**Reference guide: Date/time manipulation**

The following tables can serve as reference guides to remind you of the shorthand code for manipulating datetime strings into individual objects.

## Manipulating datetime strings in Python

Below that you will find a table with the datetime functions you can use to help you manipulate datetime objects in different ways.

| **Code** | **Format** | **Example** |
| --- | --- | --- |
| **%a** | Abbreviated workday | Sun |
| **%A** | Weekday | Sunday |
| **%b** | Abbreviated month | Jan |
| **%B** | Month name | January |
| **%c** | Date and time | Sun Jan 1 00:00:00 2021 |
| **%d** | Day (leading zeros) | 01 to 31 |
| **%H** | 24 hours | 00 to 23 |
| **%I** | 12 hours | 01 to 12 |
| **%j** | Day of year | 001 to 366 |
| **%m** | Month | 01 to 12 |
| **%M** | Minute | 00 to 59 |
| **%p** | AM or PM | AM/PM |
| **%S** | Seconds | 00 to 61 |
| **%U** | Week number (Sun) | 00 to 53 |
| **%W** | Week number (Mon) | 00 to 53 |
| **%w** | Weekday | 0 to 6 |
| **%x** | Locale’s appropriate date representation | 08/16/88 (None);  08/16/1988 (en\_US);  16.08.1988 (de\_DE) |
| **%X** | A locale’s appropriate time representation | 21:30:00 (en\_US);  21:30:00 (de\_DE) |
| **%y** | Year without century | 00 to 99 |
| **%Y** | Year | 2022 |
| **%z** | Offset | +0900 |
| **%Z** | Time zone | EDT/JST/WET etc (GMT) |

## 

## Datetime functions to remember

All of the following date string manipulations require the datetime package to be imported first. (from datetime import datetime)

| **Code** | **Input Type** | **Input Example** | **Output Type** | **Output Example** |
| --- | --- | --- | --- | --- |
| datetime.strptime(“25/11/2022”, “%d/%m/%Y”) | string | “25/11/2022” | DateTime | “2022-11-25 00:00:00” |
| datetime.strftime(dt\_object, “%d/%m/%Y”) | DateTime | “2022-11-25 00:00:00” | string | “25/11/2022” |
| dt\_object = datetime.strptime(“25/11/2022”, “%d/%m/%Y”)  datetime.timestamp(dt\_object) | string | “25/11/2022” | float (UTC timestamp in seconds) | 1617836400.0 |
| datetime.strptime(“25/11/2022”, “%d/%m/%Y”).strftime(“%Y-%m-%d”) | string | “25/11/2022” | string | “2022-11-25” |
| datetime.fromtimestamp(1617836400.0) | float (UTC timestamp in seconds) | 1617836400.0 | DateTime | “2022-11-25 00:00:00” |
| datetime.fromtimestamp(1617836400.0).strftime(“%d/%m/%Y”) | float (UTC timestamp in seconds) | 1617836400.0 | string | “25/11/2022” |
| from pytz import timezone  My\_time = datetime.strptime(“25-11-2022 09:34:00-0700”, “%d-%m-%Y %H:%M:%S%f%z”)  Tokyo\_time = ny\_time.astimezone(timezone(‘Asia/Tokyo’)) | string | NewYork timezone “25-11-2022 09:34:00-0700” | DateTime | Tokyo timezone “2022-11-25 22:34:00+08:00” |
| datetime.strptime(“20:00”, “%H:%M”).strftime(“%I:%M %p”) | string | “20:00” | string | “09:00 AM” |
| datetime.strptime(“08:00 PM”, “%I:%M %p”).strftime(“%H:%M”) | string | “08:00 PM” | string | “20:00” |

## 

## Key takeaways

Use reference guides like the tables above throughout your career to help remind you of the different ways to manipulate datetime objects. Even experts in the field use reference guides, rather than memorizing all this information. Getting familiar with guides like these will be beneficial because you will be using them throughout your career as a data professional.

**Citations:**

1. While drafting a reading, add citations to the table below.
2. Later, the PgM will ask writers to move citations to the cert’s citations tracker.

| **#** | **Title** | **Link** | |
| --- | --- | --- | --- |
|  | Python date & time conversion cheat sheet | [**https://dev.to/maikomiyazaki/python-date-time-conversion-cheatsheet-3m69**](https://dev.to/maikomiyazaki/python-date-time-conversion-cheatsheet-3m69) | |
|  | Python Cheat Sheet | [**https://cheatography.com/davechild/cheat-sheets/python/**](https://cheatography.com/davechild/cheat-sheets/python/) | |
|  | datetime — Basic date and time types | [**https://docs.python.org/3/library/datetime.html**](https://docs.python.org/3/library/datetime.html) | |