



Published in Towards Data Science

This is your **last** free member-only story this month.

[Sign up for Medium and get an extra one](#)



Soner Yildirim

Follow

Nov 12, 2021 · 7 min read · ✨ · 🎧 Listen



Save



8 Methods For Handling Missing Values With Python Pandas

#7: Using the previous or next value



151



2



Photo by [Irina](#) on [Unsplash](#)

All the images were created by the author unless stated otherwise.

Missing values might be the most undesired values in data science. We definitely do not want to have them. However, they are always around.

Since it is not reasonable to ignore missing values, we need to find ways to handle them efficiently and properly.

Pandas, being one of the best data analysis and manipulation libraries, is quite flexible in handling missing values.

In this article, we will go over 8 different methods to make the missing values go away without causing a lot of trouble. Which method fits best to a particular situation depends on the data and task.

Let's start by creating a sample data frame and adding some missing values to it.

```

import numpy as np
import pandas as pd

df = pd.DataFrame({
    "Date": pd.date_range(start="2021-10-01", periods=10, freq="D"),
    "Item": 1014,
    "Measure_1": np.random.randint(1, 10, size=10),
    "Measure_2": np.random.random(10).round(2),
    "Measure_3": np.random.random(10).round(2),
    "Measure_4": np.random.randn(10)
})

```

(image by author)

	Date	Item	Measure_1	Measure_2	Measure_3	Measure_4
0	2021-10-01	1014	8	0.53	0.79	-0.062719
1	2021-10-02	1014	1	0.84	0.16	0.488597
2	2021-10-03	1014	4	0.78	0.20	0.097334
3	2021-10-04	1014	3	0.91	0.26	-0.994064
4	2021-10-05	1014	5	0.08	0.91	0.319882

First 5 rows of df (image by author)

We have a data frame with 10 rows and 6 columns.

The next step is to add the missing values. We will use the loc method to select the row and column combinations and make them equal to “np.nan” which is one of the standard missing value representations.



```
df.loc[[2,9], "Item"] = np.nan  
df.loc[[2,7,9], "Measure_1"] = np.nan  
df.loc[[2,3], "Measure_2"] = np.nan  
df.loc[[2], "Measure_3"] = np.nan  
df.loc[:6, "Measure_4"] = np.nan
```

(image by author)


Here is how the data frame looks now:

	Date	Item	Measure_1	Measure_2	Measure_3	Measure_4
0	2021-10-01	1014.0	8.0	0.53	0.79	NaN
1	2021-10-02	1014.0	1.0	0.84	0.16	NaN
2	2021-10-03	NaN	NaN	NaN	NaN	NaN
3	2021-10-04	1014.0	3.0	NaN	0.26	NaN
4	2021-10-05	1014.0	5.0	0.08	0.91	NaN
5	2021-10-06	1014.0	6.0	0.94	0.64	NaN
6	2021-10-07	1014.0	8.0	0.17	0.60	NaN
7	2021-10-08	1014.0	NaN	0.45	0.00	-1.000761
8	2021-10-09	1014.0	1.0	0.44	0.72	-1.871924
9	2021-10-10	NaN	NaN	0.01	0.39	1.745967

df (image by author)

The item and measure 1 columns had integer values but they have been upcasted to float because of the missing values.

With Pandas 1.0, an integer type missing value representation (<NA>) was introduced so we can have missing values in integer columns as well. However, we need to explicitly declare the data type.



```
df = df.astype({  
    "Item": pd.Int64Dtype(),  
    "Measure_1": pd.Int64Dtype()})
```

(image by author)

	Date	Item	Measure_1	Measure_2	Measure_3	Measure_4
0	2021-10-01	1014	8	0.53	0.79	NaN
1	2021-10-02	1014	1	0.84	0.16	NaN
2	2021-10-03	<NA>	<NA>	NaN	NaN	NaN
3	2021-10-04	1014	3	NaN	0.26	NaN
4	2021-10-05	1014	5	0.08	0.91	NaN
5	2021-10-06	1014	6	0.94	0.64	NaN
6	2021-10-07	1014	8	0.17	0.60	NaN
7	2021-10-08	1014	<NA>	0.45	0.00	-1.000761
8	2021-10-09	1014	1	0.44	0.72	-1.871924
9	2021-10-10	<NA>	<NA>	0.01	0.39	1.745967

df (image by author)

We are now able to preserve the integer columns despite having missing values.

We have a data frame with some missing values. It is time to see the different methods to handle them.

1. Drop rows or columns that have a missing value

One option is to drop the rows or columns that contain a missing value.



(image by author)

	Date	Item	Measure_1	Measure_2	Measure_3	Measure_4
8	2021-10-09	1014	1	0.44	0.72	-1.871924

(image by author)

With the default parameter values, the dropna function drops the rows that contain any missing value.

There is only one row in the data frame that does not have any missing values.

We can also choose to drop columns that have at least one missing value by using the axis parameter.



(image by author)

	Date
0	2021-10-01
1	2021-10-02
2	2021-10-03
3	2021-10-04
4	2021-10-05
5	2021-10-06
6	2021-10-07
7	2021-10-08
8	2021-10-09
9	2021-10-10

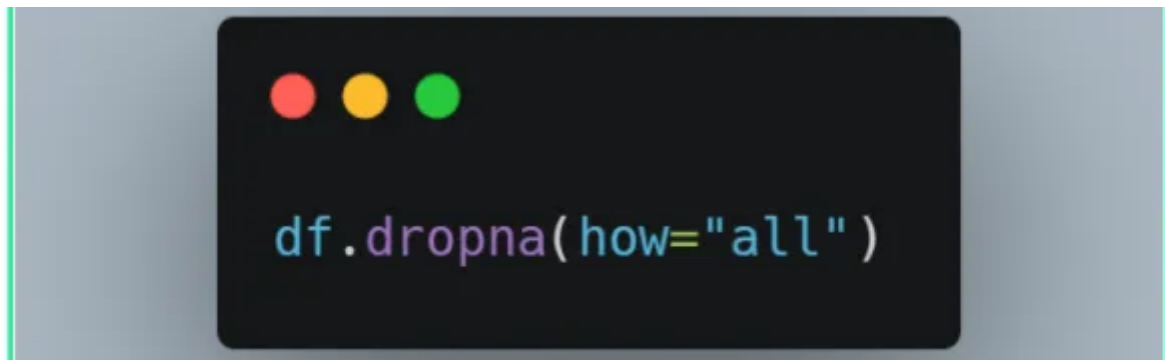
(image by author)

Only the date column does not have any missing values.

1. Drop rows or columns that only have missing values

Another situation is to have a column or row that is full of missing values. Such columns or rows are useless so we can drop them.

The dropna function can be used for this as well. We just need to change the value of how parameter.



(image by author)

	Date	Item	Measure_1	Measure_2	Measure_3	Measure_4
0	2021-10-01	1014	8	0.53	0.79	NaN
1	2021-10-02	1014	1	0.84	0.16	NaN
2	2021-10-03	<NA>	<NA>	NaN	NaN	NaN
3	2021-10-04	1014	3	NaN	0.26	NaN
4	2021-10-05	1014	5	0.08	0.91	NaN
5	2021-10-06	1014	6	0.94	0.64	NaN
6	2021-10-07	1014	8	0.17	0.60	NaN
7	2021-10-08	1014	<NA>	0.45	0.00	-1.000761
8	2021-10-09	1014	1	0.44	0.72	-1.871924
9	2021-10-10	<NA>	<NA>	0.01	0.39	1.745967

(image by author)

Since the data frame does not have a row full of missing values, no row has been dropped.

1. Drop rows or columns based on a threshold value

Dropping based on “any” or “all” is not always the best option. We sometimes need to drop rows or columns with “lots of” or “some” missing values.

We cannot assign such expressions to the how parameter but Pandas gives us a more accurate way which is the thresh parameter.

For instance, “thresh=4” means that the rows that have at least 4 non-missing values will be kept. The other ones will be dropped.

Our data frame has 6 columns so the rows that have 3 or more missing values will be dropped.



(image by author)

	Date	Item	Measure_1	Measure_2	Measure_3	Measure_4
0	2021-10-01	1014	8	0.53	0.79	NaN
1	2021-10-02	1014	1	0.84	0.16	NaN
3	2021-10-04	1014	3	NaN	0.26	NaN
4	2021-10-05	1014	5	0.08	0.91	NaN
5	2021-10-06	1014	6	0.94	0.64	NaN
6	2021-10-07	1014	8	0.17	0.60	NaN
7	2021-10-08	1014	<NA>	0.45	0.00	-1.000761
8	2021-10-09	1014	1	0.44	0.72	-1.871924
9	2021-10-10	<NA>	<NA>	0.01	0.39	1.745967

(image by author)

Only the third row had more than 2 missing values so it was the only one dropped.

4. Drop based on a particular subset of columns

We can take only some of the columns into consideration when dropping columns.

The subset parameter of the dropna function is used for this task. For instance, we can drop the rows that have a missing value in measure 1 or measure 2 columns as follows:



```
df.dropna(subset=["Measure_2","Measure_3"])
```

(image by author)

	Date	Item	Measure_1	Measure_2	Measure_3	Measure_4
0	2021-10-01	1014	8	0.53	0.79	NaN
1	2021-10-02	1014	1	0.84	0.16	NaN
4	2021-10-05	1014	5	0.08	0.91	NaN
5	2021-10-06	1014	6	0.94	0.64	NaN
6	2021-10-07	1014	8	0.17	0.60	NaN
7	2021-10-08	1014	<NA>	0.45	0.00	-1.000761
8	2021-10-09	1014	1	0.44	0.72	-1.871924
9	2021-10-10	<NA>	<NA>	0.01	0.39	1.745967

(image by author)

Up to this point, we have seen different methods for dropping rows or columns based on the missing values.

Dropping is not the only option. In some cases, we may choose to fill missing values instead of dropping them.

In fact, the filling might be a better option since data means value. How to fill the missing values, of course, depend on the structure of data and the task.

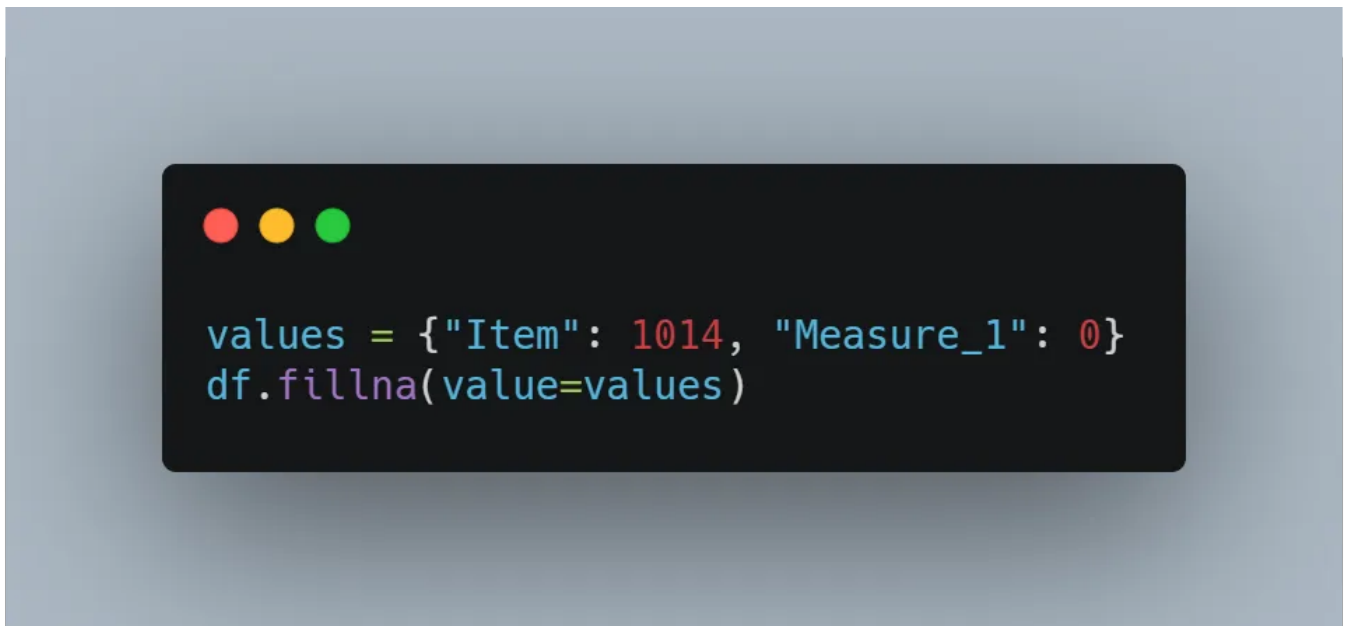
The fillna function is used for filling the missing values.

5. Fill with a constant value

We can choose a constant value to be used as a replacement for the missing values.

If we just give one constant value to the fillna function, it will replace all the missing values in the data frame with that value.

A more reasonable method is to determine separate constant values for different columns. We can write them in a dictionary and pass it to the values parameter.



(image by author)

	Date	Item	Measure_1	Measure_2	Measure_3	Measure_4
0	2021-10-01	1014	8	0.53	0.79	NaN
1	2021-10-02	1014	1	0.84	0.16	NaN
2	2021-10-03	1014	0	NaN	NaN	NaN
3	2021-10-04	1014	3	NaN	0.26	NaN
4	2021-10-05	1014	5	0.08	0.91	NaN
5	2021-10-06	1014	6	0.94	0.64	NaN
6	2021-10-07	1014	8	0.17	0.60	NaN
7	2021-10-08	1014	0	0.45	0.00	-1.000761
8	2021-10-09	1014	1	0.44	0.72	-1.871924
9	2021-10-10	1014	0	0.01	0.39	1.745967

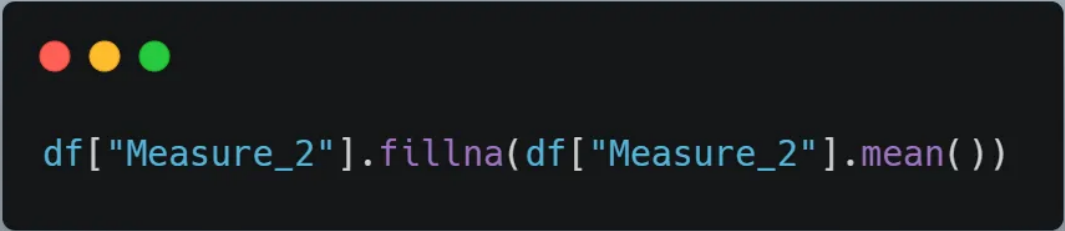
(image by author)

The missing values in the item column are replaced with 1014 and the ones in the measure 1 column are replaced with 0.

6. Fill with an aggregated value

Another option is to use an aggregated value such as mean, median, or mode.

The following line of code replaces the missing values in the measure 2 column with the average value of this column.



```
df["Measure_2"].fillna(df["Measure_2"].mean())
```

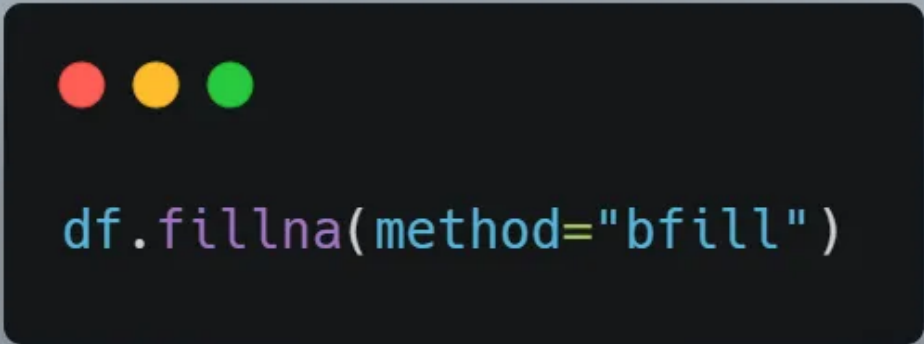
(image by author)

7. Replace with the previous or next value

It is possible to replace the missing values in a column with the previous or next value in that column.

This method might come in handy when working with time-series data. Consider you have a data frame that contains the daily temperature measurement and the temperature in one day is missing. The optimal solution would be to use the temperature in the next or previous day.

The method parameter of the fillna function is used for performing this task.



```
df.fillna(method="bfill")
```

(image by author)


	Date	Item	Measure_1	Measure_2	Measure_3	Measure_4
0	2021-10-01	1014	8	0.53	0.79	-1.000761
1	2021-10-02	1014	1	0.84	0.16	-1.000761
2	2021-10-03	1014	3	0.08	0.26	-1.000761
3	2021-10-04	1014	3	0.08	0.26	-1.000761
4	2021-10-05	1014	5	0.08	0.91	-1.000761
5	2021-10-06	1014	6	0.94	0.64	-1.000761
6	2021-10-07	1014	8	0.17	0.60	-1.000761
7	2021-10-08	1014	1	0.45	0.00	-1.000761
8	2021-10-09	1014	1	0.44	0.72	-1.871924
9	2021-10-10	<NA>	<NA>	0.01	0.39	1.745967

(image by author)

The “bfill” fills the missing values backward so they are replaced with the next value.

Take a look at the last column. The missing values are replaced up to the first row. This may not be suitable for some cases.

Thankfully, we can limit the number of missing values replaced with this method. If we set the limit parameter as 1, then a missing value can only be replaced with its next value. The second or third following value will not be used for replacement.



```
df.fillna(method="bfill", limit=1)
```

(image by author)

	Date	Item	Measure_1	Measure_2	Measure_3	Measure_4
0	2021-10-01	1014	8	0.53	0.79	NaN
1	2021-10-02	1014	1	0.84	0.16	NaN
2	2021-10-03	1014	3	NaN	0.26	NaN
3	2021-10-04	1014	3	0.08	0.26	NaN
4	2021-10-05	1014	5	0.08	0.91	NaN
5	2021-10-06	1014	6	0.94	0.64	NaN
6	2021-10-07	1014	8	0.17	0.60	-1.000761
7	2021-10-08	1014	1	0.45	0.00	-1.000761
8	2021-10-09	1014	1	0.44	0.72	-1.871924
9	2021-10-10	<NA>	<NA>	0.01	0.39	1.745967

(image by author)

8. Fill by using another data frame

We can also pass another data frame to the `fillna` function. The values in the new data frame will be used to replace the missing values in the current data frame.

The values will be selected according to the row indices and column names. For instance, if there is a missing value in the second row in the `item` column, the value in the same location in the new data frame will be used.

	Date	Item	Measure_1	Measure_2	Measure_3	Measure_4
0	2021-10-01	1014	1	0.27	0.65	-2.138382
1	2021-10-02	1014	1	0.66	0.83	0.160934
2	2021-10-03	1014	1	0.23	0.97	0.852453
3	2021-10-04	1014	2	0.52	0.34	-0.716053
4	2021-10-05	1014	1	0.85	0.23	0.378237
5	2021-10-06	1014	4	0.65	0.23	0.330072
6	2021-10-07	1014	5	0.62	0.93	-0.335675
7	2021-10-08	1014	4	0.60	0.86	1.072047
8	2021-10-09	1014	2	0.65	0.52	-2.471809
9	2021-10-10	1014	5	0.91	0.41	-2.032034

df2 (image by author)

	Date	Item	Measure_1	Measure_2	Measure_3	Measure_4
0	2021-10-01	1014	8	0.53	0.79	NaN
1	2021-10-02	1014	1	0.84	0.16	NaN
2	2021-10-03	<NA>	<NA>	NaN	NaN	NaN
3	2021-10-04	1014	3	NaN	0.26	NaN
4	2021-10-05	1014	5	0.08	0.91	NaN
5	2021-10-06	1014	6	0.94	0.64	NaN
6	2021-10-07	1014	8	0.17	0.60	NaN
7	2021-10-08	1014	<NA>	0.45	0.00	-1.000761
8	2021-10-09	1014	1	0.44	0.72	-1.871924
9	2021-10-10	<NA>	<NA>	0.01	0.39	1.745967

df (image by author)

Above are two data frames with the same columns. The first one (df2) does not have any missing values.

We can use the fillna function as follows:



(image by author)

	Date	Item	Measure_1	Measure_2	Measure_3	Measure_4
0	2021-10-01	1014	8	0.53	0.79	-2.138382
1	2021-10-02	1014	1	0.84	0.16	0.160934
2	2021-10-03	1014	1	0.23	0.97	0.852453
3	2021-10-04	1014	3	0.52	0.26	-0.716053
4	2021-10-05	1014	5	0.08	0.91	0.378237
5	2021-10-06	1014	6	0.94	0.64	0.330072
6	2021-10-07	1014	8	0.17	0.60	-0.335675
7	2021-10-08	1014	4	0.45	0.00	-1.000761
8	2021-10-09	1014	1	0.44	0.72	-1.871924
9	2021-10-10	1014	5	0.01	0.39	1.745967

(image by author)

The values in df are replaced with the values in df2 with respect to the column names and row indices.

Missing values will always be in our lives. There is no best method for handling them but we can lower their impact by applying accurate and reasonable methods.

We have covered 8 different methods for handling missing values. Which one to use depends on the data and the task.

If you are not a Medium member yet and plan to become one, I kindly ask you to do so using the following link. I will receive a portion of your membership fee at no additional cost to you.

Join Medium with my referral link - Soner Yıldırım

As a Medium member, a portion of your membership fee goes to writers you read, and you get full access to every story...

sonery.medium.com

Thank you for reading. Please let me know if you have any feedback.

Python

Programming

Data Science

Artificial Intelligence

Machine Learning

Enjoy the read? Reward the writer.^{Beta}

Your tip will go to Soner Yıldırım through a third-party platform of their choice, letting them know you appreciate their story.

Give a tip

Sign up for The Variable

By Towards Data Science

Every Thursday, the Variable delivers the very best of Towards Data Science: from hands-on tutorials and cutting-edge research to original features you don't want to miss. [Take a look.](#)

By signing up, you will create a Medium account if you don't already have one. Review our [Privacy Policy](#) for more information about our privacy practices.



Get this newsletter

Get the Medium app

