

 Member-only story

35 Pandas Tricks to Save in Your List

Shantun Parmar · [Follow](#)

Published in Python in Plain English · 5 min read · Nov 26, 2024





Shantun Parmar

Pandas is one of the powerful dataset processing library not matter if you are a beginner in data science or an expert of python, you should use pandas to tidy, process and examine your dataset. It's a tool that's very versatile, but you can miss some of the more or **less-known tricks** that could save you time and make your code better.

In this article, I'll go over 35 Pandas tricks you should save in your list. This will make you a more efficient data scientist and these tips will make you more productive, optimize your workflow.

1. Quickly Load Data with `read_csv`

Loaded datasets into DataFrames using an easy to use method called `read_csv()` in Pandas. In addition you can pass more arguments such as `usecols` to read only certain columns, saving both time and memory.

```
import pandas as pd
df = pd.read_csv('data.csv', usecols=['Column1', 'Column2'])
```

2. View the First Few Rows with `head()`

The `head()` method quickly preview a dataset. They by default show you the first 5 rows.

```
df.head()
```

3. Check for Missing Values with `isna()`

`isna()` returns a DataFrame of booleans, indicating missing data before cleaning your data.

```
df.isna().sum()
```

4. Remove Duplicates with `drop_duplicates()`

`drop_duplicates()` removes duplicate rows from your dataset if your dataset has duplicate rows. This is especially helpful when you clean data to analyze.

```
df.drop_duplicates(inplace=True)
```

5. Rename Columns with `rename()`

Using the `rename()` method you can easily rename columns in case they have unclear names.

```
df.rename(columns={'old_name': 'new_name'}, inplace=True)
```

6. Convert Data Types with `astype()`

Your columns might sometimes be in the wrong data type. Use `astype()` to convert them.

```
df['Column1'] = df['Column1'].astype(float)
```

7. Handle Missing Data with `fillna()`

Rather than dropping rows with missing data, you can fill such rows with a specified value or method such as forward-fill or backward-fill.

```
df.fillna(method='ffill', inplace=True)
```

8. Set a Column as the Index

If you want your DataFrame to have an index, you can set a column to be index of your dataframe.

```
df.set_index('Column1', inplace=True)
```

9. Use `groupby()` for Aggregation

Aggregate group data based on a column you provide, allowing you to sum, mean, or count.

```
df.groupby('Category').agg({'Value': 'sum'})
```

10. Sort Values with `sort_values()`

To sort your DataFrame by a column `sort_values()`. You can order it ascending or descending.

```
df.sort_values(by='Column1', ascending=False)
```

11. Use `apply()` for Custom Functions

We use the `apply()` function for applying your custom function to DataFrame column or to rows.

```
df['new_column'] = df['Column1'].apply(lambda x: x * 2)
```

12. Filter Rows Based on Conditions

To filter rows of your DataFrame using conditional statements, use conditional statements.

```
df_filtered = df[df['Column1'] > 10]
```

13. Use `query()` for More Complex Filters

You can use `query()` , for better readable filtering.

```
df.query('Column1 > 10 and Column2 == "Yes"')
```

14. Concatenate DataFrames with `concat()`

So if you have multiple DataFrames, you can concatenate them on one axis using `concat()`.


```
df_combined = pd.concat([df1, df2], axis=0)
```

15. Merge DataFrames with `merge()`

Merging is like a SQL JOIN. When combining datasets based on common columns, it's incredibly useful.

```
df_merged = pd.merge(df1, df2, on='ID')
```

16. Reset Index with `reset_index()`

If you modify the index, or drop rows, use `reset_index()` to reset the index.

```
df.reset_index(drop=True, inplace=True)
```

17. Export Data to Excel with `to_excel()`

`to_excel()` is what you need to save your DataFrame to an Excel file.

```
df.to_excel('output.xlsx', index=False)
```

18. Convert to CSV with `to_csv()`

You can also export your DataFrame into a CSV file similarly.

```
df.to_csv('output.csv', index=False)
```

19. Quickly Get Descriptive Statistics with `describe()`

To quickly get an overview of the statistical summary of your DataFrame, use `describe()` .

```
df.describe()
```

20. Lambda Functions on Columns

`apply()` can be used, to apply a lambda function to multiple columns or rows.

```
df['new_column'] = df.apply(lambda row: row['Column1'] + row['Column2'], axis=1)
```

21. Drop Unwanted Columns with `drop()`

Remove unnecessary columns with `drop()` . You are able to drop one or more columns at once.

```
df.drop(columns=['Column1', 'Column2'], inplace=True)
```

22. Random sampling — use `sample()`

The `sample()` method in your DataFrame allows you to get a random sample from your dataframe.

```
df_sample = df.sample(n=5)
```

23. Extract Year, Month, or Day from DateTime Columns

If you are working with DateTime data you can extract the individual components such as year, month or day.

```
df['year'] = df['DateColumn'].dt.year  
df['month'] = df['DateColumn'].dt.month
```

24. Convert Column to Categorical Type

If you don't need to do much, storing columns with few unique values can be saved using categorical type and could be fast as well.

```
df['Category'] = df['Category'].astype('category')
```

25. Find Duplicates Across Specific Contacts

`df.duplicated()` helps you check for duplicates in columns.

```
df[df.duplicated(subset=['Column1', 'Column2'])]
```

26. Use `shift()` for Lagging Data

On time series or sequential data `shift()` can create lagged columns.

```
df['lagged_column'] = df['Value'].shift(1)
```

27. Pivot Data with `pivot_table()`

With Create Pivot Tables, you can summarize data in a pretty similar way to what you'd do in a spreadsheet (Excel).

```
df_pivot = df.pivot_table(values='Value', index='Category', columns='Region', ag
```



28. Plot with Pandas' Built-In Plotting

You can also plot quick plots with pandas integrated with Matplotlib.

```
df['Value'].plot(kind='line')
```

29. Working with Strings on the Columns

String data can be used for applying string methods to columns directly.

```
df['name'] = df['name'].str.lower()
```

30. Make a New Column Using a Condition

With `np.where()` you can create a new column based on conditional logic.

```
import numpy as np
df['NewColumn'] = np.where(df['Value'] > 10, 'High', 'Low')
```

31. Using `nunique()` for Unique Values Count

Check how many unique values are in a column fast.

```
df['Category'].nunique()
```

32. Get Column Data Types with `dtypes`

Use `dtypes` attribute to quickly check what you have in your columns.

```
df.dtypes
```

33. Find Correlations with `corr()`

To correlate numerical columns you use `corr()` .

```
df.corr()
```

34. Change Display Options

Pandas allows us to customize display of DataFrames, such as changing the max number of rows and columns shown.

```
pd.set_option('display.max_rows', 100)
```

35. Efficient Row-wise Operations

When you need to do something on a row by row basis then use `iterrows()`, otherwise go for vectorized solutions for better performance.

```
for idx, row in df.iterrows():  
    print(row['Column1'])
```

Conclusion

Pandas is just plain powerful and these 35 tricks will help you doing things faster, can help make your code more readable and increase your productivity. Remember these tips and when you sit down to work with data again, you'll be more prepared to get through it more quickly.

Which of these tricks do you think is the most useful? I'd love to hear from you! So, in the comment section below, drop in any other tricks, or Pandas tips that you love and want to see in this list.

Happy coding!

Image created using Canva.

In Plain English 

Thank you for being a part of the **In Plain English** community! Before you go:

- Be sure to **clap** and **follow** the writer 🙌
- Follow us: [X](#) | [LinkedIn](#) | [YouTube](#) | [Discord](#) | [Newsletter](#) | [Podcast](#)
- [Create a free AI-powered blog on Differ.](#)
- More content at [PlainEnglish.io](#)

Python

Pandas

Data Science

Libraries

Programming



Published in Python in Plain English

44K Followers · Last published 23 hours ago

New Python content every day. Follow to join our 3.5M+ monthly readers.

Follow



Written by Shantun Parmar

3.1K Followers · 22K Following

Software Engineer and Full Stack Developer specializing in Python & JavaScript |

Link : <https://www.linkedin.com/in/shantun-parmar/>

Follow

Responses (1)



Larrimer Prestosa

What are your thoughts?



Muhammad Naveed Arshad, MSc | Writer | Editor | AI Engr him

Nov 26, 2024



Which of these tricks do you think is the most useful?

Great insightful tricks for m adding a new column using condition is cool.



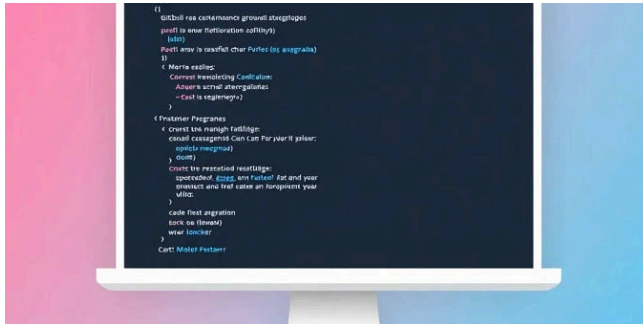
4



1 reply

[Reply](#)

More from Shantun Parmar and Python in Plain English



JS In JavaScript in Plain English by Shantun Parmar

Top 30 Coding Tools Every Developer Should Know About

As a software software engineer, I found several tools that simplify and improve fun o...

★ Oct 23, 2024 🖱 1.1K 💬 25 📌 ⋮




PY In Python in Plain English by Dhruv Ahuja

How I Learned to Love `__init__.py`: A Simple Guide 😊

💡 Heads Up! Click here to unlock this article for free if you're not a Medium member!

★ Feb 3 🖱 1.1K 💬 8 📌 ⋮

```
4- class B(A):
5-     def show(self):
6-         print("B")
7-         super().show()
8- class C(A):
9-     def show(self):
10-        print("C")
11-        super().show()
12- class D(B, C): # Multiple inheritance
13-     def show(self):
```

 In Python in Plain English by Kiran Maan

I Just Discovered Python's super() Works Differently Than I Thought

I always thought that I knew how super() worked in Python. But one incident broke thi...

★ Mar 9 🖱 363 💬 11  ⋮



Top 30 Coding Tools Every Developer Should Know About

★ · 6 min read · Oct 24, 2024

👤 899 💬 21

Lifetime

Oct 23, 2024 – Today · Updated every 24 hours

\$84.07

Earnings

7.7K


Views

4K

Reads

53%

Read ratio ⓘ

 In Readers Club by Shantun Parmar

4K Reads Later: What I'd Change in My Article

Hey everyone! I wrote “Top 30 Coding Tools Every Developer Should Know About” few...

★ Jan 26 🖱 647 💬 17  ⋮

See all from Shantun Parmar

See all from Python in Plain English

Recommended from Medium

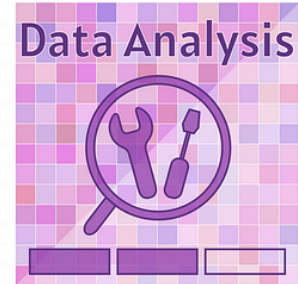



 In DataDrivenInvestor by Angelica Lo Duca 

Introducing PyNarrative: A Python Library for Data Storytelling

An overview of the new Python library for data storytelling.

★ Mar 9 🖱 281 💬 10  ⋮




 In Level Up Coding by Pawel Jastrzebski

Must-Know Python Data Analysis Tools to Learn in 2025

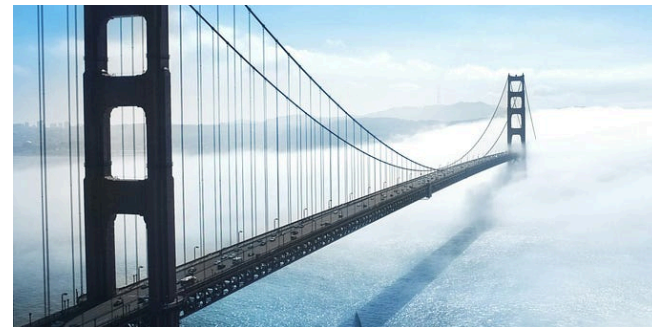
This is a second article in a 3 part series.

★ Mar 11 🖱 48 💬 1  ⋮



 In Towards AI by Thuwarakesh Murallie

Try This to Keep Your Python Code Clean Forever



 Kyle Jones

LLMs for Time Series Forecasting

Time series analysis is used for finance, healthcare, industrial IoT, and many other...

6d ago 142 2



Discover how FinGPT is disrupting traditional financial tools like Bloomberg Terminal,...

★ Feb 16 🖐️ 325 💬 6 📖+ ⋮

 In Tierra Insights by Stephen Chege

Unlocking the Power of Geospatial Data: The Best Python & R Tools for Mapping the World.

★ Mar 10 🖐️ 100

See more recommendations

