

Dates

Date of Derafu Support

Anonymous · 08/10/2026

Dates

Date extends what [Carbon](#) gives with what the business dates need: formatting in Spanish, periods (YYYYMM), working days and holidays, and moving in units of time. It needs `nesbot/carbon` (see [Install](#)).

Everywhere a date is asked, a Carbon instance or a text that Carbon understands ('2024-01-15') is valid, and the dates that are given are Carbon instances. A **period** is an integer YYYYMM (202401), and a year is YYYY. The **holidays** are a list of texts Y-m-d.

Formatting in Spanish

```
use Derafu\Support\Date;

Date::formatSpanish('2024-01-15');           // "Lunes, 15 de Enero del 2024"
Date::formatSpanish('2024-01-15', false);    // "15 de Enero del 2024"
Date::formatPeriodSpanish(202403);           // "Marzo de 2024"
Date::agoSpanish($date);                     // "hace 5 minutos"
Date::agoSpanish($date, true);               // "hace 3 días, 2 horas"
```

`agoSpanish()` gives only the most significant part by default, and all of them with `$full`. `formatPeriodSpanish()` with a period that is not valid is an `InvalidArgumentException`.

Creating and validating

Method	What it does
<code>Date::create(\$date)</code>	A Carbon from a text; one that Carbon can not read is an <code>InvalidArgumentException</code> .
<code>Date::ensureCarbon(\$date)</code>	The same Carbon, or the one that the text says.
<code>Date::fromSerialNumber(\$n)</code>	The date of a serial number of Excel (see below).
<code>Date::validate(\$date, \$format = 'Y-m-d')</code>	Whether the text is a real date in that format (2024-02-30 is not).

Method	What it does
<code>Date::validateAndConvert(\$date, \$format = 'd/m/Y')</code>	Whether a text Y-m-d is a valid date, and if so, it written in the new format; <code>null</code> if it is not valid.

```
Date::validate('2024-02-29'); // true
Date::validate('2024-02-30'); // false
Date::validateAndConvert('2024-01-15'); // "15/01/2024"
Date::validateAndConvert('15/01/2024'); // null: the input must be Y-m-d.
```

Serial numbers of Excel

Excel counts the days from January 1, 1900 (the number 1), and it takes 1900 for a leap year, so the number 60 is a February 29, 1900 that never existed. `fromSerialNumber()` follows it: the numbers before 60 are the dates before March 1, 1900, the number 60 gives 1900-02-28, and from 61 (1900-03-01) on the number is the days after December 30, 1899. The time is the start of the day.

```
Date::fromSerialNumber(1); // 1900-01-01
Date::fromSerialNumber(61); // 1900-03-01
Date::fromSerialNumber(25569); // 1970-01-01
Date::fromSerialNumber(45292); // 2024-01-01
```

Counting

```
Date::calculateAge('2000-06-15'); // The age in years.
Date::countDays('2024-01-01', '2024-01-31'); // 30 (to the current date if
there is no `to`).
Date::countMonths('2024-01-15', '2024-04-14'); // 3
Date::countMonths(202401, 202404); // 3: periods are valid too.
Date::countDaysMatch('2024-01-01', '2024-01-31', ['2024-01-06', '2024-01-17']); // 2
Date::countDaysMatch('2024-01-01', '2024-01-31', ['2024-01-06', '2024-01-17'],
excludeWeekend: true); // 1: the 6th is a Saturday.
```

`countMonths()` is negative if the end is before the start. `countDaysMatch()` counts how many of the dates of the list (texts Y-m-d) are in the range, both ends included, and with `$excludeWeekend` it skips the Saturdays and Sundays.

Weeks and years

```
Date::firstDayWeek('2024-01-17'); // 2024-01-15, the Monday.
Date::lastDayWeek('2024-01-17'); // 2024-01-21, the Sunday.
Date::generateYears(3, 2022); // [2022, 2021, 2020]: descending, from the
current year if there is no `$from`.
```

Periods

Method	What it does
<code>Date::periodToCarbon(\$period)</code>	The first day of the period.
<code>Date::validPeriod4(\$period, \$yearFrom = 2000, \$yearTo = 2100)</code>	Whether it is a valid YYYY.
<code>Date::validPeriod6(\$period, \$yearFrom = 2000, \$yearTo = 2100)</code>	Whether it is a valid YYYYMM.
<code>Date::validPeriod(\$period, \$yearFrom = 2000, \$yearTo = 2100, \$length = null)</code>	Either of them (or the one that <code>\$length</code> , 4 or 6, says).
<code>Date::daysInPeriod(\$period)</code>	The days of its month.
<code>Date::firstDayPeriod(\$period = null), lastDayPeriod(\$period = null)</code>	The first and the last day, as Y-m-d texts (the current month if there is no period).
<code>Date::nextPeriod(\$period = null, \$steps = 1), previousPeriod(...)</code>	The period that is <code>\$steps</code> months after or before.

```
Date::nextPeriod(202412);           // 202501
Date::previousPeriod(202401);       // 202312
Date::nextPeriod(202401, 13);       // 202502
Date::daysInPeriod(202402);        // 29
Date::lastDayPeriod(202402);        // "2024-02-29"
Date::validPeriod6(202413);         // false
```

Moving in units of time

`nextDate()` and `previousDate()` move a date by units: D days, W weeks, M months, Q quarters, S semesters and Y years. The date is the current one if none is given.

```
Date::nextDate('2024-01-15', 'Q'); // 2024-04-15
Date::previousDate('2024-01-15', 'W', 2); // 2024-01-01
```

The months, quarters, semesters and years do not go over into the next month: when the day does not exist in the month that is reached, the result is the **last day of that month**. The date that is given is not changed, the result is a new `Carbon`.

```
Date::nextDate('2024-01-31', 'M'); // 2024-02-29 (2023-01-31 gives 2023-02-28)
Date::nextDate('2024-08-31', 'S'); // 2025-02-28
Date::nextDate('2024-02-29', 'Y'); // 2025-02-28
Date::previousDate('2024-03-31', 'M'); // 2024-02-29
```

The moves do not undo each other: January 31 plus a month is February 29, and minus a month it is January 29.

Working days

The working days are the Monday to Friday that are not in the list of holidays.

Method	What it does
<code>Date::addWorkingDays(\$date, \$days, \$holidays = [])</code>	The date that is that many working days later.
<code>Date::subtractWorkingDays(\$date, \$days, \$holidays = [])</code>	The date that is that many working days before.
<code>Date::getWorkingDay(\$year, \$month, \$workingDay, \$holidays = [])</code>	The date of the n-th working day of a month, or false.
<code>Date::getWorkingDayNumber(\$date, \$holidays = [])</code>	The number of the working day of its month that a date is, or false if it is not a working day.
<code>Date::isLastWorkingDay(\$date, \$holidays = [])</code>	Whether it is the last working day of its month.

```
$holidays = ['2024-01-17'];

Date::addWorkingDays('2024-01-15', 5, $holidays); // 2024-01-23: the 17th is
skipped, and the weekend.
Date::subtractWorkingDays('2024-01-22', 5, $holidays); // 2024-01-12
Date::getWorkingDay(2024, 1, 3, $holidays); // 2024-01-03
Date::getWorkingDayNumber('2024-01-18', $holidays); // 13: the 17th does not
count.
Date::getWorkingDayNumber('2024-01-20'); // false: it is a Saturday.
Date::isLastWorkingDay('2024-01-31'); // true
```

Last updated on 08/10/2026