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/David-J-Malan) (https://www.reddit.com/user/davidjmalan) (https://www.tiktok.com/@davidjmalan) (https://twitter.com/davidjmalan)

Making Faces

Before there were emoji, there were emoticons (https://en.wikipedia.org/wiki/List_of_emoticons), whereby text like `:)` was a happy face and text like `:(` was a sad face. Nowadays, programs tend to convert emoticons to emoji automatically!

In a file called `faces.py`, implement a function called `convert` that accepts a `str` as input and returns that same input with any `:)` converted to 🙂 (otherwise known as a slightly smiling face (https://emojipedia.org/slightly-smiling-face/)) and any `:(` converted to 🙁 (otherwise known as a slightly frowning face (https://emojipedia.org/slightly-frowning-face/)). All other text should be returned unchanged.

Then, in that same file, implement a function called `main` that prompts the user for input, calls `convert` on that input, and prints the result. You’re welcome, but not required, to prompt the user explicitly, as by passing a `str` of your own as an argument to `input`. Be sure to call `main` at the bottom of your file.

▾ Hints

- Recall that `input` returns a `str`, per docs.python.org/3/library/functions.html#input (https://docs.python.org/3/library/functions.html#input).
- 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).
- An emoji is actually just a character, so you can quote it like any `str`, a la `"😐"`. And you can copy and paste the emoji from this page into your own code as needed.

Before You Begin

Execute `cd` by itself in your terminal window. You should find that your terminal window's prompt resembles the below:

```
$
```

Next execute

```
mkdir faces
```

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

Then execute

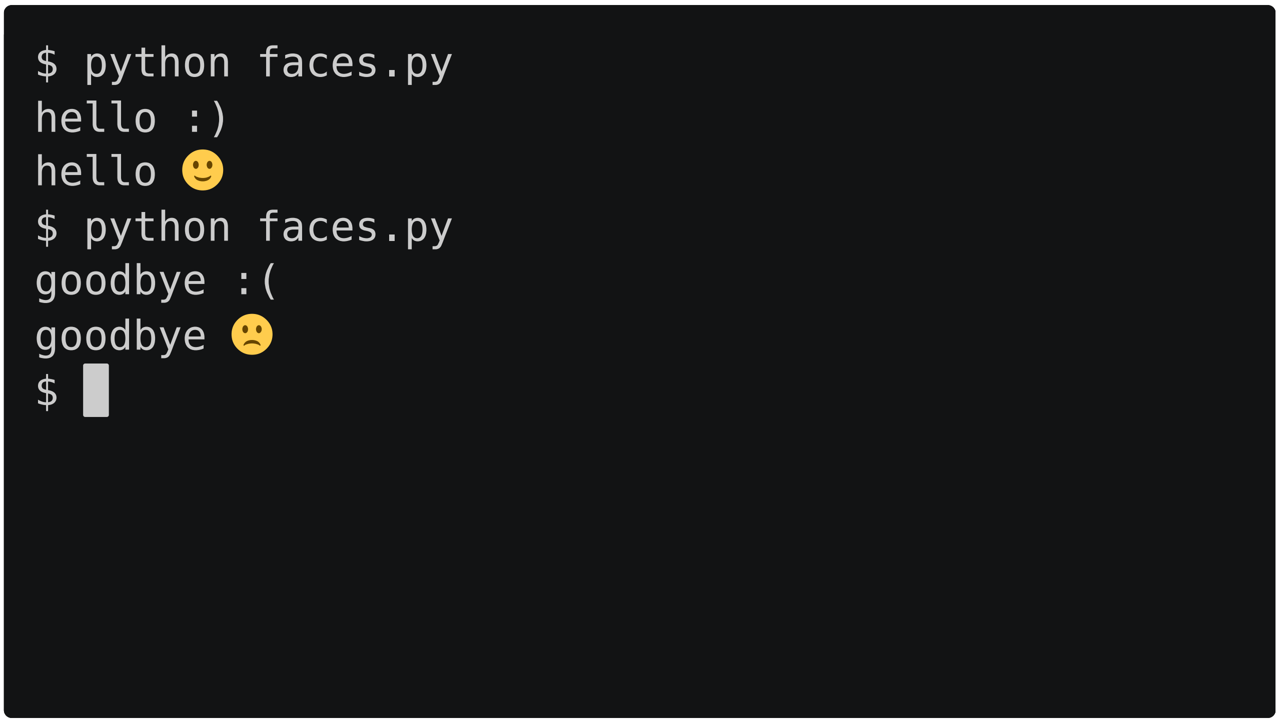
```
cd faces
```

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

```
code faces.py
```

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

Demo



```
$ python faces.py
hello :)
hello 🙂
$ python faces.py
goodbye :(
goodbye 🙁
$ 
```

Recorded with asciinema

How to Test

Here's how to test your code manually:

- Run your program with `python faces.py`. Type `Hello :)` and press Enter. Your program should output:

  ```
  Hello 🙂
  ```

- Run your program with `python faces.py`. Type `Goodbye :(` and press Enter. Your program should output:

  ```
  Goodbye 🙁
  ```

- Run your program with `python faces.py`. Type `Hello :) Goodbye :(` and press Enter. Your program should output

  ```
  Hello 🙂 Goodbye 🙁
  ```

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/faces
```

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/faces
```