time class

A time object represents a (local) time of day, independent of any particular day.

class datetime.time(hour=0, minute=0, second=0, microsecond=0)

0 <= hour < 24,
0 <= minute < 60,
0 <= second < 60,
0 <= microsecond < 1000000,

If an argument outside those ranges is given, ValueError is raised.

```
>>> import datetime
>>> t1=datetime.time()
>>> t1.hour
0
>>> t1.minute
0
>>> t1.second
0
>>> print(t1)
00:00:00
>>> t2=datetime.time(4,30,20)
>>> print(t2)
04:30:20
```

datetime class or object

A datetime object is a single object containing all the information from a date object and a time object.

class datetime.datetime(year, month, day, hour=0, minute=0, secon
d=0, microsecond=0)

Example

```
>>> d1=datetime.datetime(2024,3,18)
>>> print(d1)
2024-03-18 00:00:00
>>> print(d1.year)
2024
>>> print(d1.month)
3
>>> print(d1.day)
18
>>> print(d1.hour)
0
>>> print(d1.minute)
0
>>> print(d1.second)
0
>>> d2=datetime.datetime.today()
>>> print(d2)
2024-03-18 07:56:46.855819
>>> d3=d2.date()
>>> t2=d2.time()
>>> print(d3)
2024-03-18
>>> print(t2)
07:56:46.855819
```

timedelta Objects

A timedelta object represents a duration, the difference between two datetime or date instances.

class datetime.**timedelta**(days=0, seconds=0, microseconds=0, millise
conds=0, minutes=0, hours=0, weeks=0)

All arguments are optional and default to 0. Arguments may be integers or floats, and may be positive or negative.

Only days, seconds and microseconds are stored internally.

Arguments are converted to those units:

A millisecond is converted to 1000 microseconds.
A minute is converted to 60 seconds.
An hour is converted to 3600 seconds.
A week is converted to 7 days.

```
>>> days=datetime.timedelta(days=2)
>>> date1=date1+days
>>> print(date1)
2024-03-20
>>> date1=date1-days
>>> print(date1)
2024-03-18
month=datetime.timedelta(weeks=4)
>>> date1=date1+month
>>> print(date1)
2024-04-15
>>> date1=date1-month
>>> print(date1)
2024-03-18
>>> year=datetime.timedelta(weeks=52)
```

```
>>> date1=date1+year
>>> print(date1)
2025-03-17
>>> date1=date1-year
>>> print(date1)
2024-03-18
```

date.strftime(*format*)
Return a string representing the date, controlled by an explicit format string.

Directive	Meaning	Example
%a	Weekday as locale's abbreviated name.	Sun, Mon, …, Sat (en_US); So, Mo, …, Sa (de_DE)
%A	Weekday as locale's full name.	Sunday, Monday, …, Saturday (en_US); Sonntag, Montag, …, Samstag (de_DE)
%w	Weekday as a decimal number, where 0 is Sunday and 6 is Saturday.	0, 1, …, 6
%d	Day of the month as a zero-padded decimal number.	01, 02, …, 31

%b	Month as locale's abbreviated name.	Jan, Feb, …, Dec (en_US); Jan, Feb, …, Dez (de_DE)
%B	Month as locale's full name.	January, February, …, December (en_US); Januar, Februar, …, Dezember (de_DE)
%m	Month as a zero-padded decimal number.	01, 02, …, 12
%y	Year without century as a zero-padded decimal number.	00, 01, …, 99
%Y	Year with century as a decimal number.	0001, 0002, …, 2013, 2014, …, 9998, 9999
%H	Hour (24-hour clock) as a zero-padded decimal number.	00, 01, …, 23
%I	Hour (12-hour clock) as a zero-padded decimal number.	01, 02, …, 12
%p	Locale's equivalent of either AM or PM.	AM, PM (en_US); am, pm (de_DE)
%M	Minute as a zero-padded decimal number.	00, 01, …, 59
%S	Second as a zero-padded decimal number.	00, 01, …, 59

%f	Microsecond as a decimal number, zero-padded to 6 digits.	000000, 000001, …, 999999
%z	UTC offset in the form ±HHMM[SS[.ffffff]] (empty string if the object is naive).	(empty), +0000, -0400, +1030, +063415, -030712.345216
%Z	Time zone name (empty string if the object is naive).	(empty), UTC, GMT
%j	Day of the year as a zero-padded decimal number.	001, 002, …, 366
%U	Week number of the year (Sunday as the first day of the week) as a zero-padded decimal number. All days in a new year preceding the first Sunday are considered to be in week 0.	00, 01, …, 53
%W	Week number of the year (Monday as the first day of the week) as a zero-padded decimal number. All days in a new year preceding the first Monday are considered to be in week 0.	00, 01, …, 53
%c	Locale's appropriate date and time representation.	Tue Aug 16 21:30:00 1988 (en_US); Di 16 Aug 21:30:00 1988 (de_DE)

%x	Locale’s appropriate date representation.	08/16/88 (None); 08/16/1988 (en_US); 16.08.1988 (de_DE)
%X	Locale’s appropriate time representation.	21:30:00 (en_US); 21:30:00 (de_DE)
%%	A literal '%' character.	%

Example:

```
import datetime

dt=datetime.datetime.today()
print(dt)
print(dt.strftime("%a"))
print(dt.strftime("%A"))
print(dt.strftime("%w"))
print(dt.strftime("%d"))
print(dt.strftime("%A %d"))
print(dt.strftime("%b"))
print(dt.strftime("%B"))
print(dt.strftime("%A %d %B"))
print(dt.strftime("%m"))
print(dt.strftime("%d-%m-%y"))
print(dt.strftime("%d-%m-%Y"))
print(dt.strftime("%A %d %B %Y"))
print(dt.strftime("%H"))
print(dt.strftime("%S"))
print(dt.strftime("%M"))
print(dt.strftime("%H:%M:%S %p"))
```

```
print(dt.strftime("%j"))
print(dt.strftime("%U"))
print(dt.strftime("%c"))
print(dt.strftime("%x"))
print(dt.strftime("%X"))
```

Output

```
2024-03-18 08:26:00.382929
Mon
Monday
1
18
Monday 18
Mar
March
Monday 18 March
03
18-03-24
18-03-2024
Monday 18 March 2024
08
00
26
08:26:00 AM
078
11
Mon Mar 18 08:26:00 2024
03/18/24
08:26:00
```

Calendar Module

This module allows you to output calendars like the Unix cal program, and provides additional useful functions related to the calendar. By default, these calendars have Monday as the first day of the week, and Sunday as the last (the European convention).

calendar.month(*theyear*, *themonth*, *w=0*, *l=0*)
Returns a month's calendar in a multi-line string using the formatmonth() of the TextCalendar class.

```
>>> import calendar
>>> print(calendar.month(2024,3))
    March 2024
Mo Tu We Th Fr Sa Su
            1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31
```

calendar.calendar(*year*, *w=2*, *l=1*, *c=6*, *m=3*)
Returns a 3-column calendar for an entire year as a multi-line string using the formatyear() of the TextCalendar class.

```
>>> print(calendar.calendar(2024))
                                  2024

      January                   February                  March
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7            1  2  3  4               1  2  3
 8  9 10 11 12 13 14       5  6  7  8  9 10 11       4  5  6  7  8  9 10
15 16 17 18 19 20 21      12 13 14 15 16 17 18      11 12 13 14 15 16 17
22 23 24 25 26 27 28      19 20 21 22 23 24 25      18 19 20 21 22 23 24
```

29 30 31 26 27 28 29 25 26 27 28 29 30 31

***class* calendar.TextCalendar(*firstweekday=0*)**

This class can be used to generate plain text calendars.

Example

```
import calendar
tc=calendar.TextCalendar()
month=tc.formatmonth(2024,3)
print(month)
year=tc.formatyear(2024)
print(year)

f=open("year2024.txt","w")
f.write(year)
f.close()
```

Output

```
    March 2024
Mo Tu We Th Fr Sa Su
           1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31


                          2024

      January               February              March
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su
```

```
 1  2  3  4  5  6  7              1  2  3  4                 1  2  3
 8  9 10 11 12 13 14       5  6  7  8  9 10 11       4  5  6  7  8  9 10
15 16 17 18 19 20 21      12 13 14 15 16 17 18      11 12 13 14 15 16 17
22 23 24 25 26 27 28      19 20 21 22 23 24 25      18 19 20 21 22 23 24
29 30 31                  26 27 28 29               25 26 27 28 29 30 31
```

***class* calendar.HTMLCalendar(*firstweekday=0*)**
This class can be used to generate HTML calendars.

Example:

```
import calendar

htmlc=calendar.HTMLCalendar()

month=htmlc.formatmonth(2024,3)
print(month)
f=open("month.html","w")
f.write(month)
f.close()

year=htmlc.formatyear(2024)
print(year)
f=open("year.html","w")
f.write(year)
f.close()
```

Output

```
<table border="0" cellpadding="0" cellspacing="0" class="month">
<tr><th colspan="7" class="month">March 2024</th></tr>
<tr><th class="mon">Mon</th><th class="tue">Tue</th><th
class="wed">Wed</th><th class="thu">Thu</th><th
```

class="fri">Fri</th><th class="sat">Sat</th><th
class="sun">Sun</th></tr>
<tr><td class="noday"> </td><td
class="noday"> </td><td class="noday"> </td><td
class="noday"> </td><td class="fri">1</td><td
class="sat">2</td><td class="sun">3</td></tr>
<tr><td class="mon">4</td><td class="tue">5</td><td
class="wed">6</td><td class="thu">7</td><td class="fri">8</td><td
class="sat">9</td><td class="sun">10</td></tr>

Python Database Communication (PDBC)

Every application required to store/save data permanently.
This data can be stored permanently using two approaches

1. Files
2. Database

Limitations of file management system

1. Files cannot hold large amount of data
2. Files are not secured because these are managed by OS
3. Files does not provides Query language