Pandas DateTime

In Pandas, DateTime is a data type that represents a single point in time. It is especially useful when dealing with time-series data like stock prices, weather records, economic indicators etc.

We use the `to_datetime()` function to convert strings to the DateTime object. Let's look at an example.

```
import pandas as pd

# create a datetime string
date_string = '2001-12-24 12:38'

print("String:", date_string)

# convert string to datetime
date = pd.to_datetime(date_string)

print("DateTime:", date)
print(type(date))
```

```
String: 2001-12-24 12:38
DateTime: 2001-12-24 12:38:00
<class 'pandas._libs.tslibs.timestamps.Timestamp'>
```

In the above example, we used `to_datetime()` to convert a string to `DateTime`.

Convert String to DateTime

As shown in the example above, we can convert any valid string to DateTime using `to_datetime()`.

Let's look at some examples.

Example: `to_datetime()` With Default Arguments

```
import pandas as pd

# create a dataframe with date strings
df = pd.DataFrame({'date': ['2021-01-13', '2022-10-22', '2023-12-
03']})

# convert the 'date' column to datetime
df['date'] = pd.to_datetime(df['date'])

print(df)
```

```
        date
0 2021-01-13
1 2022-10-22
2 2023-12-03
```

In this example, we converted the column `date` from string to DateTime data type.

By default, Pandas' `to_datetime()` function expects the date string to be in the `YYYY-MM-DD` format.

Example: to_datetime() With Day First Format

```python
import pandas as pd

# create a dataframe with date strings in day-first format
df = pd.DataFrame({'date': ['13-02-2021', '22-03-2022', '30-04-
2023']})

# convert the 'date' column to datetime with day-first format
df['date'] = pd.to_datetime(df['date'], dayfirst=True)

print(df)
```

```
        date
0 2021-02-13
1 2022-03-22
2 2023-04-30
```

In this example, the date column contains strings in the format `DD-MM-YYYY`.

We passed `dayfirst=True` to `to_datetime()` function to convert the string in day first format to DateTime.

Notice that the DateTime data is always in the format `YYYY-MM-DD`.

Example: to_datetime() With Custom Format

```python
import pandas as pd

# create a dataframe with date strings in custom format
df = pd.DataFrame({'date': ['2021/22/01', '2022/13/01',
'2023/30/03']})

# convert the 'date' column to datetime with custom format
df['date'] = pd.to_datetime(df['date'], format='%Y/%d/%m')

print(df)
```

```
        date
0 2021-01-22
```

```
1 2022-01-13
2 2023-03-30
```

In this example, we converted the date column from string (in `YY/DD/MM` format) to DateTime data type

Get DateTime From Multiple Columns

We can also use the `to_datetime()` function to assemble the DateTime from multiple columns.

Let's look at an example.

```
import pandas as pd

# create a dataframe with separate date and time columns
df = pd.DataFrame({'year': [2021, 2022, 2023],
                   'month': [1, 2, 3],
                   'day': [1, 2, 3],
                   'hour': [10, 11, 12],
                   'minute': [30, 45, 0],
                   'second': [0, 0, 0]})

# combine date and time columns to create a datetime column
df['datetime'] = pd.to_datetime(df[['year', 'month', 'day', 'hour',
'minute', 'second']])

print(df)
```

```
   year  month  day  hour  minute  second            datetime
0  2021      1    1    10      30       0 2021-01-01 10:30:00
1  2022      2    2    11      45       0 2022-02-02 11:45:00
2  2023      3    3    12       0       0 2023-03-03 12:00:00
```

In this example, we assembled the complete date and time from different columns by passing the list of columns to the `to_datetime()` function.

Get Year, Month and Day From DateTime

We can use the inbuilt attributes `dt.year`, `dt.month` and `dt.day` to get year, month and day respectively from Pandas DateTime object.

Let's look at an example.

```
import pandas as pd

# create a dataframe with a datetime column
df = pd.DataFrame({'datetime': ['2021-01-01', '2022-02-02', '2023-03-
03']})
```

```
# convert the 'datetime' column to datetime type
df['datetime'] = pd.to_datetime(df['datetime'])

# extract year, month, and day into separate columns
df['year'] = df['datetime'].dt.year
df['month'] = df['datetime'].dt.month
df['day'] = df['datetime'].dt.day

print(df)

    datetime  year  month  day
0 2021-01-01  2021      1    1
1 2022-02-02  2022      2    2
2 2023-03-03  2023      3    3
```

Get Day of Week, Week of Year and Leap Year

We also have inbuilt attributes to get the day of the week, week of the year and to check whether the given year is a leap year.

For example,

```
import pandas as pd

# create a dataframe with a datetime column
df = pd.DataFrame({'datetime': ['2021-01-01', '2024-02-02', '2023-03-
03']})

# convert the 'datetime' column to datetime type
df['datetime'] = pd.to_datetime(df['datetime'])

# get the day of the week
df['day_of_week'] = df['datetime'].dt.day_name()

# get the week of the year
df['week_of_year'] = df['datetime'].dt.isocalendar().week

# check for leap year
df['leap_year'] = df['datetime'].dt.is_leap_year

print(df)

    datetime day_of_week  week_of_year  leap_year
0 2021-01-01      Friday            53      False
1 2024-02-02      Friday             5       True
2 2023-03-03      Friday             9      False
```

Here,

- `dt.day_name()`- returns the day of the week
- `dt.isocalender().week`- week returns the week of the year and
- `dt.is_leap_year`- checks if the DateTime is a leap year.

DateTime Index in Pandas

DateTime index in Pandas uses DateTime values as index values.

A datetime index is particularly useful when dealing with time series data like weather data, stock prices, and other time-dependent data, as it allows natural organization and manipulation based on timestamps.

Let's look at an example

```
import pandas as pd

# create a list of datetime values
dates = ['2021-01-01', '2021-01-02', '2021-01-03', '2021-01-04',
'2021-01-05']

# create a DataFrame with a DateTimeIndex
df = pd.DataFrame({'values': [10, 20, 30, 40, 50]},
index=pd.to_datetime(dates))

print(df)
```

```
            values
2021-01-01      10
2021-01-02      20
2021-01-03      30
2021-01-04      40
2021-01-05      50
```