Introduction to Pandas for absolute beginners

The pandas package is the backbone of most data projects. The word pandas is an acronym which is derived from "Python and data analysis" and "panel data"

Pandas First Steps: install & import

In [9]:
```
!pip install pandas
```

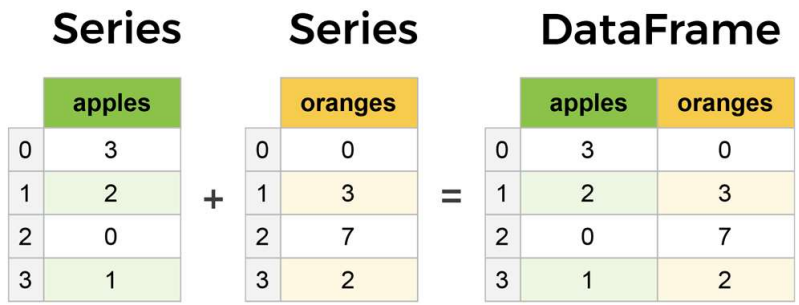
```
Requirement already satisfied: pandas in c:\users\asus\anaconda3\lib\site-packages (1.4.4)
Requirement already satisfied: numpy>=1.18.5 in c:\users\asus\anaconda3\lib\site-packages (from pandas) (1.2
1.5)
Requirement already satisfied: pytz>=2020.1 in c:\users\asus\anaconda3\lib\site-packages (from pandas) (202
2.1)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\asus\anaconda3\lib\site-packages (from pan
das) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\asus\anaconda3\lib\site-packages (from python-dateutil>=
2.8.1->pandas) (1.16.0)
```

In [10]:
```
import pandas as pd
```

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.



Creating DataFrames from scratch using "dictionaries"

Let's say we have a fruit stand that sells apples and oranges.

We want to have a **column for each fruit** and a **row for each customer purchase**.

To organize this as a dictionary for pandas we could do something like:

In [12]:
```
data = {
    'apples': [3,2,0,1],
    'oranges': [0,3,7,2]
}
```

And then pass it to the pandas DataFrame constructor:

In [15]:
```
purchases= pd.DataFrame(data)
purchases
```

Out[15]:

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

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.

Let's have customer names as our index:

In [17]:
```
purchases = pd.DataFrame(data,index = ['June', 'Robert', 'Lily', 'David'])
purchases
```

Out[17]:

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

So now we could locate a customer's order by using their name or their numerical index:

In [31]:
```
purchases.loc['June']
```

Out[31]:
```
apples     3
oranges    0
Name: June, dtype: int64
```

In [23]:
```
purchases.iloc[0]
```

Out[23]:

	apples	oranges
June	3	0
Robert	2	3

Reading data from csv

(Note you can read from JSON & SQL as well)

In [39]:
```
movies_df = pd.read_csv("IMDB-Movie-Data.csv")
movies_df.head(3)
```

Out[39]:

	Rank	Title	Genre	Description	Director	Actors	Year	Runtime (Minutes)	Rating	Votes	Revenue (Millions)	Metascore
0	1	Guardians of the Galaxy	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
1	2	Prometheus	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
2	3	Split	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

Getting info about your data

.info() should be one of the very first commands you run after loading your data. .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. We'll look at how to handle those in a bit.

In [40]:
```
movies_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Rank                1000 non-null   int64  
 1   Title               1000 non-null   object 
 2   Genre               1000 non-null   object 
 3   Description         1000 non-null   object 
 4   Director            1000 non-null   object 
 5   Actors              1000 non-null   object 
 6   Year                1000 non-null   int64  
 7   Runtime (Minutes)   1000 non-null   int64  
 8   Rating              1000 non-null   float64
 9   Votes               1000 non-null   int64  
 10  Revenue (Millions)  872 non-null    float64
 11  Metascore           936 non-null    float64
dtypes: float64(3), int64(4), object(5)
memory usage: 93.9+ KB
```

In [41]:
```
movies_df.shape
```

Out[41]:
```
(1000, 12)
```

Dropping duplicates

In [ ]:
```
movies_df.drop_duplicates(keep="first")
```

An 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.

Reading/changing columns

In [53]:
```
movies_df.columns
```

Out[53]:
```
Index(['Rank', 'Title', 'Genre', 'Description', 'Director', 'Actors', 'Year',
       'Runtime (Minutes)', 'Rating', 'Votes', 'Revenue (Millions)',
       'Metascore'],
      dtype='object')
```

In [56]:
```
movies_df.columns = [item.lower() for item in movies_df.columns]
movies_df.columns
```

Out[56]:
```
Index(['rank', 'title', 'genre', 'description', 'director', 'actors', 'year',
       'runtime (minutes)', 'rating', 'votes', 'revenue (millions)',
       'metascore'],
      dtype='object')
```

working with missing values

When exploring data, you'll most likely encounter missing or null values, which are essentially placeholders for non-existent values. Most commonly you'll see Python's None or NumPy's np.nan.

There are two options in dealing with nulls:

- Get rid of rows or columns with nulls
- Replace nulls with non-null values, a technique known as imputation

which cells in our DataFrame are null?

In [111]:
```
movies_df.isnull().head(5)
```

Out[111]:

	rank	title	genre	description	director	actors	year	runtime (minutes)	rating	votes	revenue	metascore	rating_category
0	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False

count the number of nulls in each column ?

In [61]:
```
movies_df.isnull().sum()
```

Out[61]:
```
rank                   0
title                  0
genre                  0
description            0
director               0
actors                 0
year                   0
runtime (minutes)      0
rating                 0
votes                  0
revenue (millions)   128
metascore             64
dtype: int64
```

Removing rows with null values:

In [110]:
```
movies_df.dropna().head(5)
```

Out[110]:

	rank	title	genre	description	director	actors	year	runtime (minutes)	rating	votes	revenue	me
0	1	Guardians of the Galaxy	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	
1	2	Prometheus	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	
2	3	Split	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	
3	4	Sing	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	
4	5	Suicide Squad	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	

Removing columns with null values:

In [109]:
```
movies_df.dropna(axis=1).head(5)
```

Out[109]:

	rank	title	genre	description	director	actors	year	runtime (minutes)	rating	votes	revenue	rat
0	1	Guardians of the Galaxy	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	
1	2	Prometheus	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	
2	3	Split	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	
3	4	Sing	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	
4	5	Suicide Squad	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	

Imputation

Imputation is a conventional feature engineering technique used to keep valuable data that have null values.

There may be instances where dropping every row with a null value removes too big a chunk from your dataset, so instead we can impute that null with another value, usually the **mean** or the **median** of that column.

In [76]:
```
movies_df = movies_df.rename(columns={'revenue (millions)':'revenue'})
movies_df['revenue'].fillna(movies_df['revenue'].mean(),inplace=True)
movies_df.isnull().sum()
```

Out[76]:
```
rank                  0
title                 0
genre                 0
description           0
director              0
actors                0
year                  0
runtime (minutes)     0
rating                0
votes                 0
revenue               0
metascore            64
dtype: int64
```

Understanding your variables

a summary of the distribution of continuous variables:

In [77]:
```
movies_df.describe()
```

Out[77]:

	rank	year	runtime (minutes)	rating	votes	revenue	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

a summary of a categorical variable:

In [78]:
```
movies_df['genre'].describe()
```

Out[78]:
```
count                        1000
unique                        207
top       Action,Adventure,Sci-Fi
freq                           50
Name: genre, dtype: object
```

the frequency of all values in a column:

In [80]:
```
movies_df['genre'].value_counts().head(5)
```

Out[80]:
```
Action,Adventure,Sci-Fi    50
Drama                      48
Comedy,Drama,Romance       35
Comedy                     32
Drama,Romance              31
Name: genre, dtype: int64
```

DataFrame slicing, selecting, extracting

extract a column using square brackets:

In [85]:
```
# one column
movies_df['genre']
```

Out[85]:
```
0       Action,Adventure,Sci-Fi
1      Adventure,Mystery,Sci-Fi
2               Horror,Thriller
3       Animation,Comedy,Family
4      Action,Adventure,Fantasy
                  ...
995         Crime,Drama,Mystery
996                      Horror
997         Drama,Music,Romance
998            Adventure,Comedy
999       Comedy,Family,Fantasy
Name: genre, Length: 1000, dtype: object
```

In [86]:
```
#multiple columns
movies_df[['title','genre']]
```

Out[86]:

	title	genre
0	Guardians of the Galaxy	Action,Adventure,Sci-Fi
1	Prometheus	Adventure,Mystery,Sci-Fi
2	Split	Horror,Thriller
3	Sing	Animation,Comedy,Family
4	Suicide Squad	Action,Adventure,Fantasy
...	...	...
995	Secret in Their Eyes	Crime,Drama,Mystery
996	Hostel: Part II	Horror
997	Step Up 2: The Streets	Drama,Music,Romance
998	Search Party	Adventure,Comedy
999	Nine Lives	Comedy,Family,Fantasy

1000 rows × 2 columns

getting data by rows:

In [89]:
```
movies_df.iloc[0]
# you  can use .loc to locate by name
```

Out[89]:
```
rank                                                                1
title                                         Guardians of the Galaxy
genre                                         Action,Adventure,Sci-Fi
description          A group of intergalactic criminals are forced ...
director                                                    James Gunn
actors               Chris Pratt, Vin Diesel, Bradley Cooper, Zoe S...
year                                                             2014
runtime (minutes)                                                 121
rating                                                            8.1
votes                                                          757074
revenue                                                        333.13
metascore                                                        76.0
Name: 0, dtype: object
```

Slicing:

In [90]:
```
movies_df.iloc[0:3]
```

Out[90]:

	rank	title	genre	description	director	actors	year	runtime (minutes)	rating	votes	revenue	metascore
0	1	Guardians of the Galaxy	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
1	2	Prometheus	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
2	3	Split	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

Conditional selections:

filter our movies DataFrame to show only films directed by Ridley Scott?

In [ ]:
```
movies_df[movies_df['director']=='Ridley Scott']
```

show movies with rating higher than 8.3?

In [ ]:
```
movies_df[movies_df['rating'] >=8.3]
```

filter the the DataFrame to show only movies by Christopher Nolan OR Ridley Scott?

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

We can make some richer conditionals by using logical operators | for "or" and & for "and".

Applying functions

It is possible to iterate over a DataFrame or Series as you would with a list, but doing so — especially on large datasets — is very slow.

An efficient alternative is to apply() a function to the dataset.

For example, we could use a function to convert movies with an 8.0 or greater to a string value of "good" and the rest to "bad" and use this transformed values to create a new column.

In [104]:
```
movies_df["rating_category"] = movies_df["rating"].apply(lambda x: 'good' if x >= 8.0 else 'bad')

movies_df.head(2)
```

Out[104]:

	rank	title	genre	description	director	actors	year	runtime (minutes)	rating	votes	revenue	metascore	rating_
0	1	Guardians of the Galaxy	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	
1	2	Prometheus	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	

Honorable mentions of useful methods

In [105]:
```
movies_df.dtypes
```

Out[105]:
```
rank                   int64
title                 object
genre                 object
description           object
director              object
actors                object
year                   int64
runtime (minutes)      int64
rating               float64
votes                  int64
revenue              float64
metascore            float64
rating_category       object
dtype: object
```

In [108]:
```
movies_df.sort_values(by="rating",ascending=False).head(3)
```

Out[108]:

	rank	title	genre	description	director	actors	year	runtime (minutes)	rating	votes	revenue	metascore
54	55	The Dark Knight	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
80	81	Inception	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
117	118	Dangal	Action,Biography,Drama	Former wrestler Mahavir Singh Phogat and his t...	Nitesh Tiwari	Aamir Khan, Sakshi Tanwar, Fatima Sana Shaikh,...	2016	161	8.8	48969	11.15	NaN

.concat(), .groupby() and many more useful methods