

[HOME](#) > [PROGRAMMING](#)

30 pandas Commands for Manipulating DataFrames

Read and write data to Excel sheets, modify DataFrames in one line of code, remove all rows containing null values... you can do it all with pandas.

BY IDOWU OMISOLA

PUBLISHED 3 DAYS AGO



The pandas library makes python-based data science an easy ride. It's a popular Python library for reading, merging, sorting, cleaning data, and more. Although pandas is easy to use and apply on datasets, it has many data manipulatory functions to learn.

You might use pandas, but there's a good chance you're under-utilizing it to solve data-related problems. Here's our list of valuable data manipulating pandas functions every data scientist should know.

Install pandas Into Your Virtual Environment

Before we proceed, make sure you install pandas into your virtual environment using pip:

```
pip install pandas
```

After installing it, import **pandas** at the top of your script, and let's proceed.

1. pandas.DataFrame

You use **pandas.DataFrame()** to create a DataFrame in pandas. There are two ways to use this function.

You can form a DataFrame column-wise by passing a dictionary into the **pandas.DataFrame()** function. Here, each key is a column, while the values are the rows:

```
import pandas

DataFrame = pandas.DataFrame({"A" : [1, 3, 4], "B": [5, 9, 12]})

print(DataFrame)
```

The other method is to form the DataFrame across rows. But here, you'll separate the values (row items) from the columns. The number of data in each list (row data) must also tally with the number of columns.

```
import pandas

DataFrame = pandas.DataFrame([[1, 4, 5], [7, 19, 13]], columns= ["J", "K", "L"])

print(DataFrame)
```

2. Read From and Write to Excel or CSV in pandas

You can read or write to Excel or CSV files with pandas.

Reading Excel or CSV files

To read an Excel file:

```
#Replace example.xlsx with the your Excel file path

DataFrame = DataFrame.read_excel("example.xlsx")
```

Here's how to read a CSV file:

```
#Replace example.csv with the your CSV file path

DataFrame = DataFrame.read_csv("example.csv")
```

Writing to Excel or CSV

Writing to Excel or CSV is a well-known pandas operation. And it's handy for saving newly computed tables into separate datasheets.

To write to an Excel sheet:

```
DataFrame.to_excel("full_path_of_the_destination_folder/filename.xlsx")
```

If you want to write to CSV:

```
DataFrame.to_csv("full_path_of_the_destination_folder/filename.csv")
```

3. Get the Mean, Median, and Mode

You can also compute the central tendencies of each column in a DataFrame using pandas.

Here's how to get the mean value of each column:

```
DataFrame.mean()
```

For the median or mode value, replace **mean()** with **median()** or **mode()**.

4. DataFrame.transform

pandas' **DataFrame.transform()** modifies the values of a DataFrame. It accepts a function as an argument.

For instance, the code below multiplies each value in a DataFrame by three using **Python's lambda function**:

```
DataFrame = DataFrame.transform(lambda y: y*3)
print(DataFrame)
```

5. DataFrame.isnull

This function returns a Boolean value and flags all rows containing null values as **True**:

```
DataFrame.isnull()
```

The result of the above code can be hard to read for larger datasets. So you can use the **isnull().sum()** function instead. This returns a summary of all missing values for each column:

```
DataFrame.isnull().sum()
```

6. Dataframe.info

The **info()** function is an **essential pandas operation**. It returns the summary of non-missing values for each column instead:

```
DataFrame.info()
```

7. DataFrame.describe

The **describe()** function gives you the summary statistic of a DataFrame:

```
DataFrame.describe()
```

8. DataFrame.replace

Using the **DataFrame.replace()** method in pandas, you can replace selected rows with other values.

For example, to swap invalid rows with **Nan**:

```
# Ensure that you pip install numpy for this to work

import numpy

import pandas

# Adding an inplace keyword and setting it to True makes the changes permanent:

DataFrame.replace([invalid_1, invalid_2], numpy.nan, inplace=True)

print(DataFrame)
```

9. DataFrame.fillna

This function lets you fill empty rows with a particular value. You can fill all **Nan** rows in a dataset with the mean value, for instance:

```
DataFrame.fillna(df.mean(), inplace = True)

print(DataFrame)
```

You can also be column-specific:

```
DataFrame['column_name'].fillna(df[column_name].mean(), inplace = True)

print(DataFrame)
```

10. DataFrame.dropna

The **dropna()** method removes all rows containing null values:

```
DataFrame.dropna(inplace = True)

print(DataFrame)
```

11. DataFrame.insert

You can use pandas' **insert()** function to add a new column to a DataFrame. It accepts three

keywords, the **column name**, a list of its data, and its **location**, which is a column index.

Here's how that works:

```
DataFrame.insert(column = 'C', value = [3, 4, 6, 7], loc=0)

print(DataFrame)
```

The above code inserts the new column at the zero column index (it becomes the first column).

12. DataFrame.loc

You can use **loc** to find the elements in a particular index. To view all items in the third row, for instance:

```
DataFrame.loc[2]
```

13. DataFrame.pop

This function lets you remove a specified column from a pandas DataFrame.

It accepts an **item** keyword, returns the popped column, and separates it from the rest of the DataFrame:

```
DataFrame.pop(item= 'column_name')
```



```
print(DataFrame)
```

14. DataFrame.max, min

Getting the maximum and minimum values using pandas is easy:

```
DataFrame.min()
```

The above code returns the minimum value for each column. To get the maximum, replace **min** with **max**.

15. DataFrame.join

The **join()** function of pandas lets you merge DataFrames with different column names. You can use the left, right, inner, or outer join. To left-join a DataFrame with two others:

```
#Left-join longer columns with shorter ones  
newDataFrame = df1.join([df_shorter2, df_shorter3], how='left')  
print(newDataFrame)
```

To join DataFrames with similar column names, you can differentiate them by including a suffix to the left or right. Do this by including the **lsuffix** or **rsuffix** keyword:

```
newDataFrame = df1.join([df2, rsuffix='_', how='outer'])
```

```
print(newDataFrame)
```

16. DataFrame.combine

The **combine()** function comes in handy for merging two DataFrames containing similar column names based on set criteria. It accepts a **function** keyword.

For instance, to merge two DataFrames with similar column names based on the maximum values only:

```
newDataFrame = df.combine(df2, numpy.minimum)
print(newDataFrame)
```

Note: You can also define a custom selection function and insert **numpy.minimum**.

17. DataFrame.astype

The **astype()** function changes the data type of a particular column or DataFrame.

To change all values in a DataFrame to string, for instance:

```
DataFrame.astype(str)
```

18. DataFrame.sum

The **sum()** function in pandas returns the sum of the values in each column:

```
DataFrame.sum()
```

You can also find the cumulative sum of all items using **cumsum()**:

```
DataFrame.cumsum()
```

19. DataFrame.drop

pandas' **drop()** function deletes specific rows or columns in a DataFrame. You have to supply the column names or row index and an axis to use it.

To remove specific columns, for example:

```
df.drop(columns=['column1', 'column2'], axis=0)
```

To drop rows on indexes 1, 3, and 4, for instance:

```
df.drop([1, 3, 4], axis=0)
```

20. DataFrame.corr

Want to find the correlation between integer or float columns? pandas can help you achieve that using the **corr()** function:

```
DataFrame.corr()
```

The above code returns a new DataFrame containing the correlation sequence between all integer or float columns.

21. DataFrame.add

The **add()** function lets you add a specific number to each value in DataFrame. It works by iterating through a DataFrame and operating on each item.

RELATED:

How To Use For Loops In Python

To add 20 to each of the values in a specific column containing integers or floats, for instance:

```
DataFrame['interger_column'].add(20)
```

22. DataFrame.sub

Like the addition function, you can also subtract a number from each value in a DataFrame or specific column:

```
DataFrame['interger_column'].sub(10)
```

23. DataFrame.mul

This is a multiplication version of the addition function of pandas:

```
DataFrame['interger_column'].mul(20)
```

24. DataFrame.div

Similarly, you can divide each data point in a column or DataFrame by a specific number:

```
DataFrame['interger_column'].div(20)
```

25. DataFrame.std

Using the **std()** function, pandas also lets you compute the standard deviation for each column in a DataFrame. It works by iterating through each column in a dataset and calculating the standard deviation for each:

```
DataFrame.std()
```

26. DataFrame.sort_values

You can also sort values ascendingly or descendingly based on a particular column. To sort a DataFrame in descending order, for example:

```
newDataFrame = DataFrame.sort_values(by = "colmun_name", descending = True)
```

27. DataFrame.melt

The **melt()** function in pandas flips the columns in a DataFrame to individual rows. It's like exposing the anatomy of a DataFrame. So it lets you view the value assigned to each column explicitly.

```
newDataFrame = DataFrame.melt()
```

28. DataFrame.count

This function returns the total number of items in each column:

```
DataFrame.count()
```

29. DataFrame.query

pandas' **query()** lets you call items using their index number. To get the items in the third row, for example:

```
DataFrame.query('4') # Call the query on the fourth index
```

30. DataFrame.where

The **where()** function is a pandas query that accepts a condition for getting specific values in a column. For instance, to get all ages less than 30 from an **Age** column:

```
DataFrame.where(DataFrame['Age'] < 30)
```

The above code outputs a DataFrame containing all ages less than 30 but assigns **Nan** to rows that don't meet the condition.

Handle Data Like a Pro With pandas

pandas is a treasure trove of functions and methods for handling small to large-scale datasets with Python. The library also comes in handy for cleaning, validating, and preparing data for analysis or machine learning.

Taking the time to master it definitely makes your life easier as a data scientist, and it's well worth the effort. So feel free to pick up all the functions you can handle.



20 Python Functions You Should Know

[READ NEXT >](#)



RELATED TOPICS

PROGRAMMING

PYTHON

PROGRAMMING

DATABASE

ABOUT THE AUTHOR



Idowu Omisola

(124 Articles Published)



Idowu is passionate about anything smart tech and productivity. In his free time, he plays around with coding and switches to the chessboard when he's bored, but he also loves breaking away from routine once in a while. His...


[More From Idowu Omisola →](#)

SUBSCRIBE TO OUR NEWSLETTER


Join our newsletter for tech tips, reviews, free ebooks, and exclusive deals!

[CLICK HERE TO SUBSCRIBE](#)


ON THE WIRE

 Young adult using the Kik messaging app on their mobile phone.

What Is Kik and Why Do Teenagers Love It?

 A woman speaking into a Microsoft surface device

How to Use PowerPoint Speaker Coach to Improve Your Presentation Skills

 An iMac's screen shatters and scatters against a white background

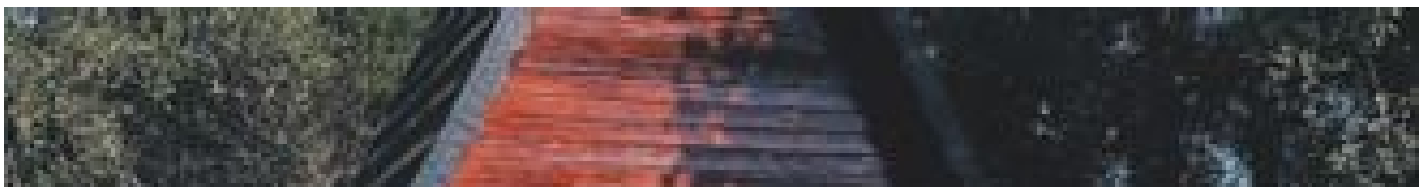
How to Clean Your Mac's Screen



The 20 Most Popular Android Apps in the Google Play Store



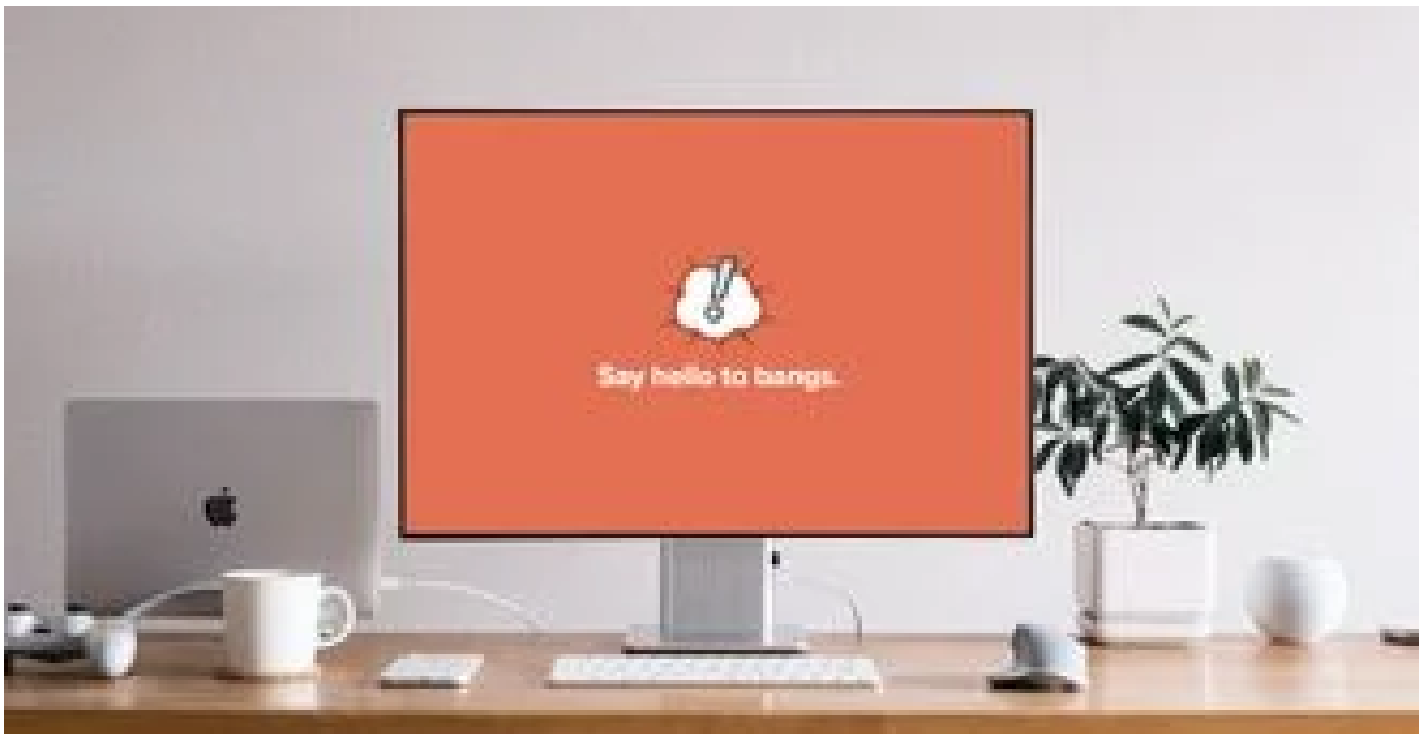
5 of the Biggest Windows 11 Issues Microsoft Needs to Fix



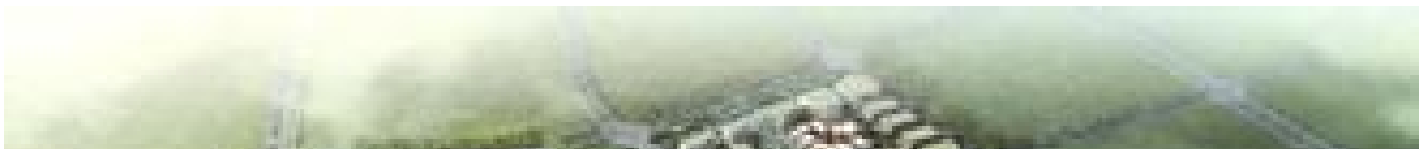


8 Awesome Gadgets for Your Tiny House

READ NEXT



What Are Bangs in DuckDuckGo?



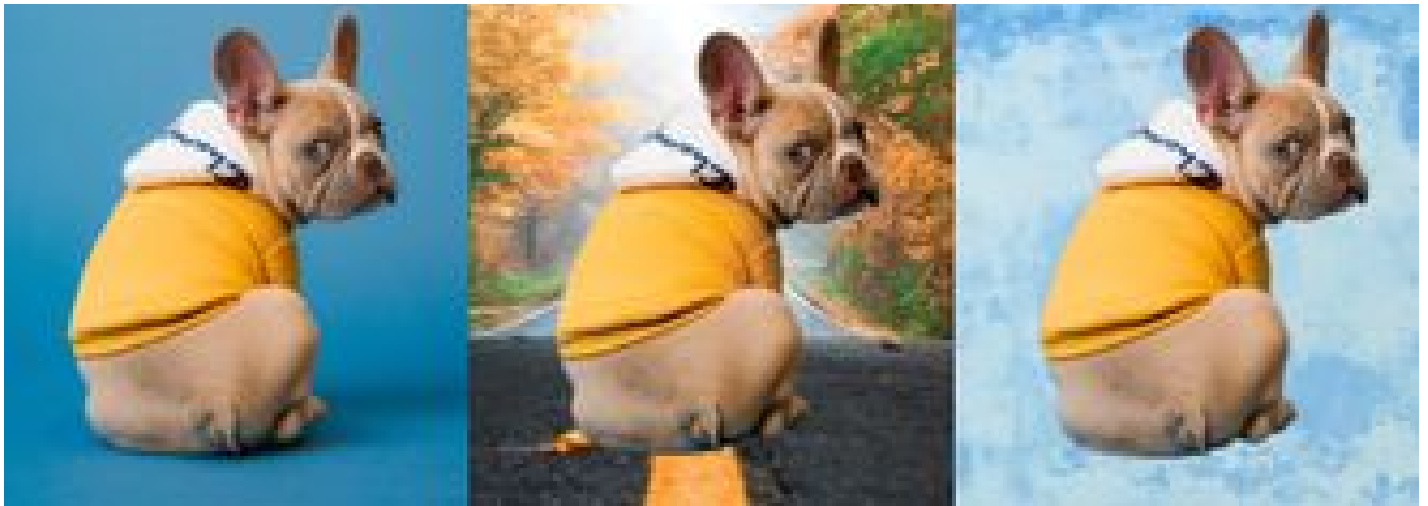


The Best Photogrammetry Software (Free and Paid)



Should Wordle Clones Be Allowed on the App Store?



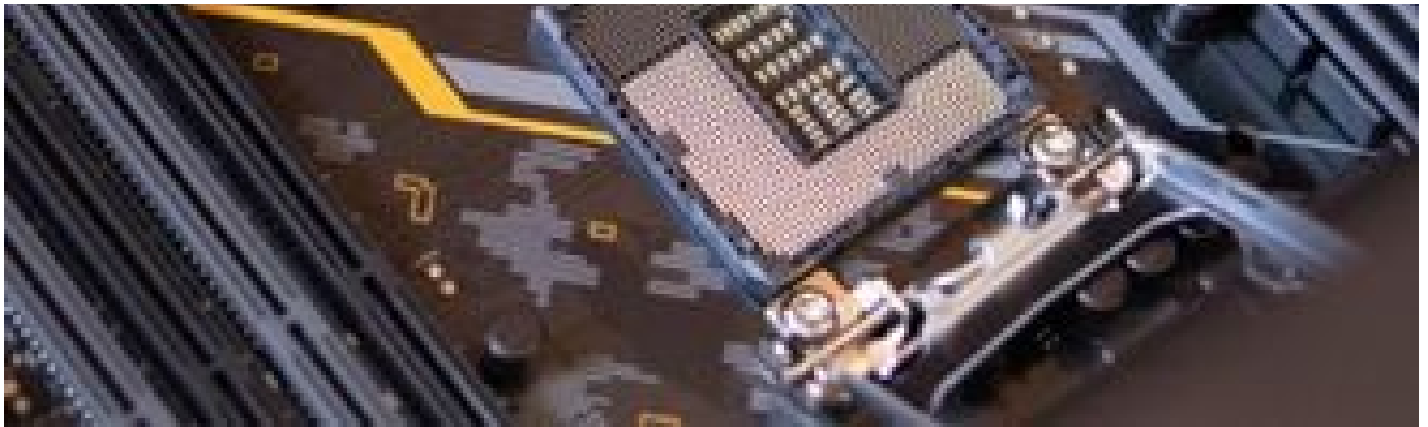


How to Remove an Image Background in CorelDRAW



What Does the Office Key on Keyboards Do?





Intel Z690 Chipset and Motherboard Guide: 6 Reasons to Upgrade




How to Overcome Collaboration Overload in Your Workplace





Cleer Ally Plus II Review: Clearly an All-Round Great Package

 youtube-alternatives-featured

12 Video Sites That Are Better Than YouTube

[Write For Us](#) [Home](#) [Contact Us](#) [Terms](#) [Privacy](#) [Copyright](#) [About Us](#) [Fact Checking Policy](#)

[Corrections Policy](#) [Ethics Policy](#) [Ownership Policy](#) [Partnership Disclaimer](#)

