

Pandas Datastructures

The primary two components of pandas are the Series and DataFrame.

Dataframes and Series

Series	Series	DataFrame																																
<table border="1"> <thead> <tr><th>apples</th></tr> </thead> <tbody> <tr><td>0</td><td>3</td></tr> <tr><td>1</td><td>2</td></tr> <tr><td>2</td><td>0</td></tr> <tr><td>3</td><td>1</td></tr> </tbody> </table>	apples	0	3	1	2	2	0	3	1	<table border="1"> <thead> <tr><th>oranges</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td></tr> <tr><td>1</td><td>3</td></tr> <tr><td>2</td><td>7</td></tr> <tr><td>3</td><td>2</td></tr> </tbody> </table>	oranges	0	0	1	3	2	7	3	2	<table border="1"> <thead> <tr><th>apples</th><th>oranges</th></tr> </thead> <tbody> <tr><td>0</td><td>3</td><td>0</td></tr> <tr><td>1</td><td>2</td><td>3</td></tr> <tr><td>2</td><td>0</td><td>7</td></tr> <tr><td>3</td><td>1</td><td>2</td></tr> </tbody> </table>	apples	oranges	0	3	0	1	2	3	2	0	7	3	1	2
apples																																		
0	3																																	
1	2																																	
2	0																																	
3	1																																	
oranges																																		
0	0																																	
1	3																																	
2	7																																	
3	2																																	
apples	oranges																																	
0	3	0																																
1	2	3																																
2	0	7																																
3	1	2																																

A Series is One Dimensional Data
A DataFrame is N-Dimensional Data

Creating a DataFrame from Scratch

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

Turns a Dictionary into a DataFrame

Adding an Index to a DataFrame

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

A DataFrame is a (key, value) based Data Structure

Dataframe with Index Set

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

Understanding your Data

```
movies_df.describe()
count    1000.000000    1000.000000    1000.000000    1000.000000    1.000000e+03    1000.000000    956.000000
mean     500.000000    2020.780000    121.272000    6.722200    2.658828e+05    82.755376    58.958241
std      288.819436    1.205962    18.819908    0.645429    1.887628e+05    46.412643    17.194757
min      1.000000    2006.000000    66.000000    1.900000    6.158885e+02    0.000000    11.000000
25%     288.750000    2020.000000    100.000000    6.200000    8.438890e+04    17.645200    47.000000
50%     500.000000    2024.000000    111.000000    6.800000    1.187990e+05    46.375000    59.500000
75%     750.250000    2026.000000    123.000000    7.400000    2.598954e+05    99.177500    72.000000
max     1000.000000    2026.000000    191.000000    9.000000    1.771512e+06    926.630000    100.000000
```

```
movies_df.describe()
```

```
# Describing individual columns
movies_df['genre'].describe()
```

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

genre	count
Action,Adventure,Sci-Fi	50
Drama	48
Comedy,Drama,Romance	35
Comedy	32
Drama,Romance	31
Animation,Adventure,Comedy	27
Action,Adventure,Fantasy	27
Comedy,Drama	27

movies_df.corr()

```
rank      year      runtime      rating      votes      revenue_millions      metacore
count    1.000000    -0.261605    -0.221739    -0.219555    -0.203378    -0.252094    -0.191869
year     -0.261605    1.000000    -0.164900    -0.213119    -0.411954    -0.117582    -0.079390
runtime  -0.221739    -0.164900    1.000000    0.192214    -0.407021    -0.247324    -0.219798
rating   -0.219555    -0.213119    0.192214    1.000000    0.111537    -0.189527    -0.451897
votes    -0.203378    -0.411954    -0.407021    0.111537    1.000000    -0.467941    -0.325684
revenue_millions -0.252094    -0.117582    -0.247324    -0.189527    -0.467941    1.000000    -0.213328
metacore -0.191869    -0.079390    -0.219798    -0.451897    -0.325684    -0.213328    1.000000
```

Generates a matrix describing correlation between data

Reading in Data

It's quite simple to load data from various file formats into a DataFrame. In the following examples we'll keep using our apples and oranges data, but this time it's coming from various files.

Reading CSV

```
df =
pd.read_csv('purchases.csv')
# Or with setting the index
column, needs to be set with CSV
df = pd.read_csv('purchases.csv', index_col=0)
```

Reading JSON

```
# The index is automatically set
with json
df = pd.read_json('purchases.json')
```

Reading in SQLite

```
import sqlite3
con = sqlite3.connect("database.db")
df = pd.read_sql_query("SELECT * FROM purchases", con)
```

Setting index in post

```
df = df.set_index('index')
```

Writing Data

```
df.to_csv('new_purchases.csv')
df.to_json('new_purchases.json')
df.to_sql('new_purchases', con)
```

con being a sql connection

When we save JSON and CSV files, all we have to input into those functions is our desired filename with the appropriate file extension. With SQL, we're not creating a new file but instead inserting a new table into the database using our con variable from before.



By CodingJinxx

Published 2nd December, 2021.
Last updated 9th December, 2021.
Page 1 of 3.

Sponsored by [CrosswordCheats.com](https://crosswordcheats.com)
Learn to solve cryptic crosswords!
<http://crosswordcheats.com>

Dataframe Slicing Extracting Selecting

Up until now we've focused on some basic summaries of our data. We've learned about simple column extraction using single brackets, and we imputed null values in a column using fillna(). Below are the other methods of slicing, selecting, and extracting you'll need to use constantly.

It's important to note that, although many methods are the same, DataFrames and Series have different attributes, so you'll need be sure to know which type you are working with or else you will receive attribute errors.

Let's look at working with columns first.

By Columns

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

To extract data as Dataframes from a dataframe pass a list key

By Rows

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

loc and iloc can be thought of as similar to Python list slicing. To show this even further, let's select multiple rows.

Conditional Selections

```
condition =
(movies_df['director'] ==
"Ridley Scott")
movies_df[movies_df['director'] == "Ridley Scott"].head() # Returns a Dataframe where Director is Ridley Scott
movies_df[movies_df['rating'] >= 8.6].head(3)
movies_df[(movies_df['director'] == 'Christopher Nolan') | (movies_df['director'] == 'Ridley Scott')].head()
# OR Simply
movies_df[movies_df['director'].isin(['Christopher Nolan', 'Ridley Scott'])].head()
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))
]
```

Similar to isnull(), this returns a Series of True and False values: True for films directed by Ridley Scott and False for ones not directed by him.

Applying Functions to Data

```
def rating_function(x):
    if x >= 8.0:
        return "good"
    else:
        return "bad"
movies_df["rating_category"] = movies_df["rating"].apply(rating_function)
```

Viewing Data

```
movies_df.head() # Print first 5 rows
movies_df.head(10) # Print first 10 rows
movies_df.tail() # Print last 5 Rows
movies_df.tail(2) # Print last 2 rows
```

Previews start or ends of dataframes

Getting metadata from a dataframe

```
movies_df.info() # Output info regarding datatypes
movies_df.shape # Outputs number of rows and columns
```

Dataframe Info

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 500 entries, Guardians of the Galaxy to Nine Lives
Data columns (total 12 columns):
# Column Non-Null Count Dtype
---
0 rank 1000 non-null int64
1 genre 1000 non-null object
2 description 1000 non-null object
3 director 1000 non-null object
4 actors 1000 non-null object
5 year 1000 non-null int64
6 runtime 1000 non-null int64
7 rating 1000 non-null float64
8 votes 1000 non-null int64
9 revenue_millions 1000 non-null float64
10 metascore 955 non-null float64
11 rating_category 1000 non-null object
dtypes: float64(3), int64(4), object(5)
memory usage: 225.0v MB
```

df.info()



By CodingJinxx

cheatography.com/codingjinxx/

Published 2nd December, 2021.

Last updated 9th December, 2021.

Page 2 of 3.

Sponsored by [CrosswordCheats.com](http://crosswordcheats.com)

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

Cleaning Data

```
### Duplicate Data Handling
temp_df = temp_df.drop_duplicates()
# Or
temp_df.drop_duplicates(inplace=True)
# Keeping duplicate data
temp_df.drop_duplicates(inplace=True, keep=True)
### Renaming Columns
movies_df.rename(columns={'Runtime (Minutes)': 'Runtime',
                           'Revenue (Millions)': 'Revenue_millions'},
                  inplace=True)
# Can also be set directly
movies_df.columns = ['rank', 'genre', 'description', 'director', 'actors', 'year', 'runtime',
                     'rating', 'votes', 'revenue_millions', 'metascore']
# Setting column names lowercase
movies_df.columns = [col.lower() for col in movies_df]
### Handling Null Values
movies_df.isnull() # Returns a column with either True or False for null or not
movies_df.isnull().sum() # Returns a count of all null entries
# Removing Nulls
```

Cleaning Data (cont)

```
movies_df.dropna()
# Specify the axis to drop against
movies_df.dropna(axis=1)
```

Plotting with Matplotlib

Another great thing about pandas is that it integrates with Matplotlib, so you get the ability to plot directly off DataFrames and Series. To get started we need to import Matplotlib (pip install matplotlib):

Scatter plot

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

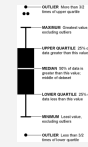
Histogram plot

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

Boxplot

```
movies_df['rating'].plot(kind="box")
```

Boxplot



By CodingJinxx

cheatography.com/codingjinxx/

Published 2nd December, 2021.

Last updated 9th December, 2021.

Page 3 of 3.

Sponsored by [CrosswordCheats.com](https://crosswordcheats.com)

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>