# Python DateTime Format Using Strftime()

Updated on: May 6, 2022 | 2 Comments

This tutorial will teach how to represent date and time into various formats in Python using the `strftime()` function of a datetime module and time module.

The `strftime()` method returns a string representing of a `datetime` object according to the format codes.

## Table of contents



## How to Format Date and Time in Python

In Python, the date and time values are stored as `datetime` objects, but there are cases where we need to print the `datetime` objects into various string formats for better readability.

For example, you may need to represent a date numerically in format, like "**17-06-2021**". On the other hand, you want to convert dates in textual string format like "**Tuesday, 23 June 2021**."

The below steps show how to convert a datetime to string format using the `strftime()` function

1. **Import datetime module**

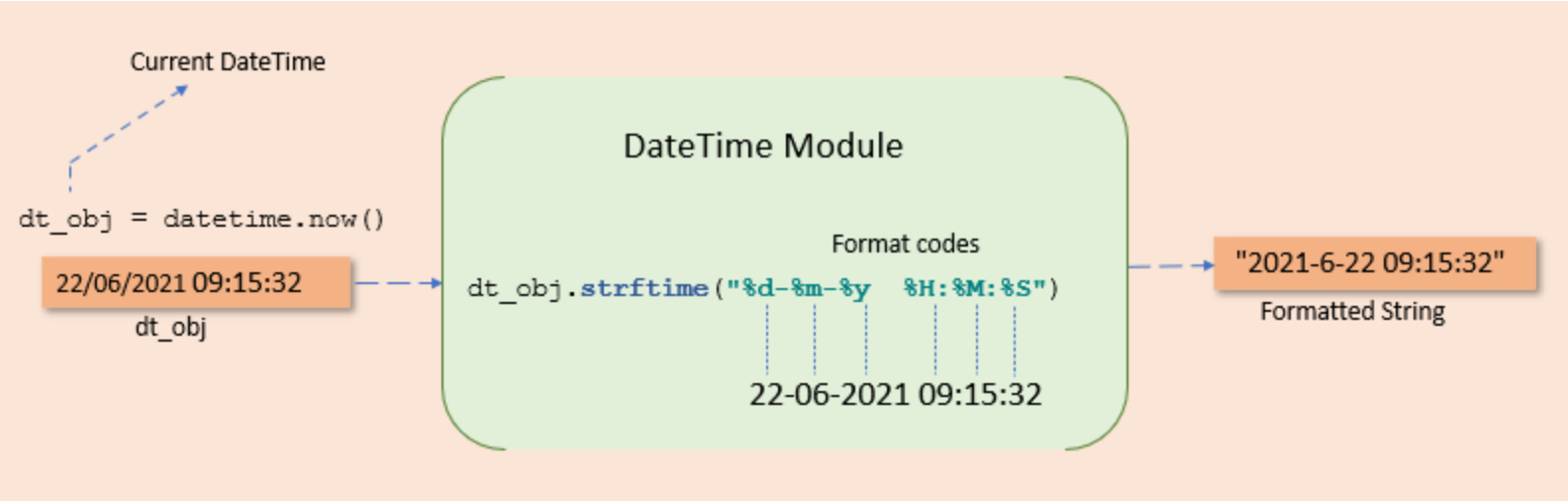
   Python's datetime module provides functions that handle many complex functionalities involving the date and time. Import `datetime` class using a `from datetime import datetime` statement.

2. **Use strftime() function of a datetime class**

   Use `datetime.strftime(format)` to convert a `datetime` object into a string as per the corresponding `format`.
   The format codes are standard directives for mentioning in which format you want to represent datetime. For example, the `%d-%m-%Y %H:%M:%S` codes convert date to `dd-mm-yyyy hh:mm:ss` format.

3. **Use strftime() function of a time module**

   Use this step if you want to convert a `time` object to string format. like, hours minutes seconds (`hh:mm:ss`). Use the `time.strptime(string[, format])` function to convert a `time` object to a string format.




strftime() to convert DateTime to string format

## Example: Convert DateTime to String Format

**Syntax**:

```
datetime_object.strftime(format)
```

- First, get the current datetime the `now()` function
- Next, use the `strftime()` with appropriate format codes.

Let us see the example to convert today's datetime in string format of `YYYY-MM-DD hh:mm:ss`

```
from datetime import datetime

# current dateTime
now = datetime.now()

# convert to string
date_time_str = now.strftime("%Y-%m-%d %H:%M:%S")
print('DateTime String:', date_time_str)

# Output 2021-07-20 16:26:24
```

Run

Also, refer to **convert a string to DateTime in Python**

**Convert individual attributes of a `datetime` object to a string format**: –

For example, you can convert only date, time, year, or day from a `datetime` object to a string using the appropriate format code.

**Example**:

```
from datetime import datetime

# current dateTime
now = datetime.now()

# convert to date String
date = now.strftime("%d/%m/%Y")
print('Date String:', date)

# convert to time String
time = now.strftime("%H:%M:%S")
print('Time String:', time)

# year
year = now.strftime("%Y")
print('Year String:', year)

# Month
month = now.strftime("%m")
print('Month String:', month)

# Day
```

```
day = now.strftime("%d")
print('Day String:', day)
```

Run

**Output**:

```
Date String: 23/06/2021
Time String: 10:07:04
Year String: 2021
Month String: 06
Day String: 23
```

# strftime() Date Format Codes

Dates have a default representation, but you may want to print them in a specific format. In that case, you can get a custom string representation using the different format codes

The `strftime()` uses some standard directives to represent a `datetime` in a string format. The same set of directives are shared between both the `strptime()` and `strftime()` methods.

Below are the character codes to format the date and time:-

- `%d` : Returns the **day** of the month, from 1 to 31.
- `%m` : Returns the **month** of the year, from 1 to 12.
- `%Y` : Returns the year in four-digit format (**Year** with century). like, 2021.
- `%y` : Reurns year in two-digit format (**year** without century). like, 19, 20, 21
- `%A` : Returns the full name of the **weekday**. Like, Monday, Tuesday
- `%a` : Returns the short name of the **weekday** (First three character.). Like, Mon, Tue
- `%B` : Returns the full name of the **month**. Like, June, March
- `%b` : Returns the short name of the **month** (First three character.). Like, Mar, Jun
- `%H` : Returns the **hour**. from 01 to 23.
- `%I` : Returns the **hour** in 12-hours format. from 01 to 12.
- `%M` : Returns the **minute**, from 00 to 59.
- `%S` : Returns the **second**, from 00 to 59.
- `%f` : Return the **microseconds** from 000000 to 999999
- `%p` : Return time in **AM/PM** format
- `%c` : Returns a **locale's appropriate date and time** representation

- `%x` : Returns a locale's appropriate date representation
- `%X` : Returns a locale's appropriate time representation
- `%z` : Return the **UTC offset** in the form `±HHMM[SS[.ffffff]]` (empty string if the object is naive).
- `%Z` : Return the **Time zone name** (empty string if the object is naive).
- `%j` : Returns the day of the year from *01 to 366*
- `%w` : Returns weekday as a decimal number, where 0 is Sunday and 6 is Saturday.
- `%U` : Returns the week number of the year (Sunday as the first day of the week) from 00 to 53
- `%W` : Returns the week number of the year (Monday as the first day of the week) from 00 to 53

We have seen how to convert dates to strings using the default format codes. Now we can see more combinations with examples for a better understanding of the format codes that you can use to format dates in Python.

## Represent Dates in Numerical Format

The numerical format means to display the day, month, year, hours, minutes, seconds in numbers. like, `2021-07-07 12:19:47.864519`

```
from datetime import datetime

# Get current Date
x_date = datetime.now()
print('Current Date:', x_date)

# Represent Dates in numerical format
print("dd-mm-yyyy HH:MM:SS:", x_date.strftime("%d-%m-%y %H:%M:%S"))
print("dd-mm-yyyy:", x_date.strftime("%d-%m-%Y"))
print("dd-mm-yy Format:", x_date.strftime("%d-%m-%y"))
```

Run

**Output**:

```
Current Date: 2021-07-07 12:19:47.864519
dd-mm-yyyy HH:MM:SS: 07-07-21 12:19:47
dd-mm-yyyy: 07-07-2021
dd-mm-yy Format: 07-07-21
```

## Represent Dates in Textual Format

The textual format means to display the **month name and day name**. like, `Wednesday,07 July, 2021` . You can print the full name and short name of a day and month.

- `%A` : Full name of the **day**. Like, Monday
- `%a` : Short name of the **day**. Like, Mon, Tue
- `%B` : Full name of the **month**. Like, December
- `%b` : Short name of the **month**. Like, Mar

```
from datetime import datetime

# Get current Date
x_date = datetime.now()
print('Current Date:', x_date)

# Represent Dates in full textual format
print("dd-MonthName-yyyy:", x_date.strftime("%d-%B-%Y"))
print("DayName-dd-MonthName-yyyy:", x_date.strftime("%A,%d %B, %Y"))

# Represent dates in short textual format
print("dd-MonthName-yyyy:", x_date.strftime("%d-%b-%Y"))
print("DDD-dd-MMM-yyyy:", x_date.strftime("%a,%d %b, %Y"))
```

Run

**Output**:

```
Current Date: 2021-07-07 12:19:47.864519

dd-MonthName-yyyy: 07-July-2021
DayName-dd-MonthName-yyyy: Wednesday,07 July, 2021

dd-MonthName-yyyy: 07-Jul-2021
DDD-dd-MMM-yyyy: Wed,07 Jul, 2021
```

# Convert Only `Date` to String

The `strftime()` method can be called using the `date` , `time` , or `datetime` objects. Let's how to format only `date` object of a datetime module to string.

**Example**:

```
from datetime import date

# current date
today = date.today()
print("Today's date:", today)

# format date
print('Date String', today.strftime("%d-%m-%y"))
```

Run

**Output**:

```
Today's date: 2021-07-07
Date String 07-07-21
```

**Note**: You can also **extract `date` object from a `datetime` object** and convert it to a string if required.

```
from datetime import datetime

# extract date object
today = datetime.now().date()
# format date
print('Date String', today.strftime("%d-%m-%y"))
```

Run

# Convert `Time` Object to String Format

Same as the `date` object you can convert the `time` object of a datetime module to a string.

- Use the `time()` constructor to create a time object Or
- Extract the time object from a `datetime` object using the `datetime.time()` method.

Let's see how to format DateTime to print time in hours, minutes, and seconds, and microsecond format.

## Represent time in 24-hours and 12-hours Format

- Use the `%H-%M-%S` format code to display time in **24-hours** format
- Use the `%I-%M-%S` format code to display time in **12-hours** format

```
from datetime import datetime

# Get current time
x_time = datetime.now().time()
print('Current Time:', x_time)

print("Time in 24 hours format:", x_time.strftime("%H-%M-%S"))
print("Time in 12 hours format:", x_time.strftime("%I-%M-%S"))
```

Run

**Output**:

```
Current Time: 15:56:49.810391
Time in 24 hours format: 15-56-49
Time in 12 hours format: 03-56-49
```

## Represent Time in Microseconds Format

- Use the `%f` format code to represent time in **microsecond**
- Use the `%p` format code to represent time in **AM/PM** format

```
from datetime import datetime

# Get current time
x_time = datetime.now().time()
print('Current Time:', x_time)

# Represent time in Microseconds (HH:MM:SS.Microsecond)
print("Time is:", x_time.strftime("%H:%M:%S.%f"))
```

Run

**Output**:

```
Current Time: 15:59:35.189231
Time is: 15:59:35.189231
```

## Represent DateTime in Milliseconds

As there is no formatting code available for milliseconds, we can only display it using the `%S` code. However, as milliseconds are 3 decimal places away from seconds, we can display that information by combining `%S` with `%f`.

**Example**:

```
from datetime import datetime

# Current Date and time with milliseconds
print(datetime.now().strftime('%Y-%m-%d %H:%M:%S.%f')[:-3])

# Output 2021-07-08 08:47:46.851
```

Run

## Represent Time in AM/PM Format

Use the `%p` format code to represent time in **AM/PM** format.

```
from datetime import datetime

# Get current Datetime
dt = datetime.now()
print('Current Time:', dt)

# %p to represent datetime in AM/PM
print("Time is:", dt.strftime("%d-%b-%Y %I.%M %p"))

# Represent only time in AM/PM
print("Time is:", dt.time().strftime("%H.%M %p"))
```

Run

**Output**:

```
Current Time: 2021-07-08 11:56:19.363470
Time is: 08-Jul-2021 11.56 AM
Time is: 11.56 AM
```

**Note**:

- For `time` objects, the format codes for the year, month, and day should not be used, as time objects have no such values. If they're used anyway, 1900 is substituted for the year, and 1 for the month and day.
- For `date` objects, the format codes for hours, minutes, seconds, and microseconds should not be used, as date objects have no such values. If they're used anyway, 0 is substituted for them.

## Format `time` Object to String Using time module

The time module provides various time-related functions. If you are using a time module in your application and wanted to format the `time` object to string representation, then use the `strftime()` method available in the time module.

This is just similar to the datetime module's method except that it accepts a two arguments.

**Syntax**:

```
time.srtftime(format[, t])
```

This method converts a tuple or `struct_time` representing a time as returned by `gmtime()` or `localtime()` to a string as specified by the `format` argument.

The `strftime()` method of a time module takes two parameters:

- `format` : The format code. It must be string
- `t` : The time tuple that needs to be converted to a string.

**Example**: Converting the current time to string using the `time.strftime()` method.

```
# time module
import time

time_obj = time.time()
# getting local time from current time in seconds
local_time = time.localtime(time_obj)

print("The time tuple:", local_time)

# Formatting the time to display in string format
print('Formatted Time:', time.strftime("%d/%m/%y %H:%M:%S", local_time))
```

Run

**Output**

```
The time tuple: time.struct_time(tm_year=2021, tm_mon=6, tm_mday=23, tm_hour=10, tm_min=33,

Formatted Time: 23/06/21 10:33:02
```

# Convert Datetime to locale's Format

The `%c` directive returns a **locale's appropriate date and time** representation of a given `datetime` object.

```
from datetime import datetime

# Get current time
x_date = datetime.now()
print('Date is:', x_date)

# %c datetime in locale
print("Date is:", x_date.strftime("%c"))
```

Run

# Convert Datetime in ISO String format

We can display the `datetime` in an ISO 8601 format String. In the ISO 8601 string, the timezone is displayed as a UTC offset. We can do this by using the `%z` and `%Z` format directive. For this requirement, we can use the pytz for getting the timezone name.

- Get the current datetime using the `datetime.now()` function
- Assign the timezone to the current timestamp using the `datetime.fromtimestamp()`
- Us the `%Z` format directive to display the datetime in ISO 8601 format.

```
from datetime import datetime

# # pip install pytz
import pytz
```

```
# get timestamp
ts = datetime.now().timestamp()

# assigning the timezone to the current timestamp
dt = datetime.fromtimestamp(ts, pytz.timezone('America/New_York'))

# displaying the datetime string in ISO format
print('ISO Date Format:', dt.strftime('%Y-%m-%d %H:%M:%S%z (%Z)'))
```

Run

**Output**

```
ISO Date Format: 2021-07-07 06:36:55-0400 (EDT)
```

# Converting Datetime to Int

We have seen how to display the datetime in different formats as a string, but there will be requirements to store it as an integer. This is equivalent to adding all the values in the date and time with their place values.

This can be done by simply giving their format strings together without space. It will add the values along with their place values.

```
from datetime import datetime

dt = datetime.now()
x_int = int(dt.strftime("%Y%m%d%H%M%S"))
print("Current date as Integer::", x_int)

# convert back to datetime
dt = datetime.strptime(str(x_int), '%Y%m%d%H%M%S')
print('DateTime is:', dt)
```

Run

**Output**:

```
Current date as Integer:: 20210707164420
DateTime is: 2021-07-07 16:44:20
```

# Convert Datetime to Float

We can convert the datetime String to float with a precision of microseconds. Or store the date and time information separately as well.

```
from datetime import datetime

dt = datetime.now()
x_float = float(dt.strftime("%Y%m%d%H%M%S.%f"))
print("Current date as Float::", x_float)

# convert back to datetime
dt = datetime.strptime(str(x_float), '%Y%m%d%H%M%S.%f')
print('DateTime is:', dt)
```

Run

**Output**

```
Current date as Float:: 20210707164825.96
DateTime is: 2021-07-07 16:48:25.960000
```

Filed Under: Python, Python DateTime

**Did you find this page helpful?** Let others know about it. **Sharing helps me** continue to create free Python resources.



## About Vishal

Founder of PYnative.com I am a Python developer and I love to write articles to help developers. Follow me on Twitter. All the best for your future Python endeavors!