How to Easily Convert Timedelta to Integer in Pandas

Authored by

stats writer

November 28, 2025

Working with time series data often requires manipulating duration measurements. In the Pandas library, these durations are encapsulated in Timedelta objects. While the Timedelta format is excellent for precise arithmetic operations, converting it into a simple numeric type--specifically an integer--is frequently necessary for analysis, modeling, or storage efficiency.

This article provides an in-depth guide on how to safely and effectively convert a Timedelta column within a DataFrame into an integer representation, such as total days, hours, or minutes. We will explore several powerful techniques, including the use of the specialized .dt accessor and division operations using the Timedelta constructor itself, ensuring the resulting data retains precision or is appropriately rounded.

We focus on generating clean, numerical output suitable for various data science tasks. Understanding these conversion methods is fundamental when dealing with temporal data in Python. You can use the standard Python `int()` function or the astype() method to finalize the conversion of any resulting floating-point value to an integer.

Setting Up the Sample DataFrame

To illustrate the conversion methods, we first need a sample DataFrame containing a Timedelta column. We will start by creating two columns containing datetime objects and then calculate the difference between them to generate our target column, named `duration`, which will hold the timedelta objects.

It is crucial to ensure that the initial date and end date columns are correctly parsed as datetime objects before performing subtraction, as this step automatically generates the **Timedelta** data type required for conversion. We use the `pd.to_datetime()` function for robust conversion.

The following code block sets up our environment, creates the sample dataset, and calculates the initial duration column:

import pandas as pd

#create DataFrame
df = pd.DataFrame({'promotion': ,
'start': ,
'end': })

#convert start date and end date columns to datetime
df = pd.to_datetime(df)
df = pd.to_datetime(df)

#create new column that contains timedelta between start and end

df = df - df

#view DataFrame
print(df)

promotion start end duration
0 A 2021-10-04 13:29:00 2021-10-08 11:29:06 3 days 22:00:06
1 B 2021-10-07 12:30:00 2021-10-15 10:30:07 7 days 22:00:07
2 C 2021-10-15 04:20:00 2021-10-29 05:50:15 14 days 01:30:15
3 D 2021-10-18 15:45:03 2021-10-22 15:40:03 3 days 23:55:00

Method 1: Extracting Integer Days Using the .dt Accessor

The simplest way to obtain an integer representation from a **Timedelta** object, specifically focusing on the number of full days, is by utilizing the .dt accessor. This accessor exposes properties like `.days`, `.seconds`, `.microseconds`, etc., which provide the discrete, component parts of the duration.

When you access the `.dt.days` attribute on a Timedelta Series, Pandas automatically extracts the integer component representing the number of whole days spanned by the duration. It is important to note that this method only captures the whole days and discards any remaining hours, minutes, or seconds, making it ideal for scenarios where only the major time unit is relevant for aggregation or filtering.

The following example demonstrates how to apply this technique to our `duration` column, creating a new column called `days`:

#create new column that converts timedelta into integer number of days
df = df.dt.days

#view updated DataFrame
print(df)

promotion start end duration days
0 A 2021-10-04 13:29:00 2021-10-08 11:29:06 3 days 22:00:06 3
1 B 2021-10-07 12:30:00 2021-10-15 10:30:07 7 days 22:00:07 7
2 C 2021-10-15 04:20:00 2021-10-29 05:50:15 14 days 01:30:15 14
3 D 2021-10-18 15:45:03 2021-10-22 15:40:03 3 days 23:55:00 3

As evident from the output, the fractional time component (e.g., 22 hours, 00 minutes, 06 seconds) has been ignored, leaving only the whole integer count of days. Verifying the data type confirms

that Pandas correctly assigned an **integer** type (`int64`) to the new column:

#check data type
df.days.dtype

dtype('int64')

Method 2: Converting Timedelta to Total Hours (Division Approach)

Often, instead of just extracting whole days, we need the total duration expressed in a smaller unit, such as hours or minutes, potentially including fractional parts. A highly effective and idiomatic Pandas method for this conversion is using division against a reference **Timedelta** unit.

To calculate the total hours, we divide the `duration` column (which is a **Timedelta** Series) by a unit of one hour, created using `pd.Timedelta(hours=1)`. This operation yields a Series of floating-point numbers (`float64`), representing the total time in hours, inclusive of the fractional component. This precision is essential when sub-day differences matter.

The initial conversion to hours using division is shown below:

#create new column that converts timedelta into total number of hours
df = df / pd.Timedelta(hours=1)

#view updated DataFrame
print(df)

promotion start end duration hours
0 A 2021-10-04 13:29:00 2021-10-08 11:29:06 3 days 22:00:06 94.001667
1 B 2021-10-07 12:30:00 2021-10-15 10:30:07 7 days 22:00:07 190.001944
2 C 2021-10-15 04:20:00 2021-10-29 05:50:15 14 days 01:30:15 337.504167
3 D 2021-10-18 15:45:03 2021-10-22 15:40:03 3 days 23:55:00 95.916667

As verified by the data type check, this intermediate step results in a floating-point number, which is necessary if high precision is needed:

#check data type
df.hours.dtype

dtype('float64')

Converting the Float Result to a True Integer (Hours)

Since the division method results in a floating-point type (`float64`), we must explicitly convert it to an integer if that is the final required output format. This conversion can be performed using the astype() method.

When converting from a float to an integer, you must decide on the desired rounding behavior. Using `.astype(int)` performs truncation (dropping the decimal part). If standard mathematical rounding is desired, you should use the `.round()` method before applying astype().

To obtain the total hours as an integer, we apply truncation using astype():

 # Convert float hours to truncated integer hours
df = df.astype('int')

#view updated DataFrame
print(df])

duration hours total_hours_int
0 3 days 22:00:06 94.001667 94
1 7 days 22:00:07 190.001944 190
2 14 days 01:30:15 337.504167 337
3 3 days 23:55:00 95.916667 95

Method 3: Converting Timedelta to Total Minutes (Division Approach)

The division methodology is scalable and easily adapted to any time unit supported by Pandas. To convert the **Timedelta** to total minutes, we repeat the division process, this time using `pd.Timedelta(minutes=1)` as the divisor.

This conversion is particularly useful when analyzing durations that are typically less than an hour, or when high granularity is required, such as in network latency analysis or short-term process monitoring. As with the hour conversion, the result will initially be a float (`float64`) due to the underlying implementation of time division in Pandas.

We execute the division to get the total minutes, including fractional seconds:

#create new column that converts timedelta into total number of minutes
df = df / pd.Timedelta(minutes=1)

#view updated DataFrame
print(df)

promotion start end duration minutes
0 A 2021-10-04 13:29:00 2021-10-08 11:29:06 3 days 22:00:06 5640.100000
1 B 2021-10-07 12:30:00 2021-10-15 10:30:07 7 days 22:00:07 11400.116667
2 C 2021-10-15 04:20:00 2021-10-29 05:50:15 14 days 01:30:15 20250.250000
3 D 2021-10-18 15:45:03 2021-10-22 15:40:03 3 days 23:55:00 5755.000000

The data type verification confirms the floating-point output:

#check data type
df.minutes.dtype

dtype('float64')

Converting the Float Result to a True Integer (Minutes)

To finalize the conversion to a true integer representation of total minutes, we apply the astype() conversion. Remember that this step inherently involves losing precision unless the original Timedelta was exactly divisible by the minute unit.

If the requirement is to capture the count of whole minutes elapsed, truncation is appropriate. If the goal is to round to the nearest minute, apply `.round()` before astype().

Here we convert the float minutes column to an integer using truncation:

 # Convert float minutes to truncated integer minutes
df = df.astype('int')

#view updated DataFrame
print(df])

duration minutes total_minutes_int
0 3 days 22:00:06 5640.100000 5640
1 7 days 22:00:07 11400.116667 11400
2 14 days 01:30:15 20250.250000 20250
3 3 days 23:55:00 5755.000000 5755

Alternative Method: Using total_seconds()

Another robust way to convert a **Timedelta** to a numeric representation, often favored for its simplicity, is the total_seconds() method. This method calculates the entire duration in seconds

and returns the result as a float.

While this immediately provides the total duration in a single, standard unit (seconds), it still requires an additional step to convert the resulting float Series into a required integer format, typically using `.astype('int')`.

If you need the total duration in units other than seconds (e.g., total minutes), you must divide the result of total_seconds() by the appropriate factor (e.g., 60 for minutes, 3600 for hours) before applying the final integer conversion.

Example: Converting to Total Integer Seconds

This demonstrates the direct application of total_seconds() followed by immediate conversion to a truncated integer:

 # Convert duration to total seconds (float)
df = df.dt.total_seconds()

# Convert to integer seconds
df = df.astype('int64')

#view updated DataFrame
print(df])

duration int_seconds
0 3 days 22:00:06 338406
1 7 days 22:00:07 684007
2 14 days 01:30:15 1215015
3 3 days 23:55:00 345300

Summary of Conversion Techniques and Use Cases

Choosing the correct method depends entirely on the required unit of measurement and whether fractional time units should be preserved or truncated.

We have identified two primary strategies for obtaining an integer representation from a **Timedelta** Series:

Using the .dt accessor (`.dt.days`): This is the cleanest and fastest method if you only require the count of whole days. It intrinsically returns an integer (`int64`) and automatically truncates any remaining time units.

Using Division by `pd.Timedelta()` or total_seconds(): These methods are necessary when you need the duration expressed in smaller units (hours, minutes, seconds, milliseconds) or when you require the total accumulated value. Because they return a float initially, an explicit conversion using `.astype('int')` or `.round().astype('int')` is mandatory to achieve the final integer format.

Always verify the resulting data type using `.dtype` after conversion to ensure that the Pandas Series is correctly interpreted as an integer by subsequent analytical tools or database systems.