The *calendar* module

In addition to the *datetime* and *time* modules, the Python standard library provides a module called *calendar* which, as the name suggests, offers calendar-related functions.

One of them is of course displaying the calendar. It's important that the days of the week are displayed from Monday to Sunday, and each day of the week has its representation in the form of an integer:

Day of the week	Integer value	Constant
Monday	0	$calendar.MONDAY$
Tuesday	1	$calendar.TUESDAY$
Wednesday	2	$calendar.WEDNESDAY$
Thursday	3	$calendar.THURSDAY$
Friday	4	$calendar.FRIDAY$
Saturday	5	$calendar.SATURDAY$
Sunday	6	$calendar.SUNDAY$

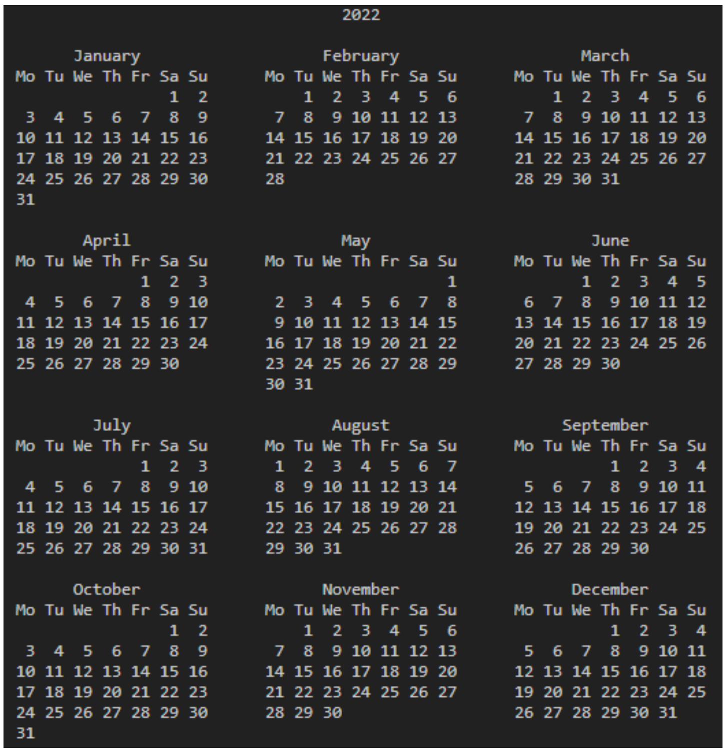
The table above shows the representation of the days of the week in the calendar module. The first day of the week (Monday) is represented by the value 0 and the $calendar.MONDAY$ constant, while the last day of the week (Sunday) is represented by the value 6 and the $calendar.SUNDAY$ constant.

For months, integer values are indexed from 1, i.e., January is represented by 1, and December by 12. Unfortunately, there aren't constants that express the months.

You will start your adventure with the calendar module with a simple function called calendar, which allows you to **display the calendar for the whole year**.

```python
import calendar
print(calendar.calendar(2022))
```

Expected output:



```
                                  2022

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

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

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

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

The result displayed is similar to the result of the *cal* command available in Unix. If you want to change the default calendar formatting, you can use the following parameters:

- *w* – date column width (default 2)
- *l* – number of lines per week (default 1)
- *c* – number of spaces between month columns (default 6)
- *m* – number of columns (default 3)

The calendar function requires you to specify the year, while the other parameters responsible for formatting are optional. You are encouraged to try these parameters yourself.

A good alternative to the above function is the function called *prcal*, which also takes the same parameters as the calendar function, but doesn't require the use of the print function to display the calendar. Its use looks like this:

The *calendar* module has a function called *month*, which allows you to display a calendar for a specific month. Its use is really simple, you just need to specify the year and month - check out the code here.

```python
import calendar
print(calendar.month(2022, 4))
```

The example displays the calendar for November 2020. As in the *calendar* function, you can change the default formatting using the following parameters:

- *w* – date column width (default 2)
- *l* – number of lines per week (default 1)

You can also use the *prmonth* function, which has the same parameters as the *month* function, but doesn't require you to use the *print* function to display the calendar.

As you already know, by default in the *calendar* module, the first day of the week is Monday. However, you can change this behavior using a function called *setfirstweekday*.

Do you remember the table showing the days of the week and their representation in the form of integer values? It's time to use it, because the *setfirstweekday* method requires a parameter expressing the day of the week in the form of an integer value. Take a look at the example here.

```python
import calendar
calendar.setfirstweekday(calendar.SUNDAY)
calendar.prmonth(2022, 4)
```

The example uses the *calendar.SUNDAY* constant, which contains a value of 6. Of course, you could pass this value directly to the *setfirstweekday* function, but the version with a constant is more elegant.

As a result, we get a calendar showing the month of April 2022, in which the first day of all the weeks is Sunday.

Another useful function provided by the *calendar* module is the function called *weekday*, which returns the day of the week as an integer value for the given year, month, and day. Let's see it here.

```python
import calendar
print(calendar.weekday(2021, 12, 26)) # 6
```

The code here prints the day of the week that falls on December 26, 2021, remember that 6 means Sunday.

You've probably noticed that the calendar contains weekly headers in a shortened form. If needed, you can get short weekday names using the *weekheader* method.

The *weekheader* method requires you to specify the width in characters for one day of the week. If the width you provide is greater than 3, you'll still get the abbreviated weekday names consisting of three characters.

```
import calendar
print(calendar.weekheader(2)) # Mo Tu We Th Fr Sa Su
```

If you change the first day of the week, e.g., using the *setfirstweekday* function, it'll affect the result of the *weekheader* function.

The *calendar* module provides two useful functions to check whether years are leap years.

The first one, called *isleap*, returns *True* if the **past** year is leap, or *False* otherwise.

The second one, called *leapdays*, returns the number of leap years in the given range of years (exclusive).

```
import calendar
print(calendar.isleap(2020)) # True
print(calendar.leapdays(2010, 2021)) # 3
print(calendar.leapdays(2010, 2020)) # 2
```

The presented functions aren't everything the *calendar* module offers. In addition to them, we can use the following classes:

- *calendar.Calendar* – provides methods to prepare calendar data for formatting.
- *calendar.TextCalendar* – is used to create regular text calendars.
- *calendar.HTMLCalendar* – is used to create HTML calendars.
- *calendar.LocalTextCalendar* – is a subclass of the *calendar.TextCalendar* class. The constructor of this class takes the locale parameter, which is used to return the appropriate months and weekday names.
- *calendar.LocalHTMLCalendar* – is a subclass of the *calendar.HTMLCalendar* class. The constructor of this class takes the locale parameter, which is used to return the appropriate months and weekday names.

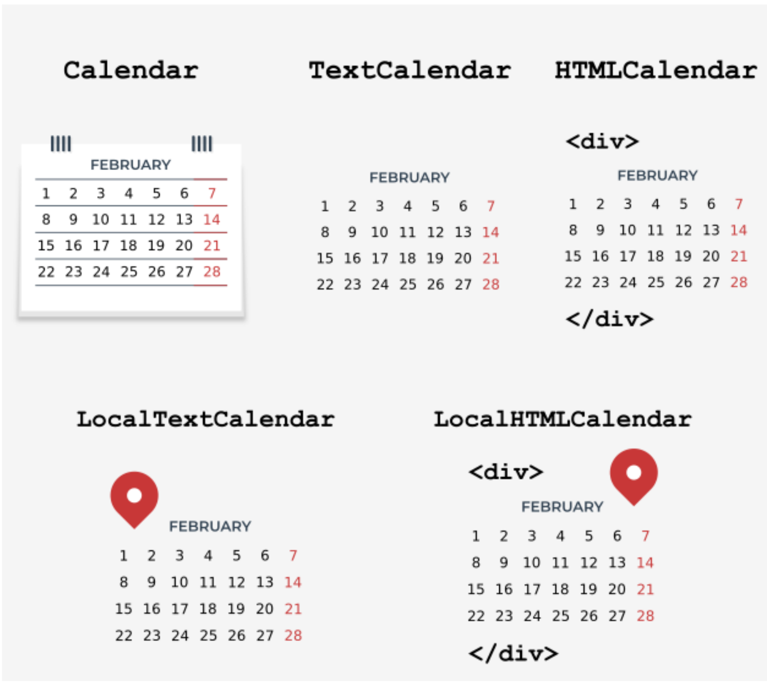


Image source: Cisco/Python Institute

Creating a *Calendar* object

The *Calendar* class constructor takes one optional parameter named *firstweekday*, by default equal to 0 (Monday).

The $firstweekday$ parameter must be an integer between 0-6. For this purpose, we can use the already-known constants.

```python
import calendar

c = calendar.Calendar(calendar.SUNDAY)
for weekday in c.iterweekdays():
    print(weekday, end = " ") # 6 0 1 2 3 4 5
```

The code example uses the $Calendar$ class method named $iterweekdays$, which returns an iterator for week day numbers.

The first value returned is always equal to the value of the $firstweekday$ property. As in our example the first value returned is 6, it means that the week starts on a Sunday.

The $Calendar$ class has several methods that return an iterator. One of them is the $itermonthdates$ method, which requires specifying the year and month.

As a result, all days in the specified month and year are returned, as well as all days before the beginning of the month or the end of the month that are necessary to get a complete week.

Each day is represented by a $datetime.date$ object. Take a look here.

```python
import calendar

c = calendar.Calendar()
for date in c.itermonthdates(2022, 4):
    print(date, end = " ")
```

The code displays all days in April 2022. Because the first day of April 2022 was a Friday, some days are also returned to get the complete week.

Expected output:

2022-03-28 2022-03-29 2022-03-30 2022-03-31 2022-04-01 2022-04-02 2022-04-03 2022-04-04 2022-04-05 2022-04-06 2022-04-07 2022-04-08 2022-04-09 2022-04-10 2022-04-11 2022-04-12 2022-04-13 2022-04-14 2022-04-15 2022-04-16 2022-04-17 2022-04-18 2022-04-19 2022-04-20 2022-04-21 2022-04-22 2022-04-23 2022-04-24 2022-04-25 2022-04-26 2022-04-27 2022-04-28 2022-04-29 2022-04-30 2022-05-01

Another useful method in the $Calendar$ class is the method called $itermonthdates$, which takes year and month as parameters, and then returns the iterator to the days of the week represented by numbers.

```python
import calendar

c = calendar.Calendar()
for iter in c.itermonthdays(2022, 4):
    print(iter, end = " ")
```

You'll have certainly noticed the large number of 0s returned as a result of the example code. These are days outside the specified month range that are added to keep the complete week. The remaining numbers are days in the month.

There are four other similar methods in the $Calendar$ class that differ in data returned:

- $itermonthdates2$ – returns days in the form of tuples consisting of a day of the month number and a week day number.
- $itermonthdates3$ – returns days in the form of tuples consisting of a year, a month, and a day of the month numbers. This method has been available since version 3.7.
- $itermonthdates4$ – returns days in the form of tuples consisting of a year, a month, a day of the month, and a day of the week numbers. This method has been available since Python version 3.7.

The $Calendar$ class has several other useful methods that you can learn more about in the documentation here: calendar — General calendar-related functions — Python 3.10.4 documentation.

One of them is the $monthdays2calendar$ method, which takes the year and month, and then returns a list of weeks in a specific month. Each week is a tuple consisting of day numbers and weekday numbers.

```python
import calendar

c = calendar.Calendar()
for data in c.monthdays2calendar(2022, 4):
    print(data)
```

Note that the days numbers outside the month are represented by 0, while the weekday numbers are a number from 0-6, where 0 is Monday and 6 is Sunday.