SELECTED TOPICS IN ENGINEERING

INTR. TO PROG. FOR DATA SCIENCE
ENGR 350

Tuesday-Thursday 10:00-12:45
ENG B05
2019 Summer

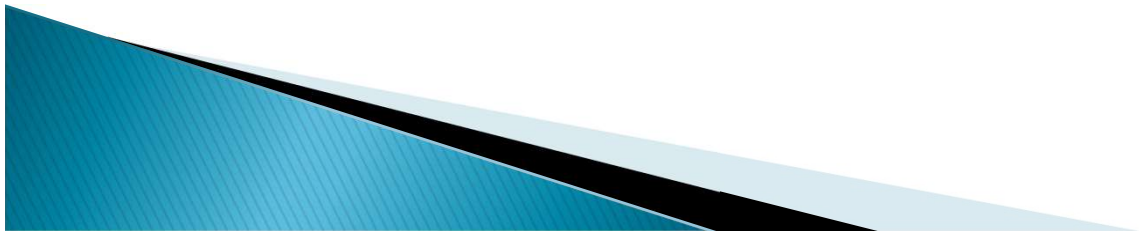
Dr. Banu Yobaş

Info on Date and Time

- DST is Daylight Saving Time
- The *epoch* is the point where the time starts, and is *platform dependent*.

For Unix, the epoch is January 1, 1970, 00:00:00 (UTC).

- Note that even though the time is always returned as a floating point number, not all systems provide time with a better precision than 1 second.

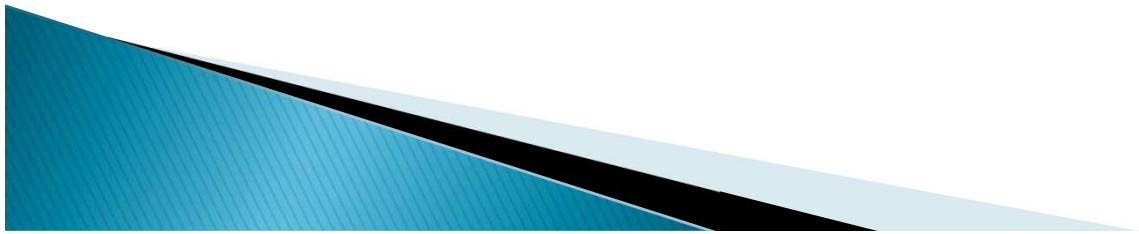


Seconds

- The precision of the various real-time functions may be less than suggested by the units in which their value or argument is expressed.

E.g. on most Unix systems, the clock “ticks” only 50 or 100 times a second.

- **Leap seconds**: Unlike **leap** years, **leap seconds** are not a regular occurrence. Instead, they are decreed by the International Earth Rotation and Reference Systems Service, or IERS, in Paris, which measures Earth's rotation and compares it with the time kept by atomic clocks



time module

- time.time()

1532777125.345091

Return the time in seconds since the epoch as a floating point number. The specific date of the epoch and the handling of leap seconds is platform dependent. On Windows and most Unix systems, the epoch is January 1, 1970, 00:00:00 (UTC) and leap seconds are not counted towards the time in seconds since the epoch.

To work with Epoch Seconds, you need to use the time module

The number returned by time() may be converted into a more common time format (i.e. year, month, day, hour, etc…) in UTC by passing it to gmtime() function or in local time by passing it to the localtime() function. In both cases a struct_time object is returned, from which the components of the calendar date may be accessed as attributes.

Time structure

The time value is a sequence of 9 integers, a named tuple: Values can be accessed by index and by attribute name

Index	Attribute	Values
0	tm_year	(for example, 1993)
1	tm_mon	range [1, 12]
2	tm_mday	range [1, 31]
3	tm_hour	range [0, 23]
4	tm_min	range [0, 59]
5	tm_sec	range [0, 61]; see **(2)** in strftime() description
6	tm_wday	range [0, 6], Monday is 0
7	tm_yday	range [1, 366]
8	tm_isdst	0, 1 or -1; see below
N/A	tm_zone	abbreviation of timezone name
N/A	tm_gmtoff	offset east of UTC in seconds

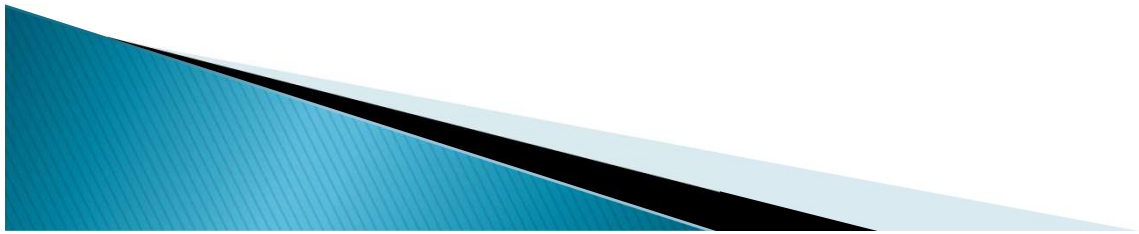
tm_isdst may be set to 1 when daylight savings time is in effect, and 0 when it is not.
A value of -1 indicates that this is not known, and will usually result in the correct state being filled in.

time module

- To find out what the epoch is on a given platform : **time.gmtime(0)**

time.struct_time(tm_year=1970, tm_mon=1, tm_mday=1, tm_hour=0, tm_min=0, tm_sec=0, tm_wday=3, tm_yday=1, tm_isdst=0)

time.asctime([t]) : Convert a tuple or struct_time representing a time as returned by gmtime() or localtime() to a string of the following form: 'Sun Jun 20 23:21:05 1993'. If t is not provided, the current time as returned by localtime() is used.



time module

- time.localtime()

#time.struct_time(tm_year=2018, tm_mon=7, tm_mday=28, tm_hour=14, tm_min=44, tm_sec=47, tm_wday=5, tm_yday=209, tm_isdst=0)

- time.mktime()
- time.sleep(*secs*)
- time.strftime("%a, %d %b %Y %H:%M:%S +0000", time.gmtime())
- time.time_ns() → int Similar to time() but returns time as an integer number of nanoseconds since the epoch.

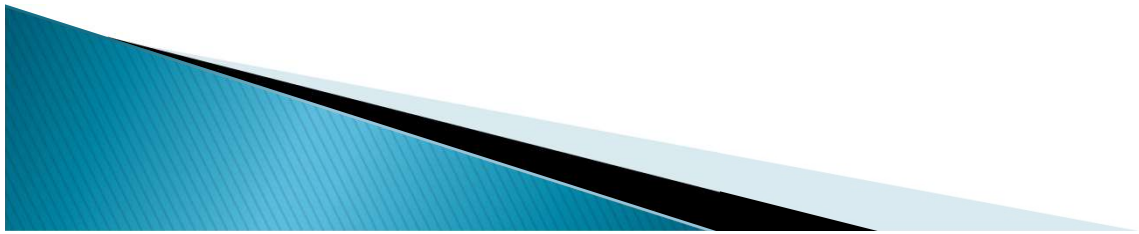
New in version 3.7.

Directive	Meaning	Notes
%a	Locale's abbreviated weekday name.	
%A	Locale's full weekday name.	
%b	Locale's abbreviated month name.	
%B	Locale's full month name.	
%c	Locale's appropriate date and time representation.	
%d	Day of the month as a decimal number [01,31].	
%H	Hour (24-hour clock) as a decimal number [00,23].	
%I	Hour (12-hour clock) as a decimal number [01,12].	
%j	Day of the year as a decimal number [001,366].	
%m	Month as a decimal number [01,12].	
%M	Minute as a decimal number [00,59].	
%p	Locale's equivalent of either AM or PM.	(1)
%S	Second as a decimal number [00,61].	(2)
%U	Week number of the year (Sunday as the first day of the week) as a decimal number [00,53]. All days in a new year preceding the first Sunday are considered to be in week 0.	(3)
%w	Weekday as a decimal number [0(Sunday),6].	
%W	Week number of the year (Monday as the first day of the week) as a decimal number [00,53]. All days in a new year preceding the first Monday are considered to be in week 0.	(3)
%x	Locale's appropriate date representation.	
%X	Locale's appropriate time representation.	

datetime module

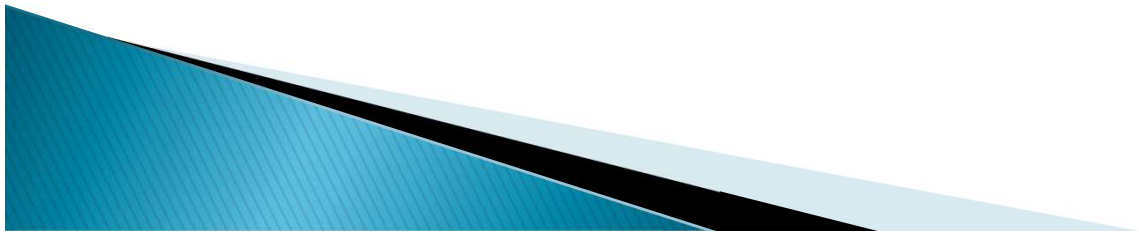
- datetime.date(*year*, *month*, *day*)
- date.today()
- date.fromisoformat(*date_string*)

Return a date corresponding to a *date_string*
'YYYY-MM-DD'



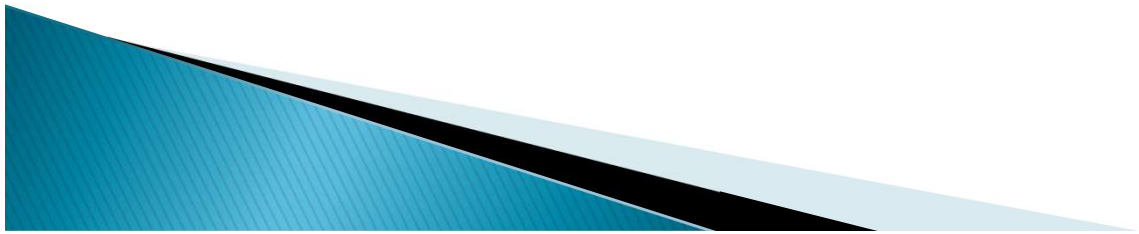
Date <-> String conversions

- **strftime(format)** method, to create a string representing the time under the control of an explicit format string.
- datetime.strptime() class method creates a datetime object from a string representing a date and time and a corresponding format string.



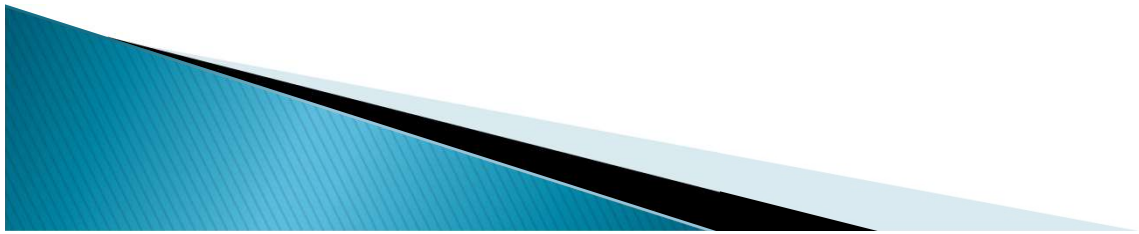
calendar module

- Check the code named basic_datetime.py



Date Time

```
from datetime import date
date.today()
datetime.date(2018, 7, 16)
```



Date & time

- The parameters are optional, so if you call Time with no arguments, you get the default values.

```
>>>>> time.print_time()
00:00:00
If you provide one argument, it overrides hour:
>>> time = Time (9)
>>> time.print_time()
09:00:00
If you provide two arguments, they override hour and minute.
>>> time = Time(9, 45)
>>> time.print_time()
09:45:00
> time = Time()
```