CS50's Introduction to Programming with Python

OpenCourseWare

Donate (https://cs50.harvard.edu/donate)

David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.facebook.com/dmalan) (https://github.com/dmalan)
(https://www.instagram.com/davidjmalan/) (https://www.linkedin.com/in/malan/)
(https://www.reddit.com/user/davidjmalan)
(https://www.threads.net/@davidjmalan) (https://twitter.com/davidjmalan)

Working 9 to 5

Whereas most countries (https://en.wikipedia.org/wiki/Date_and_time_representation_by_country#Time) use a 24-hour clock (https://en.wikipedia.org/wiki/24-hour_clock), the United States tends to use a 12-hour clock (https://en.wikipedia.org/wiki/12-hour_clock). Accordingly, instead of "09:00 to 17:00", many Americans would say they work "9:00 AM to 5:00 PM" (or "9 AM to 5 PM"), wherein "AM" is

an abbreviation for “ante meridiem” and “PM” is an abbreviation for “post meridiem”, wherein “meridiem” means midday (i.e., noon).

▼ **Conversion Table**

Just as “12:00 AM” in 12-hour format would be “00:00” in 24-hour format, so would “12:01 AM” through “12:59 AM” be “00:01” through “00:59”, respectively.

12-Hour	24-Hour
12:00 AM	00:00
1:00 AM	01:00
2:00 AM	02:00
3:00 AM	03:00
4:00 AM	04:00
5:00 AM	05:00
6:00 AM	06:00
7:00 AM	07:00
8:00 AM	08:00
9:00 AM	09:00
10:00 AM	10:00
11:00 AM	11:00
12:00 PM	12:00
1:00 PM	13:00
2:00 PM	14:00
3:00 PM	15:00
4:00 PM	16:00
5:00 PM	17:00
6:00 PM	18:00
7:00 PM	19:00

12-Hour	24-Hour
8:00 PM	20:00
9:00 PM	21:00
10:00 PM	22:00
11:00 PM	23:00
12:00 AM	00:00

In a file called `working.py`, implement a function called `convert` that expects a `str` in either of the 12-hour formats below and returns the corresponding `str` in 24-hour format (i.e., `9:00 to 17:00`). Expect that `AM` and `PM` will be capitalized (with no periods therein) and that there will be a space before each. Assume that these times are representative of actual times, not necessarily 9:00 AM and 5:00 PM specifically.

- `9:00 AM to 5:00 PM`
- `9 AM to 5 PM`

Raise a `ValueError` instead if the input to `convert` is not in either of those formats or if either time is invalid (e.g., `12:60 AM`, `13:00 PM`, etc.). But do not assume that someone's hours will start ante meridiem and end post meridiem; someone might work late and even long hours (e.g., `5:00 PM to 9:00 AM`).

Structure `working.py` as follows, wherein you're welcome to modify `main` and/or implement other functions as you see fit, but you may not import any other libraries. You're welcome, but not required, to use `re` and/or `sys`.

```
import re
import sys


def main():
    print(convert(input("Hours: ")))


def convert(s):
    ...


...


if __name__ == "__main__":
    main()
```

Either before or after you implement `convert` in `working.py`, additionally implement, in a file called `test_working.py`, **three or more** functions that collectively test your implementation of `convert` thoroughly, each of whose names should begin with `test_` so that you can execute your tests with:

```
pytest test_working.py
```

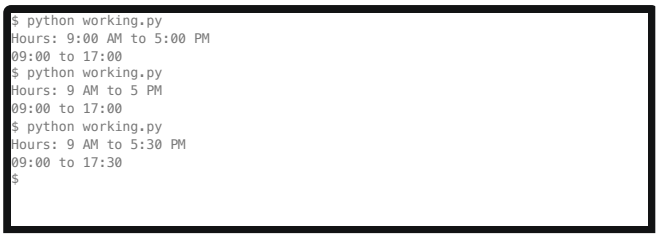
▼ **Hints**

- Recall that the `re` module comes with quite a few functions, per docs.python.org/3/library/re.html (https://docs.python.org/3/library/re.html), including `search`.
- Recall that regular expressions support quite a few special characters, per docs.python.org/3/library/re.html#regular-expression-syntax (https://docs.python.org/3/library/re.html#regular-expression-syntax).
- Because backslashes in regular expressions could be mistaken for escape sequences (like `\n`), best to use Python's raw string notation for regular expression patterns (https://docs.python.org/3/library/re.html#module-re), else `pytest` will warn with `DeprecationWarning: invalid escape sequence`. Just as format strings are prefixed with `f`, so are raw strings prefixed with `r`. For instance, instead of `"harvard\.edu"`, use `r"harvard\.edu"`.
- Note that `re.search`, if passed a pattern with "capturing groups" (i.e., parentheses), returns a "match object," per docs.python.org/3/library/re.html#match-objects (https://docs.python.org/3/library/re.html#match-objects), wherein matches are 1-indexed, which you can access individually with `group`, per docs.python.org/3/library/re.html#re.Match.group (https://docs.python.org/3/library/re.html#re.Match.group), or collectively with `groups`, per docs.python.org/3/library/re.html#re.Match.groups (https://docs.python.org/3/library/re.html#re.Match.groups).
- 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



Recorded with asciinema

Before You Begin

Log into cs50.dev (https://cs50.dev/), 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 working
```

to make a folder called `working` in your codespace.

Then execute

```
cd working
```

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

```
code working.py
```

to make a file called `working.py` where you'll write your program. Be sure to also execute

```
code test_working.py
```

to create a file called `test_working.py` where you'll write tests for your program.

How to Test

How to Test `working.py`

Here's how to test `working.py` manually:

- Run your program with `python working.py`. Ensure your program prompts you for a time. Type `9 AM to 5 PM`, followed by Enter. Your program should output `09:00 to 17:00`.
- Run your program with `python working.py`. Type `9:00 AM to 5:00 PM`, followed by Enter. Your program should again output `09:00 to 17:00`.
- Run your program with `python working.py`. Ensure your program prompts you for a time. Type `10 PM to 8 AM`, followed by Enter. Your program should output `22:00 to 08:00`.
- Run your program with `python working.py`. Ensure your program prompts you for a time. Type `10:30 PM to 8:50 AM`, followed by Enter. Your program should again output `22:30 to 08:50`.
- Run your program with `python working.py`. Ensure your program prompts you for a time. Try intentionally inducing a `ValueError` by typing `9:60 AM to 5:60 PM`, followed by Enter. Your program should indeed raise a `ValueError`.
- Run your program with `python working.py`. Ensure your program prompts you for a time. Try intentionally inducing a `ValueError` by typing `9 AM - 5 PM`, followed by Enter. Your program should indeed raise a `ValueError`.
- Run your program with `python working.py`. Ensure your program prompts you for a time. Try intentionally inducing a `ValueError` by typing `09:00 AM - 17:00 PM`, followed by Enter. Your program should indeed raise a `ValueError`.

How to Test `test_working.py`

To test your tests, run `pytest test_working.py`. Try to use correct and incorrect versions of `working.py` to determine how well your tests spot errors:

- Ensure you have a correct version of `working.py`. Run your tests by executing `pytest test_working.py`. `pytest` should show that all of your tests have passed.
- Modify the correct version of `working.py`, particularly its function `convert`. Your program might, for example, fail to raise a `ValueError` when it should. Run your tests by executing `pytest test_working.py`. `pytest` should show that at least one of your tests has failed.

- Similarly, modify the correct version of `working.py`, changing the return values of `convert`. Your program might, for example, mistakenly omit minutes. Run your tests by executing `pytest test_working.py`. `pytest` should show that at least one of your tests has failed.

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/working
```

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/working
```