My Python Pandas Cheat Sheet

The pandas functions I use every day as a data scientist and software engineer

Chris I.
Mar 22 · 8 min read ★

```
anime.groupby(["type"]).agg({
  "rating": "sum",
  "episodes": "count",
  "name": "last"
}).reset_index()
```

	type	rating	episodes	name
0	Movie	14512.58	2348	Yasuji no Pornorama: Yacchimae!!
1	Music	2727.43	488	Yuu no Mahou
2	ONA	3679.43	659	Docchi mo Maid
3	OVA	20942.60	3311	Violence Gekiga Shin David no Hoshi: Inma Dens...
4	Special	10900.77	1676	Junjou Shoujo Et Cetera Specials
5	TV	25338.34	3787	Yuuki Yuuna wa Yuusha de Aru: Yuusha no Shou

A mentor once told me that software engineers are like indexes not textbooks; we don't memorize everything but we know how to look it up quickly.

Being able to look up and use functions fast allows us to achieve a certain flow when writing code. So I've created this cheatsheet of functions I use everyday building web apps and machine learning models.

This is not a comprehensive list but contains the functions I use most, an example, and my insights as to when it's most useful.

Contents:

1) Setup

If you want to run these examples yourself, download the Anime recommendationdataset from Kaggle, unzip and drop it in the same folder as your jupyter notebook.

Next Run these commands and you should be able to replicate my results for any of the below functions.

```
import pandas as pd
import numpy as np

anime = pd.read_csv('anime-recommendations-database/anime.csv')
rating = pd.read_csv('anime-recommendations-database/rating.csv')

anime_modified = anime.set_index('name')
```

2) Importing

Load CSV

Convert a CSV directly into a data frame. Sometimes loading data from a CSV also requires specifying an `encoding` (ie: `encoding='ISO-8859-1'`). It's the first thing you should try if your data frame contains unreadable characters.

Another similar function also exists called `pd.read_excel` for excel files.

```
anime = pd.read_csv('anime-recommendations-database/anime.csv')
```

	anime_id	name	genre	type	episodes	rating	members
0	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630

1	5114	Fullmetal Alchemist: Brotherhood	Action, Adventure, Drama, Fantasy, Magic, Mili...	TV	64	9.26	793665
2	28977	Gintama°	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.25	114262
3	9253	Steins;Gate	Sci-Fi, Thriller	TV	24	9.17	673572
4	9969	Gintama'	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.16	151266

Build data frame from inputted data

Useful when you want to manually instantiate simple data so that you can see how it changes as it flows through a pipeline.

```
df = pd.DataFrame([[1,'Bob', 'Builder'],
                   [2,'Sally', 'Baker'],
                   [3,'Scott', 'Candle Stick Maker']],
columns=['id','name', 'occupation'])
```

	id	name	occupation
0	1	Bob	Builder
1	2	Sally	Baker
2	3	Scott	Candle Stick Maker

df.head()

Copy a data frame

Useful when you want to make changes to a data frame while maintaining a copy of the original. It's good practise to `copy` all data frames immediately after loading them.

```
anime_copy = anime.copy(deep=True)
```

	anime_id	name	genre	type	episodes	rating	members
0	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630
1	5114	Fullmetal Alchemist: Brotherhood	Action, Adventure, Drama, Fantasy, Magic, Mili...	TV	64	9.26	793665
2	28977	Gintama°	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.25	114262
3	9253	Steins;Gate	Sci-Fi, Thriller	TV	24	9.17	673572
4	9969	Gintama'	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.16	151266

3) Exporting

Save to CSV

This dumps to the same directory as the notebook. I'm only saving the 1st 10 rows below but you don't need to do that. Again, `df.to_excel()` also exists and functions basically the same for excel files.

```
rating[:10].to_csv('saved_ratings.csv', index=False)
```

4) Viewing and Inspecting

Get top or bottom `n` records

Display the first `n` records from a data frame. I often print the top record of a data frame somewhere in my notebook so I can refer back to it if I forget what's inside.

```
anime.head(3)
rating.tail(1)
```

	anime_id	name	genre	type	episodes	rating	members
0	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630
1	5114	Fullmetal Alchemist: Brotherhood	Action, Adventure, Drama, Fantasy, Magic, Mili...	TV	64	9.26	793665
2	28977	Gintama°	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.25	114262

	anime_id	name	genre	type	episodes	rating	members
12293	26081	Yasuji no Pornorama: Yacchimae!!	Hentai	Movie	1	5.46	142

Count rows

This is not a pandas function per se but `len()` counts rows and can be saved to a variable and used elsewhere.

```
len(df)
#=> 3
```

Count unique rows

Count unique values in a column.

```
len(ratings['user_id'].unique())
```

Get data frame info

Useful for getting some general information like header, number of values and datatype by column. A similar but less useful function is `df.dtypes` which just gives column data types.

```
anime.info()
```

```
RangeIndex: 12294 entries, 0 to 12293
Data columns (total 7 columns):
anime_id    12294 non-null int64
name        12294 non-null object
genre       12232 non-null object
type        12269 non-null object
episodes    12294 non-null object
rating      12064 non-null float64
members     12294 non-null int64
dtypes: float64(1), int64(2), object(4)
memory usage: 672.4+ KB
```

Get statistics

Really useful if the data frame has a lot of numeric values. Knowing the mean, min and max of the rating column give us a sense of how the data frame looks overall.

```
anime.describe()
```

	anime_id	rating	members
count	12294.000000	12064.000000	1.229400e+04
mean	14058.221653	6.473902	1.807134e+04
std	11455.294701	1.026746	5.482068e+04
min	1.000000	1.670000	5.000000e+00
25%	3484.250000	5.880000	2.250000e+02
50%	10260.500000	6.570000	1.550000e+03
75%	24794.500000	7.180000	9.437000e+03
max	34527.000000	10.000000	1.013917e+06

Get counts of values

Get counts of values for a particular column.

```
anime.type.value_counts()
```

```
TV          3787
```

```
OVA        3311
Movie      2348
Special    1676
ONA         659
Music       488
Name: type, dtype: int64
```

5) Selecting

Get a list or series of values for a column

This works if you need to pull the values in columns into `X` and `y` variables so you can fit a machine learning model.

```
anime['genre'].tolist()
anime['genre']
```

```
['Drama, Romance, School, Supernatural',
 'Action, Adventure, Drama, Fantasy, Magic, Military, Shounen',
 'Action, Comedy, Historical, Parody, Samurai, Sci-Fi, Shounen',
 'Sci-Fi, Thriller',
 'Action, Comedy, Historical, Parody, Samurai, Sci-Fi, Shounen',
```

anime['genre'].tolist()

```
0                  Drama, Romance, School, Supernatural
1       Action, Adventure, Drama, Fantasy, Magic, Mili...
2       Action, Comedy, Historical, Parody, Samurai, S...
3                                   Sci-Fi, Thriller
```

anime['genre']

Get a list of index values

Create a list of values from index. Note I've used `anime_modified` data frame here as the index values are more interesting.

```
anime_modified.index.tolist()
```

```
['Kimi no Na wa.',
 'Fullmetal Alchemist: Brotherhood',
 'GintamaÂ°',
 'Steins;Gate',
```

Get a list of column values

```
anime.columns.tolist()
```

```
['anime_id', 'name', 'genre', 'type', 'episodes', 'rating', 'members']
```

6) Adding / Dropping

Append new column with a set value

I do this on occasion when I have test and train sets in 2 separate data frames and want to mark which rows are related to what set before combining them.

```
anime['train set'] = True
```

Create new data frame from a subset of columns

Useful when you only want to keep a few columns from a giant data frame and don't want to specify each that you want to drop.

```
anime[['name','episodes']]
```

	name	episodes
0	Kimi no Na wa.	1
1	Fullmetal Alchemist: Brotherhood	64
2	Gintama°	51
3	Steins;Gate	24
4	Gintama'	51

Drop specified columns

Useful when you only need to drop a few columns. Otherwise, it can be tedious to write them all out and I prefer the previous option.

```
anime.drop(['anime_id', 'genre', 'members'], axis=1).head()
```

	name	type	episodes	rating
0	Kimi no Na wa.	Movie	1	9.37
1	Fullmetal Alchemist: Brotherhood	TV	64	9.26
2	GintamaÂ°	TV	51	9.25
3	Steins;Gate	TV	24	9.17

4	Gintama'	TV	51	9.16

Add a row with sum of other rows

We'll manually create a small data frame here because it's easier to look at. The interesting part here is `df.sum(axis=0)` which adds the values across rows. Alternatively `df.sum(axis=1)` adds values across columns.

The same logic applies when calculating counts or means, ie: `df.mean(axis=0)` .

```
df = pd.DataFrame([[1,'Bob', 8000],
                   [2,'Sally', 9000],
                   [3,'Scott', 20]], columns=['id','name', 'power
level'])

df.append(df.sum(axis=0), ignore_index=True)
```

	id	name	power level
0	1	Bob	8000
1	2	Sally	9000
2	3	Scott	20
3	6	BobSallyScott	17020

7) Combining

Concatenate 2 dataframes

Use this if you have 2 data frames with the same columns and want to combine them.

Here we split a data frame in 2 them add them back together.

```
df1 = anime[0:2]

df2 = anime[2:4]

pd.concat([df1, df2], ignore_index=True)
```

	anime_id	name	genre	type	episodes	rating	members
0	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630
1	5114	Fullmetal Alchemist: Brotherhood	Action, Adventure, Drama, Fantasy, Magic, Mili...	TV	64	9.26	793665

	anime_id	name	genre	type	episodes	rating	members
2	28977	Gintama°	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.25	114262
3	9253	Steins;Gate	Sci-Fi, Thriller	TV	24	9.17	673572

	anime_id	name	genre	type	episodes	rating	members
0	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630
1	5114	Fullmetal Alchemist: Brotherhood	Action, Adventure, Drama, Fantasy, Magic, Mili...	TV	64	9.26	793665
2	28977	Gintama°	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.25	114262
3	9253	Steins;Gate	Sci-Fi, Thriller	TV	24	9.17	673572

Merge dataframes

This functions like a SQL left join, when you have 2 data frames and want to join on a column.

```
rating.merge(anime, left_on='anime_id', right_on='anime_id', suffixes=
('_left', '_right'))
```

	user_id	anime_id	rating_left	name	genre	type	episodes	rating_right	members
0	1	20	-1	Naruto	Action, Comedy, Martial Arts, Shounen, Super P...	TV	220	7.81	683297
1	3	20	8	Naruto	Action, Comedy, Martial Arts, Shounen, Super P...	TV	220	7.81	683297
2	5	20	6	Naruto	Action, Comedy, Martial Arts, Shounen, Super P...	TV	220	7.81	683297
3	6	20	-1	Naruto	Action, Comedy, Martial Arts, Shounen, Super P...	TV	220	7.81	683297
4	10	20	-1	Naruto	Action, Comedy, Martial Arts, Shounen, Super P...	TV	220	7.81	683297

8) Filtering

Retrieve rows with matching index values

The index values in `anime_modified` are the names of the anime. Notice how we've used those names to grab specific columns.

```
anime_modified.loc[['Haikyuu!! Second Season','Gintama']]
```

name	anime_id	genre	type	episodes	rating	members
Haikyuu!! Second Season	28891	Comedy, Drama, School, Shounen, Sports	TV	25	8.93	179342
Gintama	918	Action, Comedy, Historical, Parody, Samurai, S...	TV	201	9.04	336376

Retrieve rows by numbered index values

This differs from the previous function. Using `iloc`, the 1st row has an index of `0`, the 2nd row has an index of `1`, and so on... even if you've modified the data frame and are

now using string values in the index column.

Use this is you want the first 3 rows in a data frame.

```
anime_modified.iloc[0:3]
```

name	anime_id	genre	type	episodes	rating	members
Kimi no Na wa.	32281	Drama, Romance, School, Supernatural	Movie	1	9.37	200630
Fullmetal Alchemist: Brotherhood	5114	Action, Adventure, Drama, Fantasy, Magic, Mili...	TV	64	9.26	793665
GintamaÃ°	28977	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.25	114262

Get rows

Retrieve rows where a column's value is in a given list. `anime[anime['type'] == 'TV']` also works when matching on a single value.

```
anime[anime['type'].isin(['TV', 'Movie'])]
```

	anime_id	name	genre	type	episodes	rating	members
0	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630
1	5114	Fullmetal Alchemist: Brotherhood	Action, Adventure, Drama, Fantasy, Magic, Mili...	TV	64	9.26	793665
2	28977	GintamaÃ°	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.25	114262
3	9253	Steins;Gate	Sci-Fi, Thriller	TV	24	9.17	673572
4	9969	Gintama'	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.16	151266

Slice a dataframe

This is just like slicing a list. Slice a data frame to get all rows before/between/after specified indices.

```
anime[1:3]
```

	anime_id	name	genre	type	episodes	rating	members
1	5114	Fullmetal Alchemist: Brotherhood	Action, Adventure, Drama, Fantasy, Magic, Mili...	TV	64	9.26	793665
2	28977	GintamaÃ°	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.25	114262

Filter by value

Filter data frame for rows that meet a condition. Note this maintains existing index values.

```
anime[anime['rating'] > 8]
```

	anime_id	name	genre	type	episodes	rating	members
0	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630
1	5114	Fullmetal Alchemist: Brotherhood	Action, Adventure, Drama, Fantasy, Magic, Mili...	TV	64	9.26	793665
2	28977	GintamaÅ°	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.25	114262
3	9253	Steins;Gate	Sci-Fi, Thriller	TV	24	9.17	673572
4	9969	Gintama'	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.16	151266

9) Sorting

sort_values

Sort data frame by values in a column.

```
anime.sort_values('rating', ascending=False)
```

	anime_id	name	genre	type	episodes	rating	members
10464	33662	Taka no Tsume 8: Yoshida-kun no X-Files	Comedy, Parody	Movie	1	10.00	13
10400	30120	Spoon-hime no Swing Kitchen	Adventure, Kids	TV	Unknown	9.60	47
9595	23005	Mogura no Motoro	Slice of Life	Movie	1	9.50	62
0	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630
9078	33607	Kahei no Umi	Historical	Movie	1	9.33	44

10) Aggregating

Groupby and count

Count number of records for each distinct value in a column.

```
anime.groupby('type').count()
```

type	anime_id	name	genre	episodes	rating	members
Movie	2348	2348	2306	2348	2297	2348
Music	488	488	488	488	488	488
ONA	659	659	655	659	652	659
OVA	3311	3311	3310	3311	3285	3311
Special	1676	1676	1674	1676	1671	1676

TV	3787	3787	3777	3787	3671	3787

Groupby and aggregate columns in different ways

Note I added `reset_index()` otherwise the `type` column becomes the index column — I recommend doing the same in most cases.

```
anime.groupby(["type"]).agg({
  "rating": "sum",
  "episodes": "count",
  "name": "last"
}).reset_index()
```

Create a pivot table

Nothing better than a pivot table for pulling a subset of data from a data frame.

Note I've heavily filtered the data frame so it's quicker to build the pivot table.

```
tmp_df = rating.copy()
tmp_df.sort_values('user_id', ascending=True, inplace=True)
tmp_df = tmp_df[tmp_df.user_id < 10]
tmp_df = tmp_df[tmp_df.anime_id < 30]
tmp_df = tmp_df[tmp_df.rating != -1]

pd.pivot_table(tmp_df, values='rating', index=['user_id'], columns=
['anime_id'], aggfunc=np.sum, fill_value=0)
```

anime_id	6	15	17	18	20	22	24
user_id							
3	0	0	0	0	8	0	0
5	8	6	6	6	6	5	1
7	0	0	0	0	0	7	0

11) Cleaning

Set NaN cells to some value

Set cells with `NaN` value to `0` . In the example we create the same pivot table as before but without `fill_value=0` then use `fillna(0)` to fill them in afterwards.

```
pivot = pd.pivot_table(tmp_df, values='rating', index=['user_id'],
columns=['anime_id'], aggfunc=np.sum)

pivot.fillna(0)
```

anime_id	6	15	17	18	20	22	24
user_id							
3	NaN	NaN	NaN	NaN	8.0	NaN	NaN
5	8.0	6.0	6.0	6.0	6.0	5.0	1.0
7	NaN	NaN	NaN	NaN	NaN	7.0	NaN

anime_id	6	15	17	18	20	22	24
user_id							
3	0.0	0.0	0.0	0.0	8.0	0.0	0.0
5	8.0	6.0	6.0	6.0	6.0	5.0	1.0
7	0.0	0.0	0.0	0.0	0.0	7.0	0.0

12) Other

Sample a data frame

I use this all the time taking a small sample from a larger data frame. It allows randomly rearranging rows while maintaining indices if `frac=1`.

```
anime.sample(frac=0.25)
```

	anime_id	name	genre	type	episodes	rating	members
3174	673	Mai-HiME Specials	Comedy	Special	27	7.06	9210
11613	3998	Tokumu Sousakan Rei & Fuko	Hentai, Horror, Supernatural	OVA	4	6.37	1591
10229	17315	Shin Pro Golfer Saru	Kids, Sports	TV	10	7.00	69
53	6114	Rainbow: Nisha Rokubou no Shichinin	Drama, Historical, Seinen, Thriller	TV	26	8.64	139474
7732	6837	Tengu Taiji	Comedy	Movie	1	5.09	501

Iterate over row indices

Iterate over index and rows in data frame.

```
for idx,row in anime[:2].iterrows():
    print(idx, row)
```

```
0 anime_id                                    32281
name                                  Kimi no Na wa.
genre       Drama, Romance, School, Supernatural
type                                          Movie
episodes                                          1
rating                                         9.37
members                                      200630
Name: 0, dtype: object
1 anime_id                                                 5114
name                              Fullmetal Alchemist: Brotherhood
genre       Action, Adventure, Drama, Fantasy, Magic, Mili...
type                                                       TV
episodes                                                   64
rating                                                   9.26
members                                                793665
Name: 1, dtype: object
```

Starting jupyter notebook

Start notebook with a very high data rate limit.

```
jupyter notebook — NotebookApp.iopub_data_rate_limit=1.0e10
```

13) Conclusion

I hope this can be a reference guide for you as well. I'll try to continuously update this as I find more useful pandas functions.

If there are any functions you can't live without please post them in the comments below!