`calendar` — General calendar-related functions

Source code: Lib/calendar.py

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). Use `setfirstweekday()` to set the first day of the week to Sunday (6) or to any other weekday. Parameters that specify dates are given as integers. For related functionality, see also the `datetime` and `time` modules.

The functions and classes defined in this module use an idealized calendar, the current Gregorian calendar extended indefinitely in both directions. This matches the definition of the "proleptic Gregorian" calendar in Dershowitz and Reingold's book "Calendrical Calculations", where it's the base calendar for all computations. Zero and negative years are interpreted as prescribed by the ISO 8601 standard. Year 0 is 1 BC, year -1 is 2 BC, and so on.

class `calendar.`**`Calendar`**(*firstweekday=0*)

Creates a `Calendar` object. *firstweekday* is an integer specifying the first day of the week. `0` is Monday (the default), `6` is Sunday.

A `Calendar` object provides several methods that can be used for preparing the calendar data for formatting. This class doesn't do any formatting itself. This is the job of subclasses.

`Calendar` instances have the following methods:

`iterweekdays`()

Return an iterator for the week day numbers that will be used for one week. The first value from the iterator will be the same as the value of the `firstweekday` property.

`itermonthdates`(*year*, *month*)

Return an iterator for the month *month* (1–12) in the year *year*. This iterator will return all days (as `datetime.date` objects) for the month and all days before the start of the month or after the end of the month that are required to get a complete week.

`itermonthdays`(*year*, *month*)

Return an iterator for the month *month* in the year *year* similar to `itermonthdates()`, but not restricted by the `datetime.date` range. Days returned will simply be day of the month numbers. For the days outside of the specified month, the day number is `0`.

`itermonthdays2`(*year*, *month*)

Return an iterator for the month *month* in the year *year* similar to `itermonthdates()`, but not restricted by the `datetime.date` range. Days

returned will be tuples consisting of a day of the month number and a week day number.

itermonthdays3(*year*, *month*)

Return an iterator for the month *month* in the year *year* similar to `itermonthdates()`, but not restricted by the `datetime.date` range. Days returned will be tuples consisting of a year, a month and a day of the month numbers.

New in version 3.7.

itermonthdays4(*year*, *month*)

Return an iterator for the month *month* in the year *year* similar to `itermonthdates()`, but not restricted by the `datetime.date` range. Days returned will be tuples consisting of a year, a month, a day of the month, and a day of the week numbers.

New in version 3.7.

monthdatescalendar(*year*, *month*)

Return a list of the weeks in the month *month* of the *year* as full weeks. Weeks are lists of seven `datetime.date` objects.

monthdays2calendar(*year*, *month*)

Return a list of the weeks in the month *month* of the *year* as full weeks. Weeks are lists of seven tuples of day numbers and weekday numbers.

monthdayscalendar(*year*, *month*)

Return a list of the weeks in the month *month* of the *year* as full weeks. Weeks are lists of seven day numbers.

yeardatescalendar(*year*, *width=3*)

Return the data for the specified year ready for formatting. The return value is a list of month rows. Each month row contains up to *width* months (defaulting to 3). Each month contains between 4 and 6 weeks and each week contains 1–7 days. Days are `datetime.date` objects.

yeardays2calendar(*year*, *width=3*)

Return the data for the specified year ready for formatting (similar to `yeardatescalendar()`). Entries in the week lists are tuples of day numbers and weekday numbers. Day numbers outside this month are zero.

yeardayscalendar(*year*, *width=3*)

Return the data for the specified year ready for formatting (similar to `yeardatescalendar()`). Entries in the week lists are day numbers. Day numbers outside this month are zero.

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

This class can be used to generate plain text calendars.

`TextCalendar` instances have the following methods:

formatmonth(*theyear*, *themonth*, *w=0*, *l=0*)

Return a month's calendar in a multi-line string. If *w* is provided, it specifies the width of the date columns, which are centered. If *l* is given, it specifies the number of lines that each week will use. Depends on the first weekday as specified in the constructor or set by the `setfirstweekday()` method.

prmonth(*theyear*, *themonth*, *w=0*, *l=0*)

Print a month's calendar as returned by `formatmonth()`.

formatyear(*theyear*, *w=2*, *l=1*, *c=6*, *m=3*)

Return a *m*-column calendar for an entire year as a multi-line string. Optional parameters *w*, *l*, and *c* are for date column width, lines per week, and number of spaces between month columns, respectively. Depends on the first weekday as specified in the constructor or set by the `setfirstweekday()` method. The earliest year for which a calendar can be generated is platform-dependent.

pryear(*theyear*, *w=2*, *l=1*, *c=6*, *m=3*)

Print the calendar for an entire year as returned by `formatyear()`.

class `calendar.`**HTMLCalendar**(*firstweekday=0*)

This class can be used to generate HTML calendars.

`HTMLCalendar` instances have the following methods:

formatmonth(*theyear*, *themonth*, *withyear=True*)

Return a month's calendar as an HTML table. If *withyear* is true the year will be included in the header, otherwise just the month name will be used.

formatyear(*theyear*, *width=3*)

Return a year's calendar as an HTML table. *width* (defaulting to 3) specifies the number of months per row.

formatyearpage(*theyear*, *width=3*, *css='calendar.css'*, *encoding=None*)

Return a year's calendar as a complete HTML page. *width* (defaulting to 3) specifies the number of months per row. *css* is the name for the cascading style sheet to be used. `None` can be passed if no style sheet should be used. *encoding* specifies the encoding to be used for the output (defaulting to the system default encoding).

`HTMLCalendar` has the following attributes you can override to customize the CSS classes used by the calendar:

cssclasses

A list of CSS classes used for each weekday. The default class list is:

```
cssclasses = ["mon", "tue", "wed", "thu", "fri", "sat", "sun"]
```

more styles can be added for each day:

```
cssclasses = ["mon text-bold", "tue", "wed", "thu", "fri", "sat",
```

Note that the length of this list must be seven items.

cssclass_noday

The CSS class for a weekday occurring in the previous or coming month.

New in version 3.7.

cssclasses_weekday_head

A list of CSS classes used for weekday names in the header row. The default is the same as `cssclasses`.

New in version 3.7.

cssclass_month_head

The month's head CSS class (used by `formatmonthname()`). The default value is `"month"`.

New in version 3.7.

cssclass_month

The CSS class for the whole month's table (used by `formatmonth()`). The default value is `"month"`.

New in version 3.7.

cssclass_year

The CSS class for the whole year's table of tables (used by `formatyear()`). The default value is `"year"`.

New in version 3.7.

cssclass_year_head

The CSS class for the table head for the whole year (used by `formatyear()`). The default value is `"year"`.

New in version 3.7.

Note that although the naming for the above described class attributes is singular (e.g. `cssclass_month cssclass_noday`), one can replace the single CSS class with a space separated list of CSS classes, for example:

```
"text-bold text-red"
```

Here is an example how `HTMLCalendar` can be customized:

```
class CustomHTMLCal(calendar.HTMLCalendar):
    cssclasses = [style + " text-nowrap" for style in
                  calendar.HTMLCalendar.cssclasses]
```

```
    cssclass_month_head = "text-center month-head"
    cssclass_month = "text-center month"
    cssclass_year = "text-italic lead"
```

class `calendar.`**`LocaleTextCalendar`**(*firstweekday=0*, *locale=None*)

This subclass of `TextCalendar` can be passed a locale name in the constructor and will return month and weekday names in the specified locale. If this locale includes an encoding all strings containing month and weekday names will be returned as unicode.

class `calendar.`**`LocaleHTMLCalendar`**(*firstweekday=0*, *locale=None*)

This subclass of `HTMLCalendar` can be passed a locale name in the constructor and will return month and weekday names in the specified locale. If this locale includes an encoding all strings containing month and weekday names will be returned as unicode.

Note: The `formatweekday()` and `formatmonthname()` methods of these two classes temporarily change the current locale to the given *locale*. Because the current locale is a process-wide setting, they are not thread-safe.

For simple text calendars this module provides the following functions.

`calendar.`**`setfirstweekday`**(*weekday*)

Sets the weekday (`0` is Monday, `6` is Sunday) to start each week. The values `MONDAY`, `TUESDAY`, `WEDNESDAY`, `THURSDAY`, `FRIDAY`, `SATURDAY`, and `SUNDAY` are provided for convenience. For example, to set the first weekday to Sunday:

```
import calendar
calendar.setfirstweekday(calendar.SUNDAY)
```

`calendar.`**`firstweekday`**()

Returns the current setting for the weekday to start each week.

`calendar.`**`isleap`**(*year*)

Returns `True` if *year* is a leap year, otherwise `False`.

`calendar.`**`leapdays`**(*y1*, *y2*)

Returns the number of leap years in the range from *y1* to *y2* (exclusive), where *y1* and *y2* are years.

This function works for ranges spanning a century change.

`calendar.`**`weekday`**(*year*, *month*, *day*)

Returns the day of the week (`0` is Monday) for *year* (`1970`–…), *month* (`1`–`12`), *day* (`1`–`31`).

`calendar.`**`weekheader`**(*n*)

Return a header containing abbreviated weekday names. *n* specifies the width in characters for one weekday.

`calendar.`**`monthrange`**(*year*, *month*)

Returns weekday of first day of the month and number of days in month, for the specified *year* and *month*.

`calendar.`**`monthcalendar`**(*year*, *month*)

Returns a matrix representing a month's calendar. Each row represents a week; days outside of the month a represented by zeros. Each week begins with Monday unless set by `setfirstweekday()`.

`calendar.`**`prmonth`**(*theyear*, *themonth*, *w=0*, *l=0*)

Prints a month's calendar as returned by `month()`.

`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.

`calendar.`**`prcal`**(*year*, *w=0*, *l=0*, *c=6*, *m=3*)

Prints the calendar for an entire year as returned by `calendar()`.

`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.

`calendar.`**`timegm`**(*tuple*)

An unrelated but handy function that takes a time tuple such as returned by the `gmtime()` function in the `time` module, and returns the corresponding Unix timestamp value, assuming an epoch of 1970, and the POSIX encoding. In fact, `time.gmtime()` and `timegm()` are each others' inverse.

The `calendar` module exports the following data attributes:

`calendar.`**`day_name`**

An array that represents the days of the week in the current locale.

`calendar.`**`day_abbr`**

An array that represents the abbreviated days of the week in the current locale.

`calendar.`**`month_name`**

An array that represents the months of the year in the current locale. This follows normal convention of January being month number 1, so it has a length of 13 and `month_name[0]` is the empty string.

`calendar.`**`month_abbr`**

An array that represents the abbreviated months of the year in the current locale. This follows normal convention of January being month number 1, so it has a length of 13 and `month_abbr[0]` is the empty string.

See also:

Module `datetime`
: Object-oriented interface to dates and times with similar functionality to the `time` module.

Module `time`
: Low-level time related functions.