

CS50's Introduction to Programming with Python

OpenCourseWare

Donate [🔗](https://cs50.harvard.edu/donate) (https://cs50.harvard.edu/donate)

David J. Malan (https://cs.harvard.edu/malan/)

malan@harvard.edu

[f](https://www.facebook.com/dmalan) (https://www.facebook.com/dmalan)
 [G](https://github.com/dmalan) (https://github.com/dmalan)
 [@](https://www.instagram.com/davidjmalan/) (https://www.instagram.com/davidjmalan/)
 [in](https://www.linkedin.com/in/malan/) (https://www.linkedin.com/in/malan/)
 [id](https://orcid.org/0000-0001-5338-2522) (https://orcid.org/0000-0001-5338-2522)
 [Q](https://www.quora.com/profile/David-J-Malan) (https://www.quora.com/profile/David-J-Malan)
 [r](https://www.reddit.com/user/davidjmalan) (https://www.reddit.com/user/davidjmalan)
 [T](https://www.tiktok.com/@davidjmalan) (https://www.tiktok.com/@davidjmalan)
 [T](https://twitter.com/davidjmalan) (https://twitter.com/davidjmalan)

Outdated

In the United States, dates are typically formatted in **month-day-year order** (https://en.wikipedia.org/wiki/Date_and_time_notation_in_the_United_States) (MM/DD/YYYY), otherwise known as **middle-endian** (<https://en.wikipedia.org/wiki/Endianness#Middle-endian>) order, which is arguably bad design. Dates in that format can't be easily sorted because the date's year comes last instead of first. Try sorting, for instance, `2/2/1800`, `3/3/1900`, and `1/1/2000` chronologically in any program (e.g., a spreadsheet). Dates in that format are also ambiguous. Harvard was **founded** (<https://www.harvard.edu/about/history/>) on September 8, 1636, but `9/8/1636` could also be interpreted as August 9, 1636!

Fortunately, computers tend to use **ISO 8601** (https://en.wikipedia.org/wiki/ISO_8601), an international standard that prescribes that dates should be formatted in year-month-day (YYYY-MM-DD) order, no matter the country, formatting years with four digits, months with two digits, and days with two digits, "padding" each with leading zeroes as needed.

In a file called `outdated.py`, implement a program that prompts the user for a date, **anno Domini** (https://en.wikipedia.org/wiki/Anno_Domini), in month-day-year order, formatted like `9/8/1636` or `September 8, 1636`, wherein the month in the latter might be any of the values in the `list` below:

```
[
    "January",
    "February",
    "March",
    "April",
    "May",
    "June",
    "July",
    "August",
    "September",
    "October",
    "November",
    "December"
]
```

Then output that same date in `YYYY-MM-DD` format. If the user's input is not a valid date in either format, prompt the user again. Assume that every month has no more than 31 days; no need to validate whether a month has 28, 29, 30, or 31 days.

▼ Hints

- Recall that a `str` comes with quite a few methods, per docs.python.org/3/library/stdtypes.html#string-methods (<https://docs.python.org/3/library/stdtypes.html#string-methods>), including `split`, which separates a `str` into a list of values, all of which can be assigned to variables at once. For instance, if `expression` is a `str` like `1 + 1`, then

```
x, y, z = expression.split(" ")
```

will assign `1` to `x`, `+` to `y`, and `1` to `z`.

- Recall that a `list` comes with quite a few methods, per docs.python.org/3/tutorial/datastructures.html#more-on-lists

(<https://docs.python.org/3/tutorial/datastructures.html#more-on-lists>), among which is `index`.

- Note that you can format an `int` with leading zeroes with code like

```
print(f"{n:02}")
```

wherein, if `n` is a single digit, it will be prefixed with one `0`, per docs.python.org/3/library/string.html#format-string-syntax (<https://docs.python.org/3/library/string.html#format-string-syntax>).

Demo

```
1636-09-08
$ python outdated.py
Date: 8/9/1636
1636-08-09
$ python outdated.py
Date: 1/1/1970
1970-01-01
$ python outdated.py
Date: Jan 1, 1970
Date: January 1, 1970
1970-01-01
$
```

Recorded with [asciinema](#)

Before You Begin

Log into code.cs50.io (<https://code.cs50.io/>), click on your terminal window, and execute `cd` by itself. You should find that your terminal window's prompt resembles the below:

```
$
```

Next execute

```
mkdir outdated
```

to make a folder called `outdated` into your codespace.

Then execute

```
cd outdated
```

to change directories into that folder. You should now see your terminal prompt as `outdated/ $`. You can now execute

```
code outdated.py
```

to make a file called `outdated.py` where you'll write your program.

How to Test

Here's how to test your code manually:

- Run your program with `python outdated.py`. Type `9/8/1636` and press Enter. Your program should output:

```
1636-09-08
```

- Run your program with `python outdated.py`. Type `September 8, 1636` and press Enter. Your program should output:

```
1636-09-08
```

- Run your program with `python outdated.py`. Type `23/6/1912` and press Enter. Your program should reprompt the user.
- Run your program with `python outdated.py`. Type `December 80, 1980` and press Enter. Your program should reprompt the user.

You can execute the below to check your code using `check50`, a program that CS50 will use to test your code when you submit. But be sure to test it yourself as well!

```
check50 cs50/problems/2022/python/outdated
```

Green smilies mean your program has passed a test! Red frownies will indicate your program output something unexpected. Visit the URL that `check50` outputs to see the input `check50` handed to your program, what output it expected, and what output your program actually gave.

How to Submit

In your terminal, execute the below to submit your work.

```
submit50 cs50/problems/2022/python/outdated
```