





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://orcid.org/0000-0001-5338-2522>)  ([https://www.quora.com/profile](https://www.quora.com/profile/David-J-Malan)

[/David-J-Malan](https://www.quora.com/profile/David-J-Malan))  (<https://www.reddit.com/user/davidjmalan>) 

(<https://www.tiktok.com/@davidjmalan>)  (<https://twitter.com/davidjmalan>)

Meal Time

Suppose that you're in a country where it's customary to eat breakfast between 7:00 and 8:00, lunch between 12:00 and 13:00, and dinner between 18:00 and 19:00. Wouldn't it be nice if you had a program that could tell you what to eat when?

In `meal.py`, implement a program that prompts the user for a time and outputs whether it's `breakfast time`, `lunch time`, or `dinner time`. If it's not time for a meal, don't output anything at all. Assume that the user's input will be formatted in 24-hour time as `#:##` or `##:##`. And assume that each meal's time range is inclusive. For instance, whether it's 7:00, 7:01, 7:59, or 8:00, or anytime in between, it's time for breakfast.

Structure your program per the below, wherein `convert` is a function (that can be called by `main`) that converts `time`, a `str` in 24-hour format, to the corresponding number of hours as a `float`. For instance, given a `time` like `"7:30"` (i.e., 7 hours and 30 minutes), `convert` should return `7.5` (i.e., 7.5 hours).

```
def main():
    ...

def convert(time):
```

```
...

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

▼ 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 `time` is a `str` like `"7:30"`, then

```
hours, minutes = time.split(":")
```

will assign `"7"` to `hours` and `"30"` to `minutes`.

- Keep in mind that there are 60 minutes in 1 hour.

Demo

```
What time is it? 7:00
breakfast time
$ python meal.py
What time is it? 7:30
breakfast time
$ python meal.py
What time is it? 8:01
$ python meal.py
What time is it? 18:01
dinner time
$ python meal.py
What time is it? 18:█
```

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 meal
```

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

Then execute

```
cd meal
```

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

```
code meal.py
```

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

Challenge

If up for a challenge, optionally add support for 12-hour times, allowing the user to input times in these formats too:

- `#:## a.m.` and `##:## a.m.`
- `#:## p.m.` and `##:## p.m.`

How to Test

Here's how to test your code manually:

- Run your program with `python meal.py`. Type `7:00` and press Enter. Your program should output:

```
breakfast time
```

- Run your program with `python meal.py`. Type `7:30` and press Enter. Your program should output:

```
breakfast time
```

- Run your program with `python meal.py`. Type `12:42` and press Enter. Your program should output

```
lunch time
```

- Run your program with `python meal.py`. Type `18:32` and press Enter. Your program should output

```
dinner time
```

- Run your program with `python meal.py`. Type `11:11` and press Enter. Your program should output nothing.

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/meal
```

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/meal
```

