My Python Pandas Cheat Sheet

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

Chris I. Mar 23, 2020 · 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 recommendation dataset 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')
```


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'])
```


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)
```


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)
```

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()
```


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()
```


Get counts of values

Get counts of values for a particular column.

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


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']
```


anime['genre'].tolist()

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()
```

Get a list of column values

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

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']]
```


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()
```


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)
```

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)
```


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'))
```

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']]
```


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]
```


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'])]
```


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]
```


Filter by value

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

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

9) Sorting

sort_values

Sort data frame by values in a column.

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


10) Aggregating

Groupby and count

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

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


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)
```


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)
```

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)
```


Iterate over row indices

Iterate over index and rows in data frame.

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

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!

Pandas | Data Science | Python | Coding | Programming


Download on the
App Store
GET IT ON
Google Play