

# How can missing values be imputed in Pandas, and what is an example of this process?

Authored by  
**stats writer**

July 1, 2024

## RECOMMENDED CITATION

stats writer (2024). *How can missing values be imputed in Pandas, and what is an example of this process?*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=165359>

Missing values refer to any data that is not present or unavailable in a dataset. In Pandas, missing values can be imputed or filled in using various methods. One common method is to fill the missing values with the mean or median of the existing data. This can be done using the "fillna()" function in Pandas. For instance, if a dataset contains missing values in a particular column, the mean or median of that column can be calculated and used to fill in the missing values. This ensures that the overall data analysis is not affected by the missing values. Another method is to use the forward or backward filling technique, where the missing values are replaced with the previous or next available value in the dataset. This is useful when the data has a specific pattern. An example of this process would be filling in the missing values of daily temperature data by using the average temperature of the previous and next day. By imputing missing values, the dataset becomes more complete and can be used for further analysis.

## Impute Missing Values in Pandas (Including Example)

You can use the following basic syntax to impute missing values in a pandas DataFrame:

```
df = df.interpolate()
```

The following example shows how to use this syntax in practice.

**Example: Interpolate Missing Values in Pandas**

Suppose we have the following pandas DataFrame that shows the total sales made by a store during 15 consecutive days:

```
import pandas as pd
import numpy as np
```

```
#create DataFrame
```

```
df = pd.DataFrame({'day': ,  
'sales': })
```

```
#view DataFrame
```

```
print(df)
```

```
day sales
```

```
0 1 3.0
```

```
1 2 6.0
```

```
2 3 8.0
```

```
3 4 10.0
```

```
4 5 14.0
```

```
5 6 17.0
```

```
6 7 20.0
```

```
7 8 NaN
```

```
8 9 NaN
```

```
9 10 NaN
```

```
10 11 NaN
```

```
11 12 35.0
```

```
12 13 39.0
```

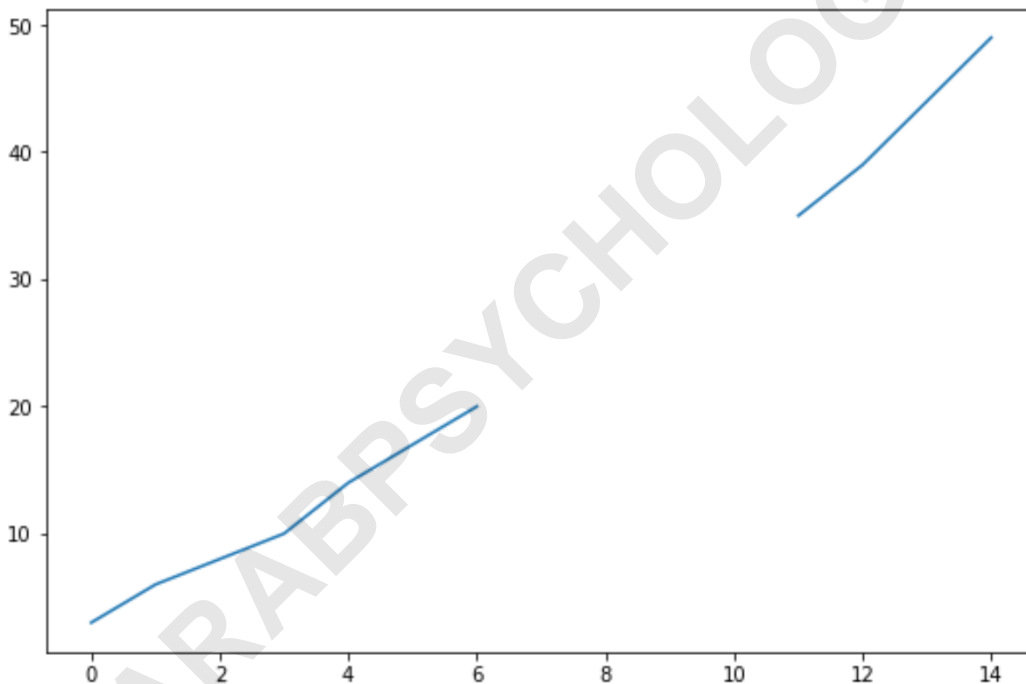
```
13 14 44.0
```

```
14 15 49.0
```

**Notice that we're missing sales numbers for four days in the data frame.**

**If we create a simple line chart to visualize the sales over time, here's what it would look like:**

```
#create line chart to visualize sales  
df.plot()
```



**To fill in the missing values, we can use the interpolate() function as follows:**

```
#interpolate missing values in 'sales' column  
df = df.interpolate()
```

```
#view DataFrame
```

```
print(df)
```

```
day sales
```

```
0 1 3.0
```

```
1 2 6.0
```

```
2 3 8.0
```

```
3 4 10.0
```

```
4 5 14.0
```

```
5 6 17.0
```

```
6 7 20.0
```

```
7 8 23.0
```

```
8 9 26.0
```

```
9 10 29.0
```

```
10 11 32.0
```

```
11 12 35.0
```

```
12 13 39.0
```

```
13 14 44.0
```

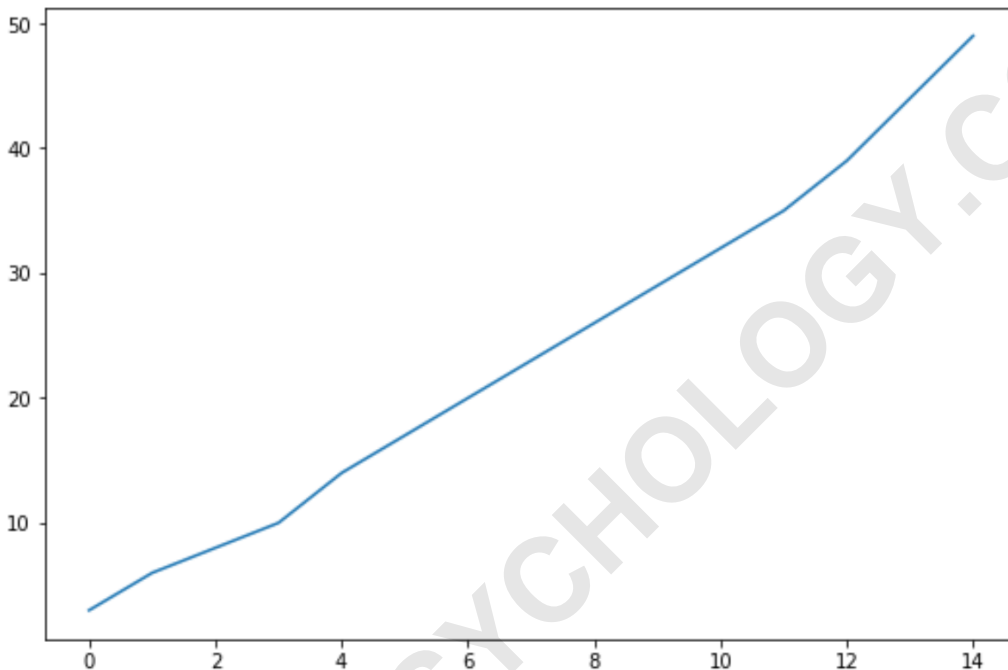
```
14 15 49.0
```

**Notice that each of the missing values has been replaced.**

**If we create another line chart to visualize the updated**

**data frame, here's what it would look like:**

```
#create line chart to visualize sales  
df.plot()
```



**Notice that the values chosen by the interpolate() function seem to fit the trend in the data quite well.**

**Additional Resources**

**The following tutorials provide additional information on how to handle missing values in pandas:**