

Chapter 11 — Date and Time

Introduction

In addition to the standard data types of integer, float, Boolean, and string we often need to keep track and do mathematics with date and time values. Python has several modules to manipulate them, but the one we will be covering is `datetime`.

Objectives

Upon completion of this chapter's exercises, you should be able to:

- Explain Coordinated Universal Time (UTC) and the use of time zones.
- Calculate locale time from UTC and UTC from a locale's time.
- Express dates and times in the ISO Format and be able to explain all parts of an ISO date-time string.
- Discuss the difference between aware and naive date-times in Python.
- Use dates and times in a Python program.
- Change between an ISO string and a Python date time.

Prerequisites

This Chapter requires...

Defining a Date and Locale

Coordinated Universal Time (UTC): AKA: Universal Coordinated Time, Greenwich Mean Time, GMT, and Zulu Time. Notice that the abbreviation is UTC which is a different order than we write it in English. UTC is the planet wide time zone used by pilots, radio operators, governments, military, and computer systems to ensure that we are all referring to the same time regardless of where we are on the globe.

Coordinated Universal Time (UTC)	Concept
Standard global time defined by the International Telecommunications Union. It	

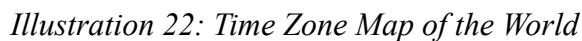
is used in communication, military, and other applications to ensure that we all are talking about the same time.

https://en.wikipedia.org/wiki/Coordinated_Universal_Time

People have become accustomed to 12:00AM being called "midnight". Time zones have been created as a way to define an offset from UTC, so that midnight is approximately 12:00AM and noon is approximately 12:00PM. There are twelve hours that are earlier in the day, east, of UCT and have an offset of +01:00 to +12:00 and twelve hours of time that are later in the day, west of UTC -01:00 to -12:00. The bands that define time-zone are not straight, along longitude, because of political, economic, and social convention.

Time Zone	Concept
There are 24 bands of time that follow the Sun as the Earth rotates. Noon, in a time zone, usually occurs when the Sun is highest in the sky. Time zones are defined as an offset from UTC. The offset is usually whole hours.	
https://en.wikipedia.org/wiki/List_of_UTC_time_offsets	

The Continental United States is covered by four time zones, Eastern Standard — EST (UTC-05:00), Central Standard — CST (UTC-06:00), Mountain Standard — MST (UTC-07:00), and Pacific Standard — PST (UTC-08:00). During the summer months, most locations reduce the UTC offset by one hour. This is known as "Daylight Savings Time". The US time zones during the summer are known as EDT (-04:00), CDT (-05:00), MDT (-06:00), and PDT (-07:00).



Free eBook Edition

Date part	Description
YYYY	4 digit year (2017 for example)
MM	2 digit month (01 to 12)
DD	2 digit day of the month (01-31)
T	The letter "T". Optional but required if time follows.
hh	2 digit hours (00 to 23) 00 is 12AM 12 is 12PM, 23 is 11PM. Required part of time.
mm	2 digit minutes (00 to 59). Required part of time.
ss	2 digit seconds (00 to 59). Optional.
.s	Decimal seconds. Optional.
Z	Time Zone. "Z" for UTC or +/-HH:MM offset from UTC. Optional.

Table 7: Parts of the ISO Date Format

ISO Date Format	Concept
<p>The International Standards Organization (ISO) has created a standard format for how to write dates. The format is YYYY-MM-DDThh:mm:ss.sZ. Where YYYY represents year, MM is month, DD is day of the month separated by dashes. If a time follows date, the letter "T" is followed by hours since mid-night, a colon, and minutes. If seconds are needed, they are added after another colon. Lastly an optional time-zone may be specified.</p> <p>https://www.w3.org/TR/NOTE-datetime</p>	

Python Dates

The datetime module includes classes called date and time that will store dates and times as separate values. The `date` and `time` are ideal dates and times. They represent dates and time separately and do not include the concept of time zone. They are useful for simple applications. For more complex or distributed applications, the `datetime` class may be better.

<code>datetime</code>	Module
-----------------------	--------

The datetime module actually consists of classes named: date, time, datetime (date and time together), timedelta (to work with differences between times), tzinfo (a abstract way to think about time zones), and timezone (to work with offset from UTC).

<https://docs.python.org/3/library/datetime.html>

`datetime.date`

`datetime.date(year, month, day)`

Class in datetime

A class to store simple dates using a proleptic Gregorian calendar. You may represent dates before 1582 and far into the future, even though the Gregorian calendar hadn't been created.

You may specify the year, month, and day number as arguments to build a specific date.

<https://docs.python.org/3/library/datetime.html#date-objects>

`datetime.date.today()`

Method in datetime.date

Returns a date object with today's date.

<https://docs.python.org/3/library/datetime.html#datetime.date.today>

```
1| import datetime
2| today = datetime.date.today()
3| print(today)
```

Will print the local date in ISO format YYYY-MM-DD.

```
4| import datetime
5| today = datetime.date.today()
6| birthdate = datetime.date(1973, 5, 9)    # MAY 9, 1973
7| age = today - birthdate
8| print("You are", age, "old")
```

You are 16036 days, 0:00:00 old

The subtraction of dates results in a special value known as a `timedelta`. You will see several uses for the `timedelta` in the remainder of this chapter. To get years we must convert the duration into days and then divide by the number of days in a year.

```
1| import datetime
2| today = datetime.date.today()
3| birthdate = datetime.date(1973, 5, 9)    # MAY 9, 1973
4| age = today - birthdate
5| print("You are", age.days//365.25, "old")
```

Python Date and Time Together

The `datetime` module includes an aptly named class called `datetime`. The `datetime.datetime` class supports dates and times together with their time zone. In many programs we need to know both, together, to track and time-stamp orders, transactions, logins, and lots of other things.

Date and times stored in the `datetime.datetime` class fall into two groups: 1) aware, and 2) naive. Aware date-times include time zone information, and naive date times do not. You should not mix aware and naive date times in a system as users from various time zones may have inconsistent results. It is for this case that all the examples in this discussion will be aware of the timezone.

<code>datetime.datetime</code>	Class of datetime
The <code>datetime.datetime</code> class is used to store dates, times, and time-zones together in one object. A <code>datetime.datetime</code> may either be aware (knows the time-zone for the date and time) or naive (does not have time-zone information)	
https://docs.python.org/3/library/datetime.html#datetime-objects	

We can get the current UTC time (in an aware datetime) by telling the `now()` method to return a date in the UTC timezone. In the example, below, you will see the timezone `+00:00` on the end of the time.

<code>datetime.datetime.now()</code> <code>datetime.datetime.now(timezone)</code>	Method of datetime.datetime
The <code>now()</code> method returns the local date and time without time-zone information. You may pass an optional <code>datetime.timezone</code> object to create an aware datetime with the time and date.	

https://docs.python.org/3/library/datetime.html#datetime.datetime.now	
---	--

<code>datetime.timezone</code> <code>datetime.timezone(timedelta)</code>	Object of datetime
---	--------------------

<code>datetime.timezone.utc</code>	Object of datetime.timezone
------------------------------------	-----------------------------

```
1| import datetime
2| dt = datetime.datetime.now(datetime.timezone.utc)
3| print("UTC",dt)
```

```
UTC 2017-04-06 19:41:26.888999+00:00
```

To create aware datetimes in other time zones we need to create a timezone by creating a `timedelta` object with the offset from UTC. With a timezone created you can then use it in the `now()` method.

```
4| import datetime
5| tz = datetime.timezone(datetime.timedelta(hours=-4))
6| print(tz)
7| now = datetime.datetime.now(tz)
8| print("EDT",now)
```

```
UTC-04:00
EDT 2017-04-06 15:43:48.242000-04:00
```

To make the timezone automatic (to use the settings on the computer), I have written a helper function called `myTimeZone`. It takes the UTC time and the computer time, subtracts the two to calculate the offset in seconds, and creates a timezone with that offset from UTC. You can use the function as the argument to the `now()` function to create an aware datetime.

```
1| def myTimeZone():
2|     ## return timezone to use in datetime for my local time
3|     import datetime
4|     return datetime.timezone(datetime.timedelta(
5|         seconds = round((datetime.datetime.now() -
```



```
6| datetime.datetime.utcnow()).total_seconds()))
```

Using the functions above, the code below returns the time in the current time zone.

```
1| import datetime
2| now = datetime.datetime.now(myTimeZone())
3| print(now)
```

```
2017-04-06 15:55:08.302000-04:00
```

The datetime object will also allow you to get the various parts of a datetime. See the documentation for a full example.

```
1| import datetime
2| d = datetime.datetime.now()
3| print(d.year,d.month, d.day)
4| # find the next valentines day
5| # get valentines day this year
6| valentines = datetime.datetime(d.year, 2, 14)
7| # has it passed?
8| if valentines < d:
9|     # if it was earlier then find it next year
10|     valentines = datetime.datetime(d.year+1, 2, 14)
11| print(valentines)
12| print("valentines day is", (valentines-d).days,"days away.")
```

Converting a String to a Date or Datetime

Often you may be asked to convert a string (that the user enters or otherwise) into a datetime or a date. The `strptime` method of `datetime` will do just that. A date format is specified as the second argument of `strptime` that tells the method how to break the string apart as a date and time. The following table shows some common strip strings. If date is stripped without time, the time will be set at 00:00. If the time is stripped without a date, the date will be set to 1900-01-01. See the documentation for complete details.¹⁰

¹⁰ <https://docs.python.org/3/library/datetime.html#strptime-strptime-behavior>



Format	Expected Date and Time
%m/%d/%y	Two digit month, day, and year with slashes. Like: 12/25/17, 2/14/16, and 3/7/18
%m/%d/%Y	Two digit month and day, four digit year, separated with slashes. Like: 01/01/2000, 02/02/2019, and 5/20/2022
%Y-%m-%d	ISO date format with four digit year, two digit month, and two digit day. Like: 2013-09-07, 2020-02-20, and 2017-03-14
%H:%M:%S	Two digit hour, minute, and seconds with a colon separator. Like: 12:34:56, 00:00:00, and 23:59:59
%H%M	Two digit hours (24 hour clock), two digit minute, with no separator. Like 0839, 1456, 2359, and 0000
%Y-%m-%d %H:%M %z	YYYY-MM-DD HH:MM -ZZZZ. Like 1941-12-07 07:55 -1000

Table 8: Sample "strptime" Formats

```
1| import datetime
2| bicen = datetime.datetime.strptime("1976-07-04", "%Y-%m-%d")
3| print(bicen)
```

NOTE: To convert a datetime into a plain date you may use the date() method.

```
1| import datetime
2| bicen = datetime.datetime.strptime("1976-07-04", "%Y-%m-%d").date()
3| print(bicen)
```

Fancy example with stripping out an aware date and then calculating how long an event was from here and now.

```
1| import datetime
2| def myTimeZone():
3|     ## return timezone to use in datetime for my local time
4|     import datetime
5|     return datetime.timezone(datetime.timedelta(
6|         seconds = round((datetime.datetime.now() -
7|         datetime.datetime.utcnow()).total_seconds())))
```

```
5| pearlharbor = "1941-12-07 07:55 -1000"
6| dl11 = datetime.datetime.strptime(pearlharbor,"%Y-%m-%d %H:%M
   | %z")
7| print(dl11)
8|
9| here = datetime.datetime.now(myTimeZone())
10|
11| ago = here - dl11
12|
13| print(ago)
```

Summary

Goes here

Important Terms

- Coordinated Universal Time
- GMT
- Greenwich Mean Time
- HH:MM:SS
- ISO
- Longitude
- Time Zone
- UCT
- YY-MM-DD
- aware
- datetime
- naive
- strptime
- timedelta

Exercises

Here

Word Search



o o a e o m r i u n . o g s e g e i l . i i i o e y
- a s e i a m . e t l t t d e e m h h t o t m p e o
c o o r d i n a t e d . u n i v e r s a l . t i m e
z m a d n e s t t n t m i t a o i n s t t : n e e .
m j a c s t . o c s i . n i i u n x a u o l r v u t
a l n d w u c t . c t . r m s t r p t i n e d . c :
c m r a w a r e e t c l g e s t m s . u g r e l a o
o y n n y v e d y e s a r d e s k s o w e m a e b b
t y t t j l n a i v e i e e . m c h n : h o d l c m
i - m . e r t d . n e h e l v t i m e . z o n e n i
m m t e l i i : l h k l n t n g m m g h l m t e z n
h m o h e a h h d n w j w a i t y s i c u a . o e i
o - n i n a i h t o a d i d d o l m . c : o - . s .
t d t e m l j : e y e o c e n t o o r v t r n a v t
o d e g m t e m k o a u h t t j . m i i r t o . e m
l m e : a e h m n c l d . c . l h e y g i c z e n e
a o d o q l e : y a o z m t m w d r i i v i e a m u
p d n i q n n s y . n t e n c e a s m d n s i t w :
r o e m l s t s a i g n a a d t t r c x s o n t r t
h m e h r a i i c m i i n c g e e h d t i i a r e s
c e y l r w - p t e t x . i o r t r i . l w d i u i
c . e f w : f w y c u d t . t o i y g i : n l i g w
- x i s t e v y l m d . i i r g m t t d t i o n r e
o r . c o n e a e q e s m . t g e p i h t e y m i a
d g i e n t e c e e - t e o t e n d - d t i m a t n
e i t n i - : r v e a e i m t x u e a h a c t m o a

Coordinated Universal Time, GMT, Greenwich Mean Time, HH:MM:SS, ISO, Longitude, Time Zone, UCT, YY-MM-DD, aware, datetime, naive, strptime, timedelta

References

Heitordop. (2015-08-07). Standard Time Zones of the World. Public Domain Image. Retrieved 2019-01-08 from https://commons.wikimedia.org/wiki/File:Standard_World_Time_Zones.png.