Intro to pandas

DS3 Kaggle Session #1
Fall 2020

By Blake Walkowiak, Carolyn Duong, & Yash Potdar

What is pandas?

- pandas is a Python Library.
- Useful in data analysis and manipulation.
- Powerful in application, yet easily utilized… through practice of course !
- When working with large datasets, pandas can be a good tool.

To Get Started

- Download Anaconda. This will install Python, numpy, pandas, and allow you to use Jupyter notebooks.
- Follow the link to the Github repository https://github.com/bwalkowiak/DS3-Intro-Pandas-
- Download the empty and filled notebooks and Movies dataset.
- If using the empty notebook, load the .csv file you just downloaded into a new Jupyter notebook.

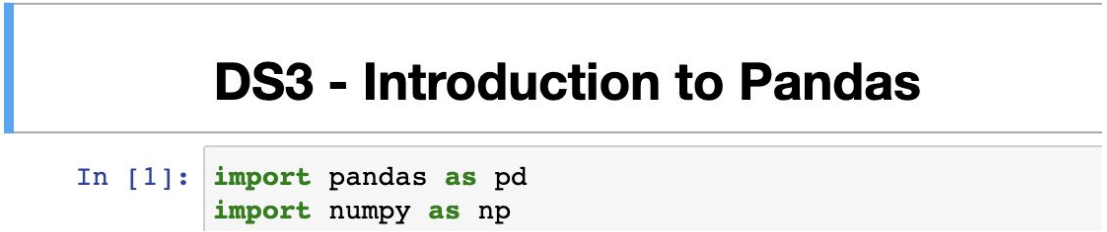
Jupyter Notebook

- Jupyter Notebook is an execution environment that supports Python programming.
- Allowing one to visualize data sets, manipulate data through cleaning and transformation, perform numerical simulation, and more!
- To create a new notebook, launch Jupyter Notebook from the Anaconda home page, then navigate to your desired directory and click ‘New’, then ‘Python 3’.


jupyter
Quit
Logout
Files
Running
Clusters
Select items to perform actions on them.
Upload
New
0
/ Downloads / DS3 / kaggle
Name
..
Intro.ipynb
Movies.csv
Notebook:
Python 3
Other:
Text File
Folder
Terminal

Importing Libraries

- Make sure the ‘Movies.csv’ is in the same directory as your notebook. To begin, import pandas and numpy into the first code cell.
- These libraries are crucial for our purposes. Working with Data!
- Run the cell by pressing ‘Run’ at the notebook menu bar or press Shift +Enter together.



Importing Data

- Let's read the csv into our Jupyter Notebook using the following code:

```
In [10]: df= pd.read_csv('Movies.csv')
         df.head()
```

Out[10]:

	Unnamed: 0	ID	Title	Year	Age	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Type	Directors	Genres	Country	
0	0	1	Inception	2010	13+	8.8	87%	1	0	0	0	0	Christopher Nolan	Action,Adventure,Sci-Fi,Thriller	United States,United Kingdom	English,Ja
1	1	2	The Matrix	1999	18+	8.7	87%	1	0	0	0	0	Lana Wachowski,Lilly Wachowski	Action,Sci-Fi	United States	
2	2	3	Avengers: Infinity War	2018	13+	8.5	84%	1	0	0	0	0	Anthony Russo,Joe Russo	Action,Adventure,Sci-Fi	United States	
3	3	4	Back to the Future	1985	7+	8.5	96%	1	0	0	0	0	Robert Zemeckis	Adventure,Comedy,Sci-Fi	United States	
4	4	5	The Good, the Bad and the Ugly	1966	18+	8.8	97%	1	0	1	0	0	Sergio Leone	Western	Italy,Spain,West Germany	

Heads or Tails

- **df.head()** and **df.tail()** are useful functions for displaying the first or last few rows of a DataFrame. Placing an integer value will produce that amount of rows. Default = 5.
- These allow for quick confirmation that data was loaded successfully.

** A tuple is a finite ordered list of elements. In our case (rows, columns)

Exploring the Data

- Various functions allow one to “dissect” a python DataFrame. For example, **df.shape** returns a tuple representing the dimensionality of your Dataframe.
- Another useful function is **df.dtypes**. Returned is a Series with the data types for each of the columns in your df.
- Try calling these functions in your notebook !

```
In [4]: df.shape
Out[4]: (16744, 16)

In [5]: df.dtypes
Out[5]: ID                 int64
        Title             object
        Year               int64
        Age               object
        IMDb             float64
        Rotten Tomatoes   object
        Netflix            int64
        Hulu               int64
        Prime Video        int64
        Disney+            int64
        Type               int64
        Directors         object
        Genres            object
        Country           object
        Language          object
        Runtime          float64
        dtype: object
```

Selecting a Column: Method 1

- **df** is the entire dataframe
- **Title** is the column name we want to access
- TO-DO: Pass the column name in quotes within the brackets following the dataframe

```
df['Title']
```

```
0                                  Inception
1                                 The Matrix
2                  Avengers: Infinity War
3                      Back to the Future
4          The Good, the Bad and the Ugly
                          ...
16739           The Ghosts of Buxley Hall
16740                       The Poof Point
16741                Sharks of Lost Island
16742                   Man Among Cheetahs
16743                     In Beaver Valley
Name: Title, Length: 16744, dtype: object
```

Selecting a Column: Method 2

- **df** is the entire dataframe
- **Title** is the column name we want to access
- TO-DO: Pass the column name following the dataframe and period

```
df.Title
```

```
0                                  Inception
1                                The Matrix
2                  Avengers: Infinity War
3                      Back to the Future
4          The Good, the Bad and the Ugly
                          ...
16739           The Ghosts of Buxley Hall
16740                        The Poof Point
16741               Sharks of Lost Island
16742                  Man Among Cheetahs
16743                    In Beaver Valley
Name: Title, Length: 16744, dtype: object
```

Selecting a Column: Method 3

- TO-DO: Pass the column name in quotes within the .get() method

```
df.get('Title')
```

```
0                                Inception
1                              The Matrix
2                  Avengers: Infinity War
3                      Back to the Future
4          The Good, the Bad and the Ugly
                        ...
16739          The Ghosts of Buxley Hall
16740                      The Poof Point
16741               Sharks of Lost Island
16742                  Man Among Cheetahs
16743                    In Beaver Valley
Name: Title, Length: 16744, dtype: object
```

** would not recommend this method UNLESS you need specific rows too

Selecting a Column: Method 4

- **.iloc** takes in [rows, columns]
- **:** specifies what slice we want
  - 0:5 → want elements in indices 0 to 5 (exclusive)
  - : → want all elements
- TO-DO: Pass the index of the column name following the rows and comma

```
df.iloc[:,2]
```

```
0                                  Inception
1                                The The Matrix
2                    Avengers: Infinity War
3                        Back to the Future
4            The Good, the Bad and the Ugly
                           ...
16739             The Ghosts of Buxley Hall
16740                         The Poof Point
16741                 Sharks of Lost Island
16742                    Man Among Cheetahs
16743                      In Beaver Valley
Name: Title, Length: 16744, dtype: object
```

** would not recommend this method UNLESS you need specific rows too

Selecting a Column: Method 5

- **.loc** takes in [rows, columns]
- **:** specifies what slice we want
  - 0:5 → want elements in indices 0 to 5 (exclusive)
  - : → want all elements
- TO-DO: Pass the column name following the rows and comma

```
df.loc[:,'Title']
```

```
0                                  Inception
1                                 The Matrix
2                  Avengers: Infinity War
3                      Back to the Future
4          The Good, the Bad and the Ugly
                          ...
16739           The Ghosts of Buxley Hall
16740                        The Poof Point
16741               Sharks of Lost Island
16742                  Man Among Cheetahs
16743                    In Beaver Valley
Name: Title, Length: 16744, dtype: object
```

Selecting Multiple Columns

- TO-DO: Pass the column names within *double* brackets
- NOTE: You can pass a *single* column name in single or double brackets, but you can not pass *multiple* column names in single brackets

```
df[['Title','Year']]
```

	Title	Year
0	Inception	2010
1	The Matrix	1999
2	Avengers: Infinity War	2018
3	Back to the Future	1985
4	The Good, the Bad and the Ugly	1966
...	...	...
16739	The Ghosts of Buxley Hall	1980
16740	The Poof Point	2001
16741	Sharks of Lost Island	2013
16742	Man Among Cheetahs	2017
16743	In Beaver Valley	1950

16744 rows × 2 columns

Data Structures: Series & Dataframe

Series: a single list with indices; a one-dimensional labeled array

Dataframe: a collection of more than one series; a two-dimensional labeled data structure

```
df['Title']
```

```
0                                Inception
1                              The Matrix
2                  Avengers: Infinity War
3                      Back to the Future
4          The Good, the Bad and the Ugly
                          ...
16739           The Ghosts of Buxley Hall
16740                       The Poof Point
16741                Sharks of Lost Island
16742                   Man Among Cheetahs
16743                     In Beaver Valley
Name: Title, Length: 16744, dtype: object
```

```
df[['Title']]
```

	Title
0	Inception
1	The Matrix
2	Avengers: Infinity War
3	Back to the Future
4	The Good, the Bad and the Ugly
...	...
16739	The Ghosts of Buxley Hall
16740	The Poof Point
16741	Sharks of Lost Island
16742	Man Among Cheetahs
16743	In Beaver Valley

16744 rows × 1 columns

```
df[['Title','Year']]
```

	Title	Year
0	Inception	2010
1	The Matrix	1999
2	Avengers: Infinity War	2018
3	Back to the Future	1985
4	The Good, the Bad and the Ugly	1966
...	...	...
16739	The Ghosts of Buxley Hall	1980
16740	The Poof Point	2001
16741	Sharks of Lost Island	2013
16742	Man Among Cheetahs	2017
16743	In Beaver Valley	1950

16744 rows × 2 columns

Selecting a Row: Methods 1 & 2

- TO-DO: Pass the row index within brackets
- Returns a *series*
- df.loc[0] gives the same result

```
df.iloc[0]
```

```
Unnamed: 0                                        0
ID                                                1
Title                                     Inception
Year                                           2010
Age                                             13+
IMDb                                            8.8
Rotten Tomatoes                                 87%
Netflix                                           1
Hulu                                              0
Prime Video                                       0
Disney+                                           0
Type                                              0
Directors                             Christopher Nolan
Genres                Action,Adventure,Sci-Fi,Thriller
Country                   United States,United Kingdom
Language                       English,Japanese,French
Runtime                                         148
Name: 0, dtype: object
```

.loc vs .iloc

.loc → retrieves rows and columns based on a *label* (or labels) that is/are passed

.iloc → retrieves rows and columns based on an *index* (or indices) that is/are passed

NOTE: .loc is different from normal Python slices because it includes both the start & stop

```
df.loc[0:5, 'Title']
```

```
0                                  Inception
1                                The Matrix
2                    Avengers: Infinity War
3                        Back to the Future
4           The Good, the Bad and the Ugly
5    Spider-Man: Into the Spider-Verse
Name: Title, dtype: object
```

```
df.iloc[0:5, 2]
```

```
0                                  Inception
1                                The Matrix
2                    Avengers: Infinity War
3                        Back to the Future
4           The Good, the Bad and the Ugly
Name: Title, dtype: object
```

Selecting a Row: Method 3

- TO-DO: Pass the row index within brackets
- 0:1 → from index 0 to 1 (exclusive), so basically just index 0
- Returns a *dataframe*

```
df[0:1]
```

	Unnamed: 0	ID	Title	Year	Age	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Type
0	0	1	Inception	2010	13+	8.8	87%	1	0	0	0	0

Practice Problem: iloc

Using **.iloc** method, how would you access the **year** *Avengers: Infinity War* was released (ie. third row, fourth column)?

NOTE: remember, indexing starts at 0, not 1

HINT: The format is df.iloc[row, column]

Solution: iloc

Remember, .iloc takes in INDICES, not strings. You must specify the indices for the rows and columns.

[row(s), column(s)]

Third Row → Index 2

Fourth Column → Index 3

```
df.iloc[2,3]
```

```
2018
```

Selecting Multiple Rows: Method 1

- TO-DO: Pass the row indices within brackets
- 0:5 → from index 0 to 5 (exclusive), so basically just indices 0 to 4
- Returns a *dataframe* that is the same as df.head()

```
#same as dataset.head()
df[0:5]
```

	Unnamed: 0	ID	Title	Year	Age	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Type	
0	0	1	Inception	2010	13+	8.8	87%	1	0	0	0	0	
1	1	2	The Matrix	1999	18+	8.7	87%	1	0	0	0	0	W
2	2	3	Avengers: Infinity War	2018	13+	8.5	84%	1	0	0	0	0	
3	3	4	Back to the Future	1985	7+	8.5	96%	1	0	0	0	0	
4	4	5	The Good, the Bad and the Ugly	1966	18+	8.8	97%	1	0	1	0	0	

Selecting Multiple Rows: Method 2

- TO-DO: With .iloc method, pass the row indices within brackets
- 0:5 → from index 0 to 5 (exclusive)
- Returns a *dataframe* that is the same as df.head()

```
df.iloc[0:5]
```

	Unnamed: 0	ID	Title	Year	Age	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Type
0	0	1	Inception	2010	13+	8.8	87%	1	0	0	0	0
1	1	2	The Matrix	1999	18+	8.7	87%	1	0	0	0	0
2	2	3	Avengers: Infinity War	2018	13+	8.5	84%	1	0	0	0	0
3	3	4	Back to the Future	1985	7+	8.5	96%	1	0	0	0	0
4	4	5	The Good, the Bad and the Ugly	1966	18+	8.8	97%	1	0	1	0	0

Selecting Multiple Rows: Method 3

- TO-DO: With .loc method, pass the row indices within brackets
- NOTE: .loc is different from normal Python slices because it includes both the start & stop
- Returns a *dataframe* that is NOT the same as df.head()

```
df.loc[0:5]
```

	Unnamed: 0	ID	Title	Year	Age	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Type
0	0	1	Inception	2010	13+	8.8	87%	1	0	0	0	0
1	1	2	The Matrix	1999	18+	8.7	87%	1	0	0	0	0
2	2	3	Avengers: Infinity War	2018	13+	8.5	84%	1	0	0	0	0
3	3	4	Back to the Future	1985	7+	8.5	96%	1	0	0	0	0
4	4	5	The Good, the Bad and the Ugly	1966	18+	8.8	97%	1	0	1	0	0
5	5	6	Spider-Man: Into the Spider-Verse	2018	7+	8.4	97%	1	0	0	0	0

Practice Problem: Selecting Multiple Rows

Using the *loc*, return the *fifth to tenth (inclusive) rows* of the dataframe. Feel free to do all if time allows.

NOTE: Think about how rows correspond with indices. What index is the first row, and so on?

Solution: Selecting Multiple Rows

Commands:

df[4:10]

df.iloc[4:10]

df.loc[4:9]

This is what your results should look like:

	Unnamed: 0	ID	Title	Year	Age	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Type	Directors	Genres	
4	4	5	The Good, the Bad and the Ugly	1966	18+	8.8	97%	1	0	1	0	0	Sergio Leone	Western	Ita
5	5	6	Spider-Man: Into the Spider-Verse	2018	7+	8.4	97%	1	0	0	0	0	Bob Persichetti,Peter Ramsey,Rodney Rothman	Animation,Action,Adventure,Family,Sci-Fi	
6	6	7	The Pianist	2002	18+	8.5	95%	1	0	1	0	0	Roman Polanski	Biography,Drama,Music,War	Kingdom,F
7	7	8	Django Unchained	2012	18+	8.4	87%	1	0	0	0	0	Quentin Tarantino	Drama,Western	
8	8	9	Raiders of the Lost Ark	1981	7+	8.4	95%	1	0	0	0	0	Steven Spielberg	Action,Adventure	
9	9	10	Inglourious Basterds	2009	18+	8.3	89%	1	0	0	0	0	Quentin Tarantino	Adventure,Drama,War	

Data Cleaning and Preparation

- *Data Cleaning*: correcting/removing erroneous values, filling in missing values
- Data Cleaning and Preparation take around 80% of a data scientist's time.
- *Garbage in → Garbage out* (GIGO): Your analysis can only be as good as the data going into it

Data Cleaning Part 1: Dropping Columns

- Notice any columns that seem redundant or aren't helpful? Use the **.drop()** function!

```
df.head(3)
```

	Unnamed: 0	ID	Title	Year	Age	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Type	Directors	Genres	Country	
0	0	1	Inception	2010	13+	8.8	87%	1	0	0	0	0	Christopher Nolan	Action,Adventure,Sci-Fi,Thriller	United States,United Kingdom	English,Japan
1	1	2	The Matrix	1999	18+	8.7	87%	1	0	0	0	0	Lana Wachowski,Lilly Wachowski	Action,Sci-Fi	United States	
2	2	3	Avengers: Infinity War	2018	13+	8.5	84%	1	0	0	0	0	Anthony Russo,Joe Russo	Action,Adventure,Sci-Fi	United States	

Dimensions

```
df.shape
```

```
(16744, 17)
```

```
df.drop(['Unnamed: 0', 'ID', 'Type'], axis = 1, inplace = True)
df.head(3)
```

	Title	Year	Age	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Directors	Genres	Country	Language	Runtime
0	Inception	2010	13+	8.8	87%	1	0	0	0	Christopher Nolan	Action,Adventure,Sci-Fi,Thriller	United States,United Kingdom	English,Japanese,French	148.0
1	The Matrix	1999	18+	8.7	87%	1	0	0	0	Lana Wachowski,Lilly Wachowski	Action,Sci-Fi	United States	English	136.0
2	Avengers: Infinity War	2018	13+	8.5	84%	1	0	0	0	Anthony Russo,Joe Russo	Action,Adventure,Sci-Fi	United States	English	149.0

```
df.shape
```

```
(16744, 14)
```

Data Cleaning Part 2: Dealing with Null Values

- First approach: Let's try getting rid of all the NaN values using the **.dropna()** function

```
df.dropna().head(3)
```

	Title	Year	Age	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Directors	Genres	Country	Language	Runtime
0	Inception	2010	13+	8.8	87%	1	0	0	0	Christopher Nolan	Action,Adventure,Sci-Fi,Thriller	United States,United Kingdom	English,Japanese,French	148.0
1	The Matrix	1999	18+	8.7	87%	1	0	0	0	Lana Wachowski,Lilly Wachowski	Action,Sci-Fi	United States	English	136.0
2	Avengers: Infinity War	2018	13+	8.5	84%	1	0	0	0	Anthony Russo,Joe Russo	Action,Adventure,Sci-Fi	United States	English	149.0

Dimensions

```
df.dropna().shape
```

```
(3301, 14)
```

- This is a prime example of why you need to understand the dataset and your end goal
  - Dropping 80% of the data is not cleaning data, it's destroying it 😞
  - Drop data wisely according to your end goal

Data Cleaning Part 2: Dealing with Null Values

- A better approach: Try using the parameters in the **.dropna()** function to drop what we want

- Problem Scenario: Create a distribution of movies comparing the IMDb ratings of movies across the major streaming platforms.

- Another Scenario: Create a distribution of the average rating (mean of the IMDb and Rotten Tomatoes ratings) of Netflix movies. Any ideas how we can drop the irrelevant rows?

```
df.dropna(how = 'all', inplace = True)
df.shape
```

```
(16744, 14)
```

```
df.dropna(subset = ['IMDb']).shape
```

```
(16173, 14)
```

```
df.dropna(subset = ['IMDb', 'Rotten Tomatoes']).shape
```

```
(5156, 14)
```

Data Cleaning Part 3: Renaming Columns

Method 1: Use the method **.rename()**

- Better for changing only a few column names
- Also useful if you want to use a function on all the column names like str.lower

```
m1 = df.rename(columns = {'Age':'Age Rating', 'Runtime':'Runtime (min)'})
m1.head(3)
```

	Title	Year	Age Rating	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Directors	Genres	Country	Language	Runtime (min)
0	Inception	2010	13+	8.8	87%	1	0	0	0	Christopher Nolan	Action,Adventure,Sci-Fi,Thriller	United States,United Kingdom	English,Japanese,French	148.0
1	The Matrix	1999	18+	8.7	87%	1	0	0	0	Lana Wachowski,Lilly Wachowski	Action,Sci-Fi	United States	English	136.0
2	Avengers: Infinity War	2018	13+	8.5	84%	1	0	0	0	Anthony Russo,Joe Russo	Action,Adventure,Sci-Fi	United States	English	149.0

Method 2: Use the attribute **.columns**

- Better for changing several column names
- Order is crucial!!!

```
df.columns
```

```
Index(['Title', 'Year', 'Age', 'IMDb', 'Rotten Tomatoes', 'Netflix', 'Hulu',
       'Prime Video', 'Disney+', 'Directors', 'Genres', 'Country', 'Language',
       'Runtime'],
      dtype='object')
```

```
df.columns = ['Title', 'Year', 'Age Rating', 'IMDb', 'Rotten Tomatoes', 'Netflix', 'Hulu',
       'Prime Video', 'Disney+', 'Directors', 'Genres', 'Country', 'Language',
       'Runtime (min)']
df.head(3)
```

	Title	Year	Age Rating	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Directors	Genres	Country	Language	Runtime (min)
0	Inception	2010	13+	8.8	87%	1	0	0	0	Christopher Nolan	Action,Adventure,Sci-Fi,Thriller	United States,United Kingdom	English,Japanese,French	148.0
1	The Matrix	1999	18+	8.7	87%	1	0	0	0	Lana Wachowski,Lilly Wachowski	Action,Sci-Fi	United States	English	136.0
2	Avengers: Infinity War	2018	13+	8.5	84%	1	0	0	0	Anthony Russo,Joe Russo	Action,Adventure,Sci-Fi	United States	English	149.0

Data Cleaning Part 4: Data Formatting

- Example: Rotten Tomatoes column values are Strings
  - Intuitively speaking, 100 > 99. However, Strings compare each digit so ‘100%’ < ‘99%’
  - We must fix this if we want to sort by Rotten Tomatoes later!
- Convert every value in the column to a numeric value by stripping the ‘%’ and casting the values

```
df['Rotten Tomatoes'] = df['Rotten Tomatoes'].str.strip('%')
df['Rotten Tomatoes'] = pd.to_numeric(df['Rotten Tomatoes'])
```

```
df.head(3)
```

	Title	Year	Age Rating	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Directors	Genres	Country	Language	Runtime (min)
0	Inception	2010	13+	8.8	87.0	1	0	0	0	Christopher Nolan	Action,Adventure,Sci-Fi,Thriller	United States,United Kingdom	English,Japanese,French	148.0
1	The Matrix	1999	18+	8.7	87.0	1	0	0	0	Lana Wachowski,Lilly Wachowski	Action,Sci-Fi	United States	English	136.0
2	Avengers: Infinity War	2018	13+	8.5	84.0	1	0	0	0	Anthony Russo,Joe Russo	Action,Adventure,Sci-Fi	United States	English	149.0

Conditionals

- Great tool for filtering out your dataframe to include only pertinent information
- Python comparison operators (==, !=, <, <=, >, >=) for each comparison
- Python bitwise operators (& for AND, | for OR) to combine comparisons
- Example: Filter out the movies dataset to include only the Disney+ movies.

Boolean Series

```
df['Disney+'] == 1
```

```
0         False
1         False
2         False
3         False
4         False
          ...
16739      True
16740      True
16741      True
16742      True
16743      True
Name: Disney+, Length: 16744, dtype: bool
```

Dataframe

```
disney = df[df['Disney+'] == 1]
disney.head()
```

	Title	Year	Age Rating	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Directors	Genres	Country
95	Saving Mr. Banks	2013	13+	7.5	79.0	1	0	0	1	John Lee Hancock	Biography,Comedy,Drama	United States,United Kingdom,Australia
103	Amy	2015	18+	7.8	95.0	1	0	1	1	NaN	Drama	United States
122	Bolt	2008	7+	6.8	89.0	1	0	0	1	Byron Howard,Chris Williams	Animation,Adventure,Comedy,Drama,Family	United States
125	The Princess and the Frog	2009	all	7.1	85.0	1	0	0	1	Ron Clements,John Musker	Animation,Adventure,Comedy,Family,Fantasy,Musi...	United States
150	Miracle	2004	7+	7.5	81.0	1	0	0	1	Gavin O'Connor	Biography,Drama,History,Sport	Canada,United States

```
disney.shape
```

```
(564, 14)
```

Conditionals

- Searching for a substring? Just use the Series function **.str.contains()**
- Example: Find all the movies that were directed by Steven Spielberg *and* were Family movies *or* Action movies.
- NOTE: we need to use **.str.contains()** in this case because a movie may have multiple directors or genres

Multiple Conditionals

```
director = df['Directors'].str.contains('Steven Spielberg')
family_genre = df['Genres'].str.contains('Family')
action_genre = df['Genres'].str.contains('Action')
df[director & (family_genre | action_genre)]
```

Condensed into 1 line

```
df[(df['Directors'].str.contains('Steven Spielberg')) &
   ((df['Genres'].str.contains('Family')) | (df['Genres'].str.contains('Action')))]
```

	Title	Year	Age Rating	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Directors	Genres	Country	
8	Raiders of the Lost Ark	1981	7+	8.4	95.0	1	0	0	0	Steven Spielberg	Action,Adventure	United States	English,(
16	Indiana Jones and the Last Crusade	1989	13+	8.2	88.0	1	0	0	0	Steven Spielberg	Action,Adventure	United States	
36	Minority Report	2002	13+	7.6	90.0	1	0	0	0	Steven Spielberg	Action,Crime,Mystery,Sci-Fi,Thriller	United States	
44	Indiana Jones and the Temple of Doom	1984	7+	7.6	85.0	1	0	0	0	Steven Spielberg	Action,Adventure	United States	
121	The Adventures of Tintin	2011	7+	7.3	74.0	1	0	0	0	Steven Spielberg	Animation,Action,Adventure,Family,Mystery	United States,New Zealand,United Kingdom	
186	War Horse	2011	13+	7.2	74.0	1	0	0	0	Steven Spielberg	Action,Adventure,Drama,History,War	United States,India	
219	Indiana Jones and the Kingdom of the Crystal S...	2008	13+	6.1	78.0	1	0	0	0	Steven Spielberg	Action,Adventure	United States	
16331	The BFG	2016	7+	6.4	75.0	0	0	0	1	Steven Spielberg	Adventure,Family,Fantasy	United States,India,United Kingdom	

Practice Problem: Conditionals

You're holding a club social virtually through Netflix Party next Friday. As all of your members are adults, you choose to watch a movie that is *suitable for '16+' or '18+'*. (This dataset doesn't do MPAA ratings like PG-13 or R). You also want to watch a *Comedy* movie that came out sometime *between 2000 and 2016 (inclusive)*. Also, you don't really have high expectations for this movie. Anything *rated above a 5.5 on IMDb* is perfect.

Create a "social" dataframe that has all the possible movies according to your criteria!

Solutions: Conditionals

```
nfx_pty = df['Netflix'] == 1
ages = (df['Age Rating'].str.contains('16+')) | (df['Age Rating'].str.contains('18+'))
genre = df['Genres'].str.contains('Comedy')
year_range = (df['Year'] >= 2000) & (df['Year'] <= 2016)
rating = df['IMDb'] > 5.5
social = df[nfx_pty & ages & genre & year_range & rating]
social.head(3)
```

```
social = df[(df['Netflix'] == 1) & ((df['Age Rating'].str.contains('16+')) | (df['Age Rating'].str.contains('18+')))
    & (df['Genres'].str.contains('Comedy')) & ((df['Year'] >= 2000) & (df['Year'] <= 2016))
    & (df['IMDb'] > 5.5)]
social.head(3)
```

	Title	Year	Age Rating	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Directors	Genres	Country	Languag
26	Silver Linings Playbook	2012	18+	7.7	92.0	1	0	0	0	David O. Russell	Comedy,Drama,Romance	United States	Englis
73	About Time	2013	18+	7.8	68.0	1	0	0	0	Richard Curtis	Comedy,Drama,Fantasy,Romance,Sci-Fi	United Kingdom	Englis
74	Kung Fu Hustle	2004	18+	7.7	90.0	1	0	0	0	Stephen Chow	Action,Comedy,Fantasy	Hong Kong,China,United States	Cantonese,Mandar

```
social.shape[0]
```

```
108
```

You have 108 possible options!

Another Practice Problem: Conditionals

You are babysitting your little sibling, who refuses to fall asleep until you watch a movie with him. You only have *Netflix and Prime Video* and want to watch *critically-acclaimed movies (IMDb of at least 8)*. You would prefer movies that were *released in 2000 or later*. You must show a movie *suitable for "all" ages* unless you want to be grounded. Since you have homework to do, the movie should be *1 ½ hours, at the most*.

Create a "babysitting" dataframe that has all the possible movies according to your criteria!

Solutions: Conditionals

```
platform = (df['Netflix'] == 1) | (df['Prime Video'] == 1)
rating = df['IMDb'] >= 8
year = df['Year'] >= 2000
age = df['Age Rating'] == 'all'
time = df['Runtime (min)'] <= 90
babysitting = df[platform & rating & year & age & time]
babysitting.head(3)
```

```
babysitting = df[((df['Netflix'] == 1) | (df['Prime Video'] == 1)) &
   (df['IMDb'] >= 8) & (df['Year'] >= 2000) & (df['Age Rating'] == 'all') &
    (df['Runtime (min)'] <= 90)]
babysitting.head(3)
```

	Title	Year	Age Rating	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Directors	Genres	Country	Language	Runtime (min)
536	I Am Kalam	2010	all	8.0	80.0	1	0	1	0	Nila Madhab Panda	Comedy,Drama,Family	India	Hindi	88.0
6987	If so	2003	all	8.1	NaN	0	0	1	0	Rajiv Whabi	Mystery	United Arab Emirates	English	80.0
8543	Fetish	2010	all	8.5	NaN	0	0	1	0	Soopum Sohn	Thriller	United States	English,Korean	87.0

```
babysitting.shape[0]
```

```
7
```

You have 7 possible options!

Sorting a Single Column

- We can use the **.sort_values()** function!
- Example: Let's sort the movies according to Rotten Tomatoes.

```
df.sort_values('Rotten Tomatoes', ascending = True).head(3)
```

	Title	Year	Age Rating	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Directors
8133	Kickin' It Old Skool	2007	13+	4.6	2.0	0	0	1	0	Harvey Glazer
6938	Strange Wilderness	2008	18+	5.3	2.0	0	0	1	0	Fred Wolf
4208	Getaway	2013	13+	4.4	2.0	0	1	0	0	Sam Peckinpah


```
df.sort_values(by = "Rotten Tomatoes", ascending = False).head(3)
```

	Title	Year	Age Rating	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Directors
481	Dance Academy: The Movie	2017	NaN	7.0	100.0	1	0	0	0	Jeffrey Walker
828	National Bird	2016	NaN	7.1	100.0	1	0	0	0	Sonia Kennebeck
6507	The Shelter	2015	NaN	3.6	100.0	0	0	1	0	María Lidón

Sorting Multiple Columns

- To do so, pass a list to the ‘by’ parameter!

```
df.sort_values(by = ["Rotten Tomatoes", 'IMDb'], ascending = False).head(3)
```

	Title	Year	Age Rating	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Directors	Genres	Country	Language	Runtime (min)
4473	Stop Making Sense	1984	NaN	8.6	100.0	0	0	1	0	Jonathan Demme	Documentary,Music	United States	English	88.0
4662	Tom Petty and the Heartbreakers: Runnin' Down ...	2007	NaN	8.6	100.0	0	0	1	0	Peter Bogdanovich	Documentary,Music	United States	English	239.0
4591	Mahanati	2018	7+	8.5	100.0	0	0	1	0	Nag Ashwin	Biography,Drama	India	Telugu,Tamil	177.0

- What if we want to sort columns in different directions? (Ex: finding critically-acclaimed classics)

Pass a list to the ‘ascending’ parameter!

```
df.sort_values(by = ['Rotten Tomatoes', 'Year'], ascending = [False, True]).head(3)
```

	Title	Year	Age Rating	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Directors	Genres	Country	Language	Runti (r
4467	A Trip to the Moon	1902	all	8.2	100.0	0	0	1	0	Georges Méliès	Short,Action,Adventure,Comedy,Fantasy,Sci-Fi	France	None,French	1
4470	The Cabinet of Dr. Caligari	1920	7+	8.1	100.0	0	0	1	0	Robert Wiene	Fantasy,Horror,Mystery,Thriller	Germany	German	7
4844	The Golem: How He Came into the World	1920	NaN	7.2	100.0	0	0	1	0	Carl Boese,Paul Wegener	Fantasy,Horror	Germany	NaN	7

Practice Problem: Sorting

There are still too many movies to choose from for your social. Since all 108 of the movies fit your criteria from before, you now decide to find the movies with the best Rotten Tomatoes ratings and IMDb ratings, in that order.

Create a “social_sorted” dataframe that has the top movies from the “social” dataframe. Then use the previous strategies for indexing and slicing to find the top 10 movies which your club can vote on!

Hint: The indices will not be in numerical order after you sort according to rating, so look for a method to reset the indices!

Solution: Sorting

Step 1: First sort the values accordingly!

```
social_sorted = social.sort_values(by = ['Rotten Tomatoes', 'IMDb'], ascending = False)
social_sorted.head(10)
```

	Title	Year	Age Rating	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Directors	Genres	Country	Language	Runtime (min)
141	Bill Burr: I'm Sorry You Feel That Way	2014	18+	8.4	100.0	1	0	0	0	Jay Karas	Comedy	United States	English	80.0
1014	Jim Gaffigan: Obsessed	2014	16+	7.6	100.0	1	0	0	0	Jay Chapman	Documentary,Comedy	United States	English	60.0
521	Melvin Goes to Dinner	2003	18+	6.8	100.0	1	0	0	0	Bob Odenkirk	Comedy,Drama,Romance	United States	English	83.0
884	Hannibal Buress: Comedy Camisado	2016	18+	6.6	100.0	1	0	0	0	Lance Bangs	Comedy	United States	English	83.0
473	Aśoka	2001	18+	6.5	100.0	1	0	0	0	NaN	Comedy	United States	English	30.0
332	Sour Grapes	2016	16+	7.3	96.0	1	0	0	0	Larry David	Comedy	United States	English	91.0
79	The Edge of Seventeen	2016	18+	7.3	94.0	1	0	0	0	Kelly Fremon Craig	Comedy,Drama	United States,China	English	104.0
244	The Death of Mr. Lazarescu	2005	18+	7.9	93.0	1	0	0	0	Cristi Puiu	Comedy,Drama	Romania	Romanian	153.0
26	Silver Linings Playbook	2012	18+	7.7	92.0	1	0	0	0	David O. Russell	Comedy,Drama,Romance	United States	English	122.0
134	Frances Ha	2013	18+	7.5	92.0	1	0	0	0	Noah Baumbach	Comedy,Drama,Romance	United States	French,English	86.0

Step 2: Try slicing!

```
social_sorted.loc[0:9, 'Title']
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/opt/anaconda3/lib/python3.7/site-packages/pandas/core/indexes/base.py in get_slice_bound(self,
   5166             try:
-> 5167                 return self._searchsorted_monotonic(label, side)
   5168             except ValueError:

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/indexes/base.py in _searchsorted_monotor
   5127
-> 5128         raise ValueError("index must be monotonic increasing or decreasing")
   5129

ValueError: index must be monotonic increasing or decreasing

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent call last)
<ipython-input-51-94b3a04541fc> in <module>
----> 1 social_sorted.loc[0:9, 'Title']

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/indexing.py in __getitem__(self, key)
   1416                 except (KeyError, IndexError, AttributeError):
   1417                     pass
-> 1418             return self._getitem_tuple(key)
   1419         else:
   1420             # we by definition only have the 0th axis

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/indexing.py in _getitem_tuple(self, tup)
    803     def _getitem_tuple(self, tup):
    804         try:
--> 805             return self._getitem_lowerdim(tup)
    806         except IndexingError:
    807             pass

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/indexing.py in _getitem_lowerdim(self, t
    959                     return section
    960                 # This is an elided recursive call to iloc/loc/etc'
--> 961                 return getattr(section, self.name)[new_key]
    962
    963         raise IndexingError("not applicable")
```

Step 3: Reset the index for slicing!

```
social_sorted.reset_index(drop = True, inplace = True)
social_sorted.head(3)
```

	Title	Year	Age Rating	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+
0	Bill Burr: I'm Sorry You Feel That Way	2014	18+	8.4	100.0	1	0	0	0
1	Jim Gaffigan: Obsessed	2014	16+	7.6	100.0	1	0	0	0
2	Melvin Goes to Dinner	2003	18+	6.8	100.0	1	0	0	0

Solution: Sorting

Step 4: Now try slicing!

```
social_sorted.loc[0:9, 'Title']
```

```
0    Bill Burr: I'm Sorry You Feel That Way
1                    Jim Gaffigan: Obsessed
2                     Melvin Goes to Dinner
3          Hannibal Buress: Comedy Camisado
4                                     Aśoka
5                               Sour Grapes
6                    The Edge of Seventeen
7                The Death of Mr. Lazarescu
8                   Silver Linings Playbook
9                                 Frances Ha
Name: Title, dtype: object
```

=

```
social_sorted.iloc[0:10, 0]
```

```
0    Bill Burr: I'm Sorry You Feel That Way
1                    Jim Gaffigan: Obsessed
2                     Melvin Goes to Dinner
3          Hannibal Buress: Comedy Camisado
4                                     Aśoka
5                               Sour Grapes
6                    The Edge of Seventeen
7                The Death of Mr. Lazarescu
8                   Silver Linings Playbook
9                                 Frances Ha
Name: Title, dtype: object
```

Exporting Data

- You can export your Pandas dataframe after your analysis!
- Several options: .to_csv(), .to_excel(), .to_html(), .to_json(), .to_string()
  - Full list on the Pandas documentation!
- Export location: your working directory
- Scenario: Your officer team loves the data analysis you did to find a good movie! Let's send our teammates a CSV file with the movies so they can make an easy decision next time.

```
social_sorted.to_csv('Club_Movie_Social.csv')
```

```
social_sorted.to_csv('Club_Movie_Social_wo_Index.csv', index = False)
```

Subtle Difference for Exporting

- Exporting data with the index leaves an extra column when you open it in Excel or read it into a new dataframe.

```
df1 = pd.read_csv("Club_Movie_Social.csv")
df1.head()
```

	Unnamed: 0	Title	Year	Age Rating	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Directors
0	0	Bill Burr: I'm Sorry You Feel That Way	2014	18+	8.4	100.0	1	0	0	0	Jay Karas
1	1	Jim Gaffigan: Obsessed	2014	16+	7.6	100.0	1	0	0	0	Jay Chapman
2	2	Melvin Goes to Dinner	2003	18+	6.8	100.0	1	0	0	0	Bob Odenkirk
3	3	Hannibal Buress: Comedy Camisado	2016	18+	6.6	100.0	1	0	0	0	Lance Bangs
4	4	Aśoka	2001	18+	6.5	100.0	1	0	0	0	NaN

```
df2 = pd.read_csv('Club_Movie_Social_wo_Index.csv')
df2.head()
```

	Title	Year	Age Rating	IMDb	Rotten Tomatoes	Netflix	Hulu	Prime Video	Disney+	Directors
0	Bill Burr: I'm Sorry You Feel That Way	2014	18+	8.4	100.0	1	0	0	0	Jay Karas
1	Jim Gaffigan: Obsessed	2014	16+	7.6	100.0	1	0	0	0	Jay Chapman
2	Melvin Goes to Dinner	2003	18+	6.8	100.0	1	0	0	0	Bob Odenkirk
3	Hannibal Buress: Comedy Camisado	2016	18+	6.6	100.0	1	0	0	0	Lance Bangs
4	Aśoka	2001	18+	6.5	100.0	1	0	0	0	NaN

We hope this workshop was both helpful and enjoyable.
Stay safe and good luck with classes!

THE END