df2 = pd.DataFrame(data, index=['first', 'second'], columns=['a', 'b1'])

# print for first data frame
print (df1, "\n")

# Print for second DataFrame.
print (df2)
```

```
        a   b
first   1   2
second  5  10

        a  b1
first   1 NaN
second  5 NaN
```

```python
#create a Series (i.e. a column data)
s = pd.Series([1,3,5,np.nan,6,8])
s1 = pd.Series(np.linspace(2,5,6))
```

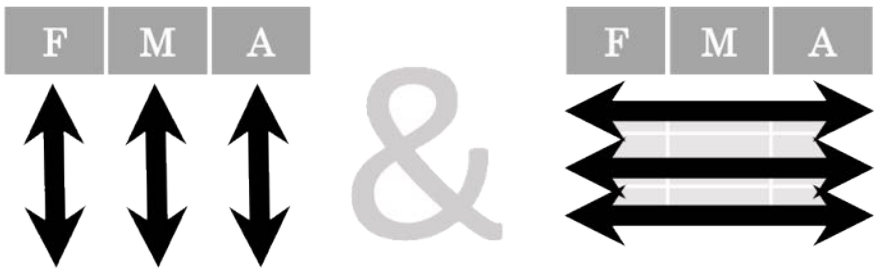
```python
#create an index column
dates = pd.date_range('20190101', periods=6)
mydata = { "dates": dates, 'First': s, 'Second': s1 }
```

```python
mydf = pd.DataFrame(mydata)
mydf
```

	dates	First	Second
0	2019-01-01	1.0	2.0
1	2019-01-02	3.0	2.6
2	2019-01-03	5.0	3.2
3	2019-01-04	NaN	3.8
4	2019-01-05	6.0	4.4
5	2019-01-06	8.0	5.0

Tidy data

In a tidy data set:



Each **variable** is saved in its own **column**

Each **observation** is saved in its own **row**

However, unfortunately, this is not what we have in real life. In real data sets, there are "missing data", "extreme" data (i.e. anamoly), "duplicate" data, "incomplete" data (you need to make it complete using other data sources) … and so on.

Hence, a lot of effort is needed to make data set clean, tidy and business ready. Remember that, business decisions are made based on facts which are derived from DATA –CORRECT DATA!!!

Adding a new column to a df

```python
import pandas as pd

# Define a dictionary containing Students data
data = {'Name': ['Ali', 'Natalie', 'Jon', 'Portman'],
        'Height': [1.87, 1.55, 1.70, 1.62],
        'Qualification': ['Msc', 'MA', 'Msc', 'PhD']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data)

# Declare a list that is to be converted into a column
address = ['Istanbul', 'USA', 'North', 'StarWars']

# Using 'Address' as the column name
# and equating it to the list
df['Address'] = address

# Observe the result
df
```

pandas01.ipynb

	Name	Height	Qualification	Address
0	Ali	1.87	Msc	Istanbul
1	Natalie	1.55	MA	USA
2	Jon	1.70	Msc	North
3	Portman	1.62	PhD	StarWars

Adding a new column to a df

```python
#add another column to an existing df using column location
# Using DataFrame.insert() to add a column
df.insert(2, "Age", [21, 23, 24, 21], True)
#if you allow duplication, put True; otherwise False

# Observe the result
df
```

	Name	Height	Age	Age	Qualification	Address
0	Ali	1.87	21	21	Msc	Istanbul
1	Natalie	1.55	23	23	MA	USA
2	Jon	1.70	24	24	Msc	North
3	Portman	1.62	21	21	PhD	StarWars

Dropping a row and column

```
x=df.drop([0,2])
x
```

Dropping rows

	Name	Height	Age	Qualification	Address
1	Natalie	1.55	23	MA	USA
3	Portman	1.62	21	PhD	StarWars

```
x=df.drop('Qualification', axis=1)
x
```

Dropping a column

	Name	Height	Age	Address
0	Ali	1.87	21	Istanbul
1	Natalie	1.55	23	USA
2	Jon	1.70	24	North
3	Portman	1.62	21	StarWars

Adding new rows to a df

```
import pandas as pd

# Creating the first Dataframe using dictionary
df1 = df = pd.DataFrame({"a":[1, 2, 3, 4],
                         "b":[5, 6, 7, 8]})

# Creating the Second Dataframe using dictionary
df2 = pd.DataFrame({"a":[1, 2, 3],
                    "b":[5, 6, 7]})
df3=df1.append(df2)
df3
```

	a	b
0	1	5
1	2	6
2	3	7
3	4	8
0	1	5
1	2	6
2	3	7

we could **loc**ate a customer's order by using their name:


```
purchases.loc['June']
```

```
apples      3
oranges     0
Name: June, dtype: int64
```

Reading from a csv file:

```
df = pd.read_csv('purchases.csv', sep=';', index_col=0)
```

```
df
```

	apples	oranges
June	3	0
Robert	2	3
Lily	0	7
David	1	2

purchases.csv

```
;apples;oranges
June;3;0
Robert;2;3
Lily;0;7
David;1;2
```


Reading data from a json file

```
#reading datafrom a json file
df = pd.read_json('purchases.json')
index_col = ['June', 'Robert', 'Lily', 'David']
df.index = index_col
df
```

	apples	oranges
June	3	0
Robert	2	3
Lily	0	7
David	1	2

purchases.json

```
{
"apples":   ["3", "2", "0", "1"],
"oranges": ["0", "3", "7", "2"]
}
```

Important df methods

Sample data we use: IMDB data taken from Kaggle

purchases.json | IMDB-Movie-Data.csv

```
Rank,Title,Genre,Description,Director,Actors,Year
1,Guardians of the Galaxy,"Action,Adventure,Sci-F
2,Prometheus,"Adventure,Mystery,Sci-Fi","Followin
3,Split,"Horror,Thriller",Three girls are kidnapp
4,Sing,"Animation,Comedy,Family","In a city of hu
5,Suicide Squad,"Action,Adventure,Fantasy",A secr
6,The Great Wall,"Action,Adventure,Fantasy",Europ
7,La La Land,"Comedy,Drama,Music",A jazz pianist
8,Mindhorn,Comedy,"A has-been actor best known fo
```

[18]:
```
movies_df = pd.read_csv("IMDB-Movie-Data.csv", index_col="Title")
```

[19]:
```
movies_df
```

[19]:

Title	Rank	Genre	Description	Director	Actors	Year	Runtime (Minutes)	Rating
Guardians of the Galaxy	1	Action,Adventure,Sci-Fi	A group of intergalactic criminals are forced ...	James Gunn	Chris Pratt, Vin Diesel, Bradley Cooper, Zoe S...	2014	121	8.1
Prometheus	2	Adventure,Mystery,Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott	Noomi Rapace, Logan Marshall-Green, Michael Fa...	2012	124	7.0

movies_df.head()

```
movies_df.head()
```

Title	Rank	Genre	Description	Director	Actors	Year	Runtime (Minutes)	Rating	Votes	Revenue (Millions)	Metascore
Guardians of the Galaxy	1	Action,Adventure,Sci-Fi	A group of intergalactic criminals are forced ...	James Gunn	Chris Pratt, Vin Diesel, Bradley Cooper, Zoe S...	2014	121	8.1	757074	333.13	76.0
Prometheus	2	Adventure,Mystery,Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott	Noomi Rapace, Logan Marshall-Green, Michael Fa...	2012	124	7.0	485820	126.46	65.0
Split	3	Horror,Thriller	Three girls are kidnapped by a man with a diag...	M. Night Shyamalan	James McAvoy, Anya Taylor-Joy, Haley Lu Richar...	2016	117	7.3	157606	138.12	62.0
Sing	4	Animation,Comedy,Family	In a city of humanoid animals, a hustling thea...	Christophe Lourdelet	Matthew McConaughey,Reese Witherspoon, Seth Ma...	2016	108	7.2	60545	270.32	59.0
Suicide Squad	5	Action,Adventure,Fantasy	A secret government agency recruits some of th...	David Ayer	Will Smith, Jared Leto, Margot Robbie, Viola D...	2016	123	6.2	393727	325.02	40.0

Try : head(7), head(12), tail()

Getting info about your data

```
movies_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1000 entries, Guardians of the Galaxy to Nine Lives
Data columns (total 11 columns):
Rank                  1000 non-null int64
Genre                 1000 non-null object
Description           1000 non-null object
Director              1000 non-null object
Actors                1000 non-null object
Year                  1000 non-null int64
Runtime (Minutes)     1000 non-null int64
Rating                1000 non-null float64
Votes                 1000 non-null int64
Revenue (Millions)    872 non-null float64
Metascore             936 non-null float64
dtypes: float64(3), int64(4), object(4)
memory usage: 93.8+ KB
```

Missing values!!!

.info() provides the essential details about your dataset, such as the number of rows and columns, the number of non-null values, what type of data is in each column, and how much memory your DataFrame is using.

Notice in our movies dataset we have some obvious missing values in the Revenue and Metascore columns.

shape

```
movies_df.shape
```

```
(1000, 11)
```

- Note that .shape has no parentheses and is a simple tuple of format (rows, columns). So we have **1000 rows** and **11 columns** in our movies DataFrame.
- You'll be going to .shape a lot when cleaning and transforming data. For example, you might filter some rows based on some criteria and then want to know quickly how many rows were removed.

Handling duplicates

```
temp_df = movies_df.append(movies_df)
temp_df.shape
```

```
(2000, 11)
```

```
temp_df = temp_df.drop_duplicates()
temp_df.shape
```

```
(1000, 11)
```

Or, (you don't need to make an assignment to a variable)

```
temp_df.drop_duplicates(inplace=True)
```


keep

Another important argument for **drop_duplicates()** is keep, which has three possible options:

- **first:** (default) Drop duplicates except for the first occurrence.
- **last:** Drop duplicates except for the last occurrence.
- **False:** Drop all duplicates.

Since we didn't define the keep argument in the previous example it was defaulted to first. This means that if two rows are the same pandas will drop the second row and keep the first row. Using last has the opposite effect: the first row is dropped.

- keep, on the other hand, will drop all duplicates if chosen as False. If two rows are the same then both will be dropped. Watch what happens to **temp_df:**

```
'''
Another important argument for drop_duplicates() is keep, which
first: (default) Drop duplicates except for the first occurrenc
last: Drop duplicates except for the last occurrence.
False: Drop all duplicates.
'''
temp_df = movies_df.append(movies_df)  # make a new copy
temp_df.drop_duplicates(inplace=True, keep=False)
temp_df.shape
```

```
(0, 11)
```

Column clean-up

- Many times datasets will have verbose column names with symbols, upper and lowercase words, spaces, and typos. To make selecting data by column name easier we can spend a little time cleaning up their names.
- Here's how to print the column names of our dataset:

```
movies_df.columns
```

```
Index(['Rank', 'Genre', 'Description', 'Director', 'Actors', 'Year',
       'Runtime (Minutes)', 'Rating', 'Votes', 'Revenue (Millions)',
       'Metascore'],
      dtype='object')
```

Now, lets make some column renaming :


```
#make some modifications on column names
movies_df.rename(columns={
        'Runtime (Minutes)': 'Runtime',
        'Revenue (Millions)': 'Revenue_millions'
    }, inplace=True)
movies_df.columns
```

```
Index(['Rank', 'Genre', 'Description', 'Director', 'Actors', 'Year', 'Runtime',
       'Rating', 'Votes', 'Revenue_millions', 'Metascore'],
      dtype='object')
```

```
#make column names all small letters
movies_df.columns = ['rank', 'genre', 'description', 'director', 'actors', 'year', 'runtime',
                     'rating', 'votes', 'revenue_millions', 'metascore']
movies_df.columns
```

```
Index(['rank', 'genre', 'description', 'director', 'actors', 'year', 'runtime',
       'rating', 'votes', 'revenue_millions', 'metascore'],
      dtype='object')
```

What about some pythoning :

```
movies_df.columns = [col.upper() for col in movies_df]
movies_df.columns
```

```
Index(['RANK', 'GENRE', 'DESCRIPTION', 'DIRECTOR', 'ACTORS', 'YEAR', 'RUNTIME',
       'RATING', 'VOTES', 'REVENUE_MILLIONS', 'METASCORE'],
      dtype='object')
```


Dealing with missing values: is_null( )

```
movies_df.isnull()
```

Title	RANK	GENRE	DESCRIPTION	DIRECTOR	ACTORS	YEAR	RUNTIME	RATING	VOTES	REVENUE_MILLIONS	METASCORE
Guardians of the Galaxy	False	False	False	False	False	False	False	False	False	False	False
Prometheus	False	False	False	False	False	False	False	False	False	False	False
Split	False	False	False	False	False	False	False	False	False	False	False
Sing	False	False	False	False	False	False	False	False	False	False	False
Suicide Squad	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...
Secret in Their Eyes	False	False	False	False	False	False	False	False	False	True	False
Hostel: Part II	False	False	False	False	False	False	False	False	False	False	False
Step Up 2: The Streets	False	False	False	False	False	False	False	False	False	False	False
Search Party	False	False	False	False	False	False	False	False	False	True	False
Nine Lives	False	False	False	False	False	False	False	False	False	False	False

1000 rows × 11 columns

To count the number of nulls in each column we use an aggregate function for summing:

```
movies_df.isnull().sum()
```

```
RANK                 0
GENRE                0
DESCRIPTION          0
DIRECTOR             0
ACTORS               0
YEAR                 0
RUNTIME              0
RATING               0
VOTES                0
REVENUE_MILLIONS   128
METASCORE           64
dtype: int64
```

We can see now that our data has **128** missing values for revenue_millions and **64** missing values for metascore.

Removing missing values

- Data Scientists and Analysts regularly face the dilemma of dropping or imputing null values, and is a decision that requires intimate knowledge of your data and its context. Overall, removing null data is only suggested if you have a small amount of missing data.
- Remove nulls is pretty simple:

```
movies_df.dropna(inplace=True)
movies_df.info()
```

When *inplace= True* is passed, the data is renamed in place (it returns nothing),

```
<class 'pandas.core.frame.DataFrame'>
Index: 838 entries, Guardians of the Galaxy to Nine Lives
Data columns (total 11 columns):
RANK                838 non-null int64
GENRE               838 non-null object
DESCRIPTION         838 non-null object
DIRECTOR            838 non-null object
ACTORS              838 non-null object
YEAR                838 non-null int64
RUNTIME             838 non-null int64
RATING              838 non-null float64
VOTES               838 non-null int64
REVENUE_MILLIONS    838 non-null float64
METASCORE           838 non-null float64
dtypes: float64(3), int64(4), object(4)
memory usage: 78.6+ KB
```

Dropping columns

- Other than just dropping rows, you can also drop columns with null values by setting axis=1:

```
movies_df.dropna(axis=1,inplace=True)
movies_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1000 entries, Guardians of the Galaxy to Nine Lives
Data columns (total 9 columns):
Rank                 1000 non-null int64
Genre                1000 non-null object
Description          1000 non-null object
Director             1000 non-null object
Actors               1000 non-null object
Year                 1000 non-null int64
Runtime (Minutes)    1000 non-null int64
Rating               1000 non-null float64
Votes                1000 non-null int64
dtypes: float64(1), int64(4), object(4)
memory usage: 78.1+ KB
```


Extracting a certain column from a df -Using square brackets is the general way we select columns in a DataFrame.

```
movies_df = pd.read_csv("IMDB-Movie-Data.csv", index_col="Title")
movies_df.rename(columns={
        'Runtime (Minutes)': 'Runtime',
        'Revenue (Millions)': 'Revenue_millions'
    }, inplace=True)
movies_df.columns = [col.lower() for col in movies_df]
revenue = movies_df['revenue_millions']
```

```
revenue.head()
```

```
Title
Guardians of the Galaxy    333.13
Prometheus                 126.46
Split                      138.12
Sing                       270.32
Suicide Squad              325.02
Name: revenue_millions, dtype: float64
```

Filling na values with the average

```
revenue_mean = revenue.mean()
revenue_mean
```

```
82.95637614678897
```

```
revenue.fillna(revenue_mean, inplace=True)
```

We have now replaced all nulls in revenue with the mean of the column. Notice that by using inplace=True we have actually affected the original movies_df:

```
movies_df.isnull().sum()
```

```
rank                0
genre               0
description         0
director            0
actors              0
year                0
runtime             0
rating              0
votes               0
revenue_millions    0
metascore          64
dtype: int64
```


Summary statistics with the df

```
movies_df.describe()
```

	rank	year	runtime	rating	votes	revenue_millions	metascore
count	1000.000000	1000.000000	1000.000000	1000.000000	1.000000e+03	1000.000000	936.000000
mean	500.500000	2012.783000	113.172000	6.723200	1.698083e+05	82.956376	58.985043
std	288.819436	3.205962	18.810908	0.945429	1.887626e+05	96.412043	17.194757
min	1.000000	2006.000000	66.000000	1.900000	6.100000e+01	0.000000	11.000000
25%	250.750000	2010.000000	100.000000	6.200000	3.630900e+04	17.442500	47.000000
50%	500.500000	2014.000000	111.000000	6.800000	1.107990e+05	60.375000	59.500000
75%	750.250000	2016.000000	123.000000	7.400000	2.399098e+05	99.177500	72.000000
max	1000.000000	2016.000000	191.000000	9.000000	1.791916e+06	936.630000	100.000000

Understanding which numbers are continuous also comes in handy when thinking about the type of plot to use to represent your data visually.

.describe() can also be used on a categorical variable to get the count of rows, unique count of categories, top category, and freq of top category:

```
movies_df['genre'].describe()
```

```
count                         1000
unique                         207
top       Action,Adventure,Sci-Fi
freq                            50
Name: genre, dtype: object
```

This tells us that the **genre** column has 207 unique values, the top value is Action/Adventure/Sci-Fi, which shows up 50 times (freq).

.value_counts() can tell us the frequency of all values in a column:

```
movies_df['genre'].value_counts().head(10)
```

```
Action,Adventure,Sci-Fi       50
Drama                         48
Comedy,Drama,Romance          35
Comedy                        32
Drama,Romance                 31
Comedy,Drama                  27
Animation,Adventure,Comedy    27
Action,Adventure,Fantasy      27
Comedy,Romance                26
Crime,Drama,Thriller          24
Name: genre, dtype: int64
```

Relationship between continus variables

- By using the correlation method **.corr()** we can generate the relationship between each continuous variable:

	rank	year	runtime	rating	votes	revenue_millions	metascore
rank	1.000000	-0.261605	-0.221739	-0.219555	-0.283876	-0.252996	-0.191869
year	-0.261605	1.000000	-0.164900	-0.211219	-0.411904	-0.117562	-0.079305
runtime	-0.221739	-0.164900	1.000000	0.392214	0.407062	0.247834	0.211978
rating	-0.219555	-0.211219	0.392214	1.000000	0.511537	0.189527	0.631897
votes	-0.283876	-0.411904	0.407062	0.511537	1.000000	0.607941	0.325684
revenue_millions	-0.252996	-0.117562	0.247834	0.189527	0.607941	1.000000	0.133328
metascore	-0.191869	-0.079305	0.211978	0.631897	0.325684	0.133328	1.000000

Subsetting

```
subset = movies_df[['genre', 'rating']]
subset.head()
```

Title	genre	rating
Guardians of the Galaxy	Action,Adventure,Sci-Fi	8.1
Prometheus	Adventure,Mystery,Sci-Fi	7.0
Split	Horror,Thriller	7.3
Sing	Animation,Comedy,Family	7.2
Suicide Squad	Action,Adventure,Fantasy	6.2

```python
#extracting a certain row located by name
prom = movies_df.loc["Prometheus"]
prom
```

```
rank                                                          2
genre                                   Adventure,Mystery,Sci-Fi
description         Following clues to the origin of mankind, a te...
director                                             Ridley Scott
actors              Noomi Rapace, Logan Marshall-Green, Michael Fa...
year                                                       2012
runtime                                                     124
rating                                                        7
votes                                                    485820
revenue_millions                                         126.46
metascore                                                     65
Name: Prometheus, dtype: object
```

```python
#extracting a certain row located by numerical index
prom = movies_df.iloc[1]
prom
```

```
rank                                                          2
genre                                   Adventure,Mystery,Sci-Fi
description         Following clues to the origin of mankind, a te...
director                                             Ridley Scott
actors              Noomi Rapace, Logan Marshall-Green, Michael Fa...
year                                                       2012
runtime                                                     124
rating                                                        7
votes                                                    485820
revenue_millions                                         126.46
metascore                                                     65
Name: Prometheus, dtype: object
```


Subsetting

```python
movie_subset = movies_df.loc['Prometheus':'Sing']
movie_subset = movies_df.iloc[1:4]
movie_subset
```

Title	rank	genre	description	director	actors	year	runtime	rating	votes	revenue_millions	metascore
Prometheus	2	Adventure,Mystery,Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott	Noomi Rapace, Logan Marshall-Green, Michael Fa...	2012	124	7.0	485820	126.46	65.0
Split	3	Horror,Thriller	Three girls are kidnapped by a man with a diag...	M. Night Shyamalan	James McAvoy, Anya Taylor-Joy, Haley Lu Richar...	2016	117	7.3	157606	138.12	62.0
Sing	4	Animation,Comedy,Family	In a city of humanoid animals, a hustling thea...	Christophe Lourdelet	Matthew McConaughey,Reese Witherspoon, Seth Ma...	2016	108	7.2	60545	270.32	59.0

Conditional selection

```
#conditional selection
condition = (movies_df['director'] == "Ridley Scott")
condition.head()
```

```
Title
Guardians of the Galaxy    False
Prometheus                  True
Split                      False
Sing                       False
Suicide Squad              False
Name: director, dtype: bool
```


Logic in Python (and pandas)			
<	Less than	!=	Not equal to
>	Greater than	df.column.isin(*values*)	Group membership
==	Equals	pd.isnull(*obj*)	Is NaN
<=	Less than or equals	pd.notnull(*obj*)	Is not NaN
>=	Greater than or equals	&,|,~,^,df.any(),df.all()	Logical and, or, not, xor, any, all

Conditions are given in [ ] and it makes filtering based on the conditions

```
movies_df[movies_df['director'] == "Ridley Scott"]
```

```
#First three moview with rating>=8.6
movies_df[movies_df['rating'] >= 8.6].head(3)
```

```
movies_df[(movies_df['director'] == 'Christopher Nolan') |
                    (movies_df['director'] == 'Ridley Scott')]
```

```
movies_df[movies_df['director'].isin(['Christopher Nolan', 'Ridley Scott'])]
```

How about this selection?

```
movies_df[movies_df['director'] == "Ridley Scott"]
```

Title	rank	genre	description	director	actors	year	runtime	rating	votes	revenue_millions	metascore
Prometheus	2	Adventure,Mystery,Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott	Noomi Rapace, Logan Marshall-Green, Michael Fa...	2012	124	7.0	485820	126.46	65.0
The Martian	103	Adventure,Drama,Sci-Fi	An astronaut becomes stranded on Mars after hi...	Ridley Scott	Matt Damon, Jessica Chastain, Kristen Wiig, Ka...	2015	144	8.0	556097	228.43	80.0
Robin Hood	388	Action,Adventure,Drama	In 12th century England, Robin and his band of...	Ridley Scott	Russell Crowe, Cate Blanchett, Matthew Macfady...	2010	140	6.7	221117	105.22	53.0
American Gangster	471	Biography,Crime,Drama	In 1970s America, a detective works to bring d...	Ridley Scott	Denzel Washington, Russell Crowe, Chiwetel Eji...	2007	157	7.8	337835	130.13	76.0
Exodus: Gods and Kings	517	Action,Adventure,Drama	The defiant leader Moses rises up against the ...	Ridley Scott	Christian Bale, Joel Edgerton, Ben Kingsley, S...	2014	150	6.0	137299	65.01	52.0
The Counselor	522	Crime,Drama,Thriller	A lawyer finds himself in over his head when h...	Ridley Scott	Michael Fassbender, Penélope Cruz, Cameron Dia...	2013	117	5.3	84927	16.97	48.0
A Good Year	531	Comedy,Drama,Romance	A British investment broker inherits his uncle...	Ridley Scott	Russell Crowe, Abbie Cornish, Albert Finney, M...	2006	117	6.9	74674	7.46	47.0
Body of Lies	738	Action,Drama,Romance	A CIA agent on the ground in Jordan hunts	Ridley Scott	Leonardo DiCaprio, Russell Crowe, Mark	2008	128	7.1	182305	39.38	57.0

How about this selection?

```
#First three moview with rating>=8.6
movies_df[movies_df['rating'] >= 8.6].head(3)
```

Title	rank	genre	description	director	actors	year	runtime	rating	votes	revenue_millions	metascore
Interstellar	37	Adventure,Drama,Sci-Fi	A team of explorers travel through a wormhole ...	Christopher Nolan	Matthew McConaughey, Anne Hathaway, Jessica Ch...	2014	169	8.6	1047747	187.99	74.0
The Dark Knight	55	Action,Crime,Drama	When the menace known as the Joker wreaks havo...	Christopher Nolan	Christian Bale, Heath Ledger, Aaron Eckhart,Mi...	2008	152	9.0	1791916	533.32	82.0
Inception	81	Action,Adventure,Sci-Fi	A thief, who steals corporate secrets through	Christopher Nolan	Leonardo DiCaprio, Joseph Gordon-Levitt, Ellen...	2010	148	8.8	1583625	292.57	74.0

```
#Complex selections using and/or operators; | is for OR; & is for AND
movies_df[(movies_df['director'] == 'Christopher Nolan') | (movies_df['director'] == 'Ridley Scott')].head()
```

Title	rank	genre	description	director	actors	year	runtime	rating	votes	revenue_millions	metascore
Prometheus	2	Adventure,Mystery,Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott	Noomi Rapace, Logan Marshall-Green, Michael Fa...	2012	124	7.0	485820	126.46	65.0
Interstellar	37	Adventure,Drama,Sci-Fi	A team of explorers travel through a wormhole ...	Christopher Nolan	Matthew McConaughey, Anne Hathaway, Jessica Ch...	2014	169	8.6	1047747	187.99	74.0
The Dark Knight	55	Action,Crime,Drama	When the menace known as the Joker wreaks havo...	Christopher Nolan	Christian Bale, Heath Ledger, Aaron Eckhart,Mi...	2008	152	9.0	1791916	533.32	82.0
The Prestige	65	Drama,Mystery,Sci-Fi	Two stage magicians engage in competitive one-...	Christopher Nolan	Christian Bale, Hugh Jackman, Scarlett Johanss...	2006	130	8.5	913152	53.08	66.0
Inception	81	Action,Adventure,Sci-Fi	A thief, who steals corporate secrets through	Christopher Nolan	Leonardo DiCaprio, Joseph Gordon-Levitt, Ellen...	2010	148	8.8	1583625	292.57	74.0


```
#isin is better to use
movies_df[movies_df['director'].isin(['Christopher Nolan', 'Ridley Scott'])].head()
```

Title	rank	genre	description	director	actors	year	runtime	rating	votes	revenue_millions	m
Prometheus	2	Adventure,Mystery,Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott	Noomi Rapace, Logan Marshall-Green, Michael Fa...	2012	124	7.0	485820	126.46	
Interstellar	37	Adventure,Drama,Sci-Fi	A team of explorers travel through a wormhole ...	Christopher Nolan	Matthew McConaughey, Anne Hathaway, Jessica Ch...	2014	169	8.6	1047747	187.99	
The Dark Knight	55	Action,Crime,Drama	When the menace known as the Joker wreaks havo...	Christopher Nolan	Christian Bale, Heath Ledger, Aaron Eckhart,Mi...	2008	152	9.0	1791916	533.32	
The Prestige	65	Drama,Mystery,Sci-Fi	Two stage magicians engage in competitive one-...	Christopher Nolan	Christian Bale, Hugh Jackman, Scarlett Johanss...	2006	130	8.5	913152	53.08	
Inception	81	Action,Adventure,Sci-Fi	A thief, who steals corporate secrets through ...	Christopher Nolan	Leonardo DiCaprio, Joseph Gordon-Levitt, Ellen...	2010	148	8.8	1583625	292.57	

```
#all movies that were released between 2005 and 2010, have a rating above 8.0,
#but made below the 25th percentile in revenue.
movies_df[
    ((movies_df['year'] >= 2005) & (movies_df['year'] <= 2010))
    & (movies_df['rating'] > 8.0)
    & (movies_df['revenue_millions'] < movies_df['revenue_millions'].quantile(0.25))
]
```

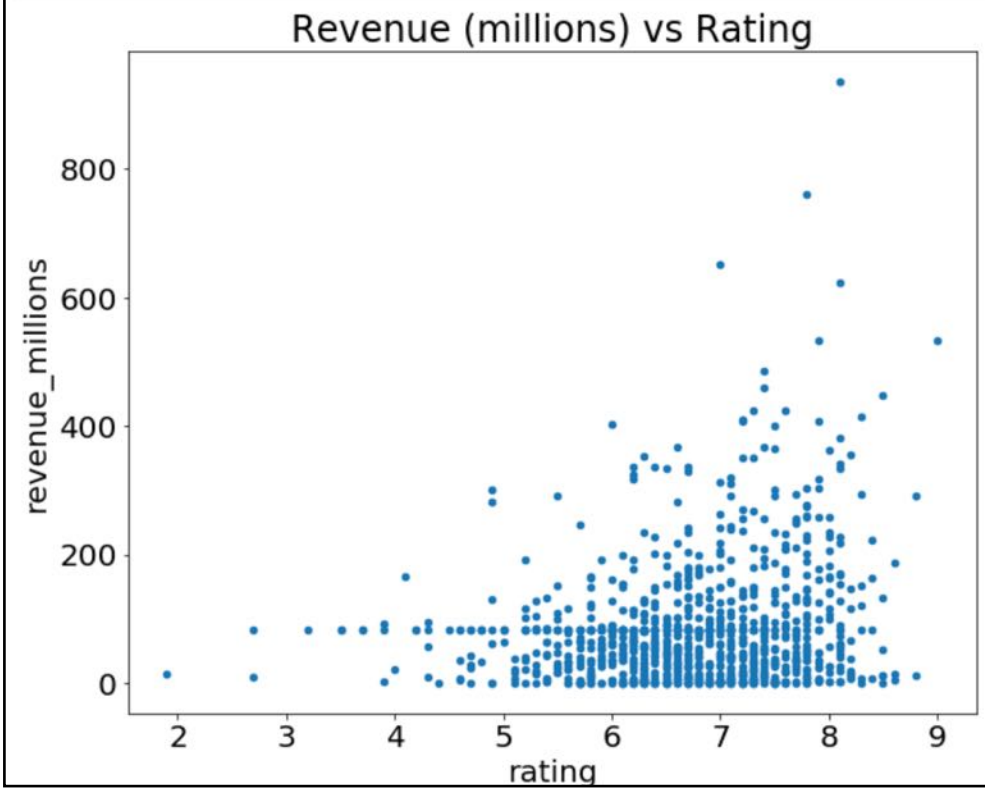
Title	rank	genre	description	director	actors	year	runtime	rating	votes	revenue_millions
3 Idiots	431	Comedy,Drama	Two friends are searching for their long lost ...	Rajkumar Hirani	Aamir Khan, Madhavan, Mona Singh, Sharman Joshi	2009	170	8.4	238789	6.52
The Lives of Others	477	Drama,Thriller	In 1984 East Berlin, an agent of the secret po...	Florian Henckel von Donnersmarck	Ulrich Mühe, Martina Gedeck,Sebastian Koch, Ul...	2006	137	8.5	278103	11.28
Incendies	714	Drama,Mystery,War	Twins journey to the Middle East to discover t...	Denis Villeneuve	Lubna Azabal, Mélissa Désormeaux-Poulin, Maxim...	2010	131	8.2	92863	6.86
Taare Zameen Par	992	Drama,Family,Music	An eight-year-old boy is thought to be a lazy ...	Aamir Khan	Darsheel Safary, Aamir Khan, Tanay Chheda, Sac...	2007	165	8.5	102697	1.20


```
#applying custom made functions
def rating_function(x):
    if x >= 8.0:
        return "good"
    else:
        return "bad"
```

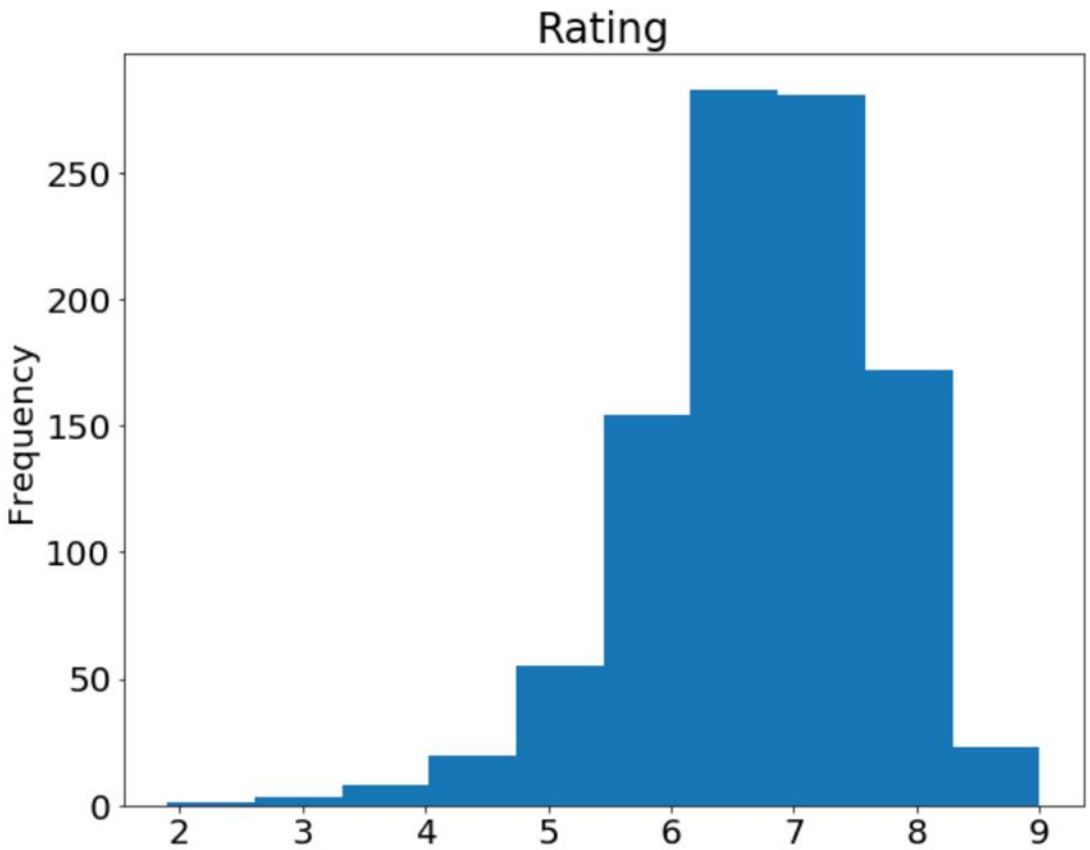
```
movies_df["rating_category"] = movies_df["rating"].apply(rating_function)
movies_df.head(2)
```

Title	rank	genre	description	director	actors	year	runtime	rating	votes	revenue_millions	metascore	rating_category
Guardians of the Galaxy	1	Action,Adventure,Sci-Fi	A group of intergalactic criminals are forced ...	James Gunn	Chris Pratt, Vin Diesel, Bradley Cooper, Zoe S...	2014	121	8.1	757074	333.13	76.0	good
Prometheus	2	Adventure,Mystery,Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott	Noomi Rapace, Logan Marshall-Green, Michael Fa...	2012	124	7.0	485820	126.46	65.0	bad

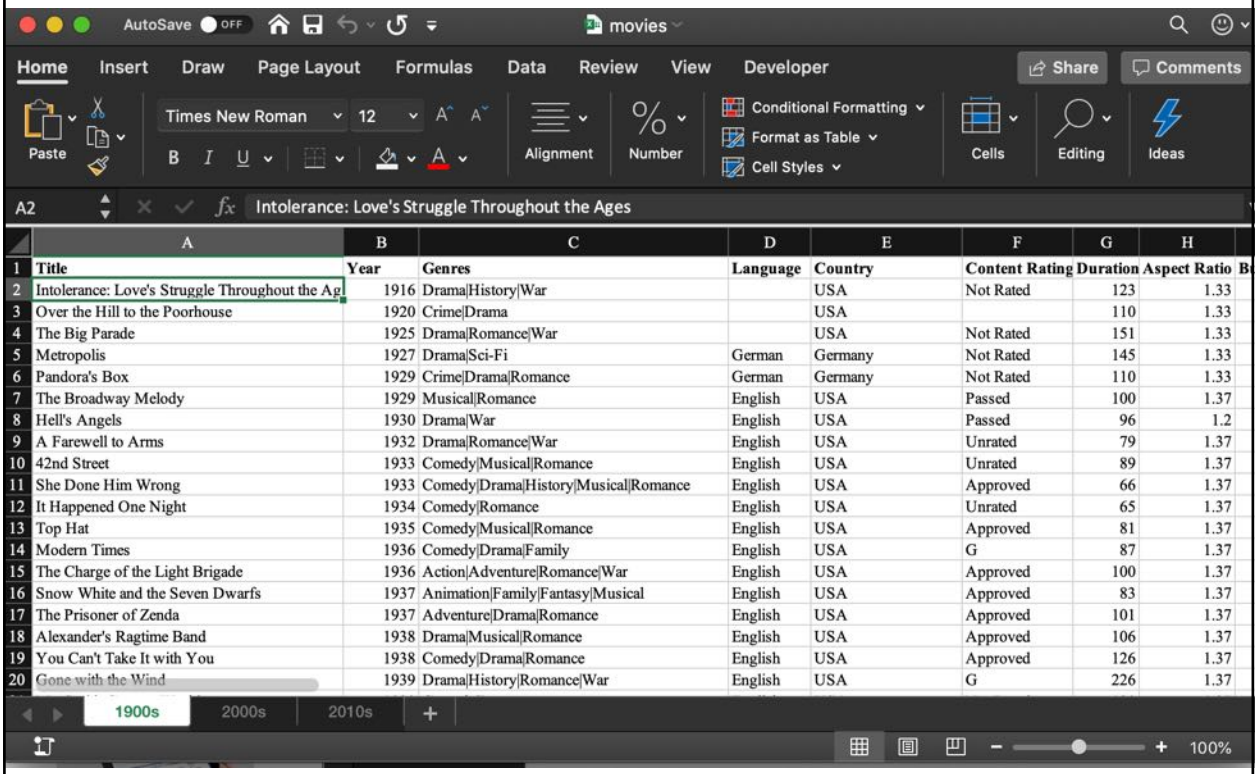
```python
#basic plotting
import matplotlib.pyplot as plt
# set font and plot size to be larger
plt.rcParams.update({'font.size': 20, 'figure.figsize': (10, 8)})
movies_df.plot(kind='scatter', x='rating', y='revenue_millions', title='Revenue (millions) vs Rating');
```




```python
movies_df['rating'].plot(kind='hist', title='Rating');
```



Reading from excel –movie database



	A	B	C	D	E	F	G	H
1	Title	Year	Genres	Language	Country	Content Rating	Duration	Aspect Ratio
2	Intolerance: Love's Struggle Throughout the Ag	1916	Drama|History|War		USA	Not Rated	123	1.33
3	Over the Hill to the Poorhouse	1920	Crime|Drama		USA		110	1.33
4	The Big Parade	1925	Drama|Romance|War		USA	Not Rated	151	1.33
5	Metropolis	1927	Drama|Sci-Fi	German	Germany	Not Rated	145	1.33
6	Pandora's Box	1929	Crime|Drama|Romance	German	Germany	Not Rated	110	1.33
7	The Broadway Melody	1929	Musical|Romance	English	USA	Passed	100	1.37
8	Hell's Angels	1930	Drama|War	English	USA	Passed	96	1.2
9	A Farewell to Arms	1932	Drama|Romance|War	English	USA	Unrated	79	1.37
10	42nd Street	1933	Comedy|Musical|Romance	English	USA	Unrated	89	1.37
11	She Done Him Wrong	1933	Comedy|Drama|History|Musical|Romance	English	USA	Approved	66	1.37
12	It Happened One Night	1934	Comedy|Romance	English	USA	Unrated	65	1.37
13	Top Hat	1935	Comedy|Musical|Romance	English	USA	Approved	81	1.37
14	Modern Times	1936	Comedy|Drama|Family	English	USA	G	87	1.37
15	The Charge of the Light Brigade	1936	Action|Adventure|Romance|War	English	USA	Approved	100	1.37
16	Snow White and the Seven Dwarfs	1937	Animation|Family|Fantasy|Musical	English	USA	Approved	83	1.37
17	The Prisoner of Zenda	1937	Adventure|Drama|Romance	English	USA	Approved	101	1.37
18	Alexander's Ragtime Band	1938	Drama|Musical|Romance	English	USA	Approved	106	1.37
19	You Can't Take It with You	1938	Comedy|Drama|Romance	English	USA	Approved	126	1.37
20	Gone with the Wind	1939	Drama|History|Romance|War	English	USA	G	226	1.37

Reading from excel

Reads first sheet. i.e. 1900s

```
[156]: #importing data from excel
       excel_file = 'movies.xlsx'
       movies = pd.read_excel(excel_file)
       movies
```

[156]:

	Title	Year	Genres	Language	Country	Content Rating	Duration	Aspect Ratio	Budget	Gross Earnings	...	Facebook Likes - Actor 1	Facebook Likes - Actor 2
0	Intolerance: Love's Struggle Throughout the Ages	1916	Drama|History|War	NaN	USA	Not Rated	123	1.33	385907.0	NaN	...	436	22
1	Over the Hill to the Poorhouse	1920	Crime|Drama	NaN	USA	NaN	110	1.33	100000.0	3000000.0	...	2	2
2	The Big Parade	1925	Drama|Romance|War	NaN	USA	Not Rated	151	1.33	245000.0	NaN	...	81	12
3	Metropolis	1927	Drama|Sci-Fi	German	Germany	Not Rated	145	1.33	6000000.0	26435.0	...	136	23
4	Pandora's Box	1929	Crime|Drama|Romance	German	Germany	Not Rated	110	1.33	NaN	9950.0	...	426	20
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1333	Twin Falls	1999	Drama	English	USA	R	111	1.85	500000.0	985341.0	...	980	505

Reading from excel –arbitrary sheet and specify the index

```
#reading from second sheet and specify index_col
movies_sheet1 = pd.read_excel(excel_file, sheet_name=1, index_col=0)
movies_sheet1.head()
```

Title	Year	Genres	Language	Country	Content Rating	Duration	Aspect Ratio	Budget	Gross Earnings	Director	...	Facebook Likes - Actor 1	Facebook Likes - Actor 2
102 Dalmatians	2000	Adventure|Comedy|Family	English	USA	G	100.0	1.85	85000000.0	66941559.0	Kevin Lima	...	2000.0	795.0
28 Days	2000	Comedy|Drama	English	USA	PG-13	103.0	1.37	43000000.0	37035515.0	Betty Thomas	...	12000.0	10000.0
3 Strikes	2000	Comedy	English	USA	R	82.0	1.85	6000000.0	9821335.0	DJ Pooh	...	939.0	706.0
Aberdeen	2000	Drama	English	UK	NaN	106.0	1.85	6500000.0	64148.0	Hans Petter Moland	...	844.0	2.0
All the Pretty Horses	2000	Drama|Romance|Western	English	USA	PG-13	220.0	2.35	57000000.0	15527125.0	Billy Bob Thornton	...	13000.0	861.0

5 rows × 24 columns

Reading from multiple sheets

```
xlsx = pd.ExcelFile(excel_file)
movies_sheets = [ ]
for sheet in xlsx.sheet_names:
        movies_sheets.append(xlsx.parse(sheet))
        movies = pd.concat(movies_sheets)
```

Sorting data

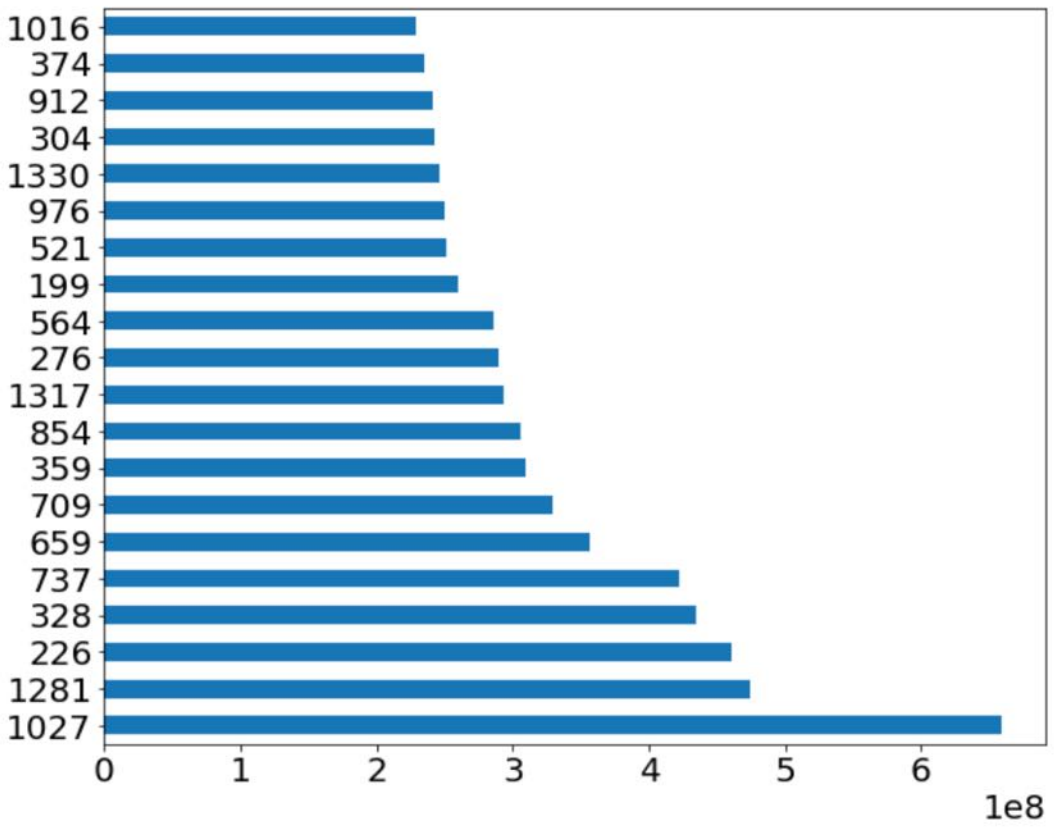
```
sorted_by_gross = movies.sort_values(['Gross Earnings'], ascending=False)
```

```
sorted_by_gross["Gross Earnings"].head(10)
```

```
1027    658672302.0
1281    474544677.0
226     460935665.0
328     434949459.0
737     422783777.0
659     356784000.0
709     329691196.0
359     309125409.0
854     306124059.0
1317    293501675.0
Name: Gross Earnings, dtype: float64
```



```
sorted_by_gross['Gross Earnings'].head(20).plot(kind="barh")
plt.show()
```



Question

- **What are the 25 most rated movies?**

SEE YOU NEXT WEEK!!!

DR. ORHAN GÖKÇÖL
gokcol@gmail.com
orhan.gokcol@ozyegin.edu.tr