



## Exercise 25: Even More Practice

We're going to do some more practice involving functions and variables to make sure you know them well. This exercise should be straightforward for you to type in, break down, and understand.

However, this exercise is a little different. You won't be running it. Instead you will import it into Python and run the functions yourself.

1  
2  
3  
4  
MAIN



PLAY VIDEO



PREVIOUS



NEXT



HELP

Follow

```
1 def break_words(stuff):
2     """This function will break up words for us."""
3     words = stuff.split(' ')
4     return words
5
6 def sort_words(words):
7     """Sorts the words."""
8     return sorted(words)
9
10 def print_first_word(words):
11     """Prints the first word after popping it off."""
12     word = words.pop(0)
13     print word
14
15 def print_last_word(words):
16     """Prints the last word after popping it off."""
17     word = words.pop(-1)
18     print word
19
20 def sort_sentence(sentence):
21     """Takes in a full sentence and returns the sorted words."""
22     words = break_words(sentence)
23     return sort_words(words)
24
25 def print_first_and_last(sentence):
26     """Prints the first and last words of the sentence."""
27     words = break_words(sentence)
28     print_first_word(words)
29     print_last_word(words)
30
31 def print_first_and_last_sorted(sentence):
32     """Sorts the words then prints the first and last one."""
33     words = sort_sentence(sentence)
34     print_first_word(words)
35     print_last_word(words)
```

First, run this with `python ex25.py` to find any errors you have made. Once you have found all of the errors you can and fixed them, you will then want to follow the *What You Should See* section to complete the exercise.

## What You Should See

In this exercise we're going to interact with your `ex25.py` file inside the `python` interpreter you used periodically to do calculations. You run `python` from the terminal like this:

```
$ python
Python 2.7.1 (r271:86832, Jun 16 2011, 16:59:05)
```

```
[GCC 4.2.1 (Based on Apple Inc. build 5658) (LLVM build
2335.15.00)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>>
```

Your output should look like mine, and after the `>` character (called the prompt) you can type Python code in and it will run immediately. Using this I want you to type each of these lines of Python code into `python` and see what it does:

```
1 import ex25
2 sentence = "All good things come to those who wait."
3 words = ex25.break_words(sentence)
4 words
5 sorted_words = ex25.sort_words(words)
6 sorted_words
7 ex25.print_first_word(words)
8 ex25.print_last_word(words)
9 words
10 ex25.print_first_word(sorted_words)
11 ex25.print_last_word(sorted_words)
12 sorted_words
13 sorted_words = ex25.sort_sentence(sentence)
14 sorted_words
15 ex25.print_first_and_last(sentence)
16 ex25.print_first_and_last_sorted(sentence)
```

Here's what it looks like when I work with the `ex25.py` module in `python`:

```
Python 2.7.10 (default, Jul 14 2015, 19:46:27)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.39)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import ex25
>>> sentence = "All good things come to those who wait."
>>> words = ex25.break_words(sentence)
>>> words
['All', 'good', 'things', 'come', 'to', 'those', 'who', 'wait.']
>>> sorted_words = ex25.sort_words(words)
>>> sorted_words
['All', 'come', 'good', 'things', 'those', 'to', 'wait.', 'who']
>>> ex25.print_first_word(words)
All
>>> ex25.print_last_word(words)
wait.
>>> words
['good', 'things', 'come', 'to', 'those', 'who']
>>> ex25.print_first_word(sorted_words)
All
>>> ex25.print_last_word(sorted_words)
who
>>> sorted_words
['come', 'good', 'things', 'those', 'to', 'wait.']
>>> sorted_words = ex25.sort_sentence(sentence)
>>> sorted_words
['All', 'come', 'good', 'things', 'those', 'to', 'wait.', 'who']
>>> ex25.print_first_and_last(sentence)
All
wait.
>>> ex25.print_first_and_last_sorted(sentence)
All
who
```

As you go through each of these lines, make sure you can find the function that's being run in `ex25.py` and you understand how each one works. If you get different results or error, you'll have to go fix your code, exit `python` and start over.

## Study Drills

1. Take the remaining lines of the *What You Should See* output and figure out what they are doing. Make sure you understand how you are running your functions in the `ex25` module.
2. Try doing this: `help(ex25)`; and also `help(ex25.break_words)`. Notice how you get help for your module, and how the help is those odd `"""` strings you put after each function in `ex25`? Those special strings are called *documentation comments* and we'll be seeing more of them.
3. Typing `ex25.` is annoying. A shortcut is to do your import like this: `from ex25 import *` which is like saying, "Import everything from `ex25`." Programmers like saying things backward. Start a new session and see how all your functions are right there.
4. Try breaking your file and see what it looks like in `python` when you use it. You will have to quit `python` with `quit()` to be able to reload it.

## Common Student Questions



I get `None` printed out for some of the functions.

You probably have a function that is missing the `return` at the end. Go backward through the file and confirm that every line is right.



I get `-bash: import: command not found` when I type `import ex25`.

Pay attention to what I'm doing in the *What You Should See* section. I'm doing this in *Python* not in the Terminal. That means you first run Python.



I get `ImportError: No module named ex25.py` when I type `import ex25.py`.

Don't add the `.py` to the end. Python knows the file ends in `.py` so you just type `import ex25`.



I get `SyntaxError: invalid syntax` when I run this.

That means you have something like a missing `(` or `"` or similar syntax error on that line or above it. Any time you get that error, start at the line it mentions and check that it's right, then go backward checking each line above that.



How can the `words.pop` function change the `words` variable?

That's a complicated question, but in this case `words` is a list, and because of that you can give it commands and it'll retain the results of those commands. This is similar to how files and many other things worked when you were working with them.



When should I `print` instead of `return` in a function?

The `return` from a function gives the line of code that called the function a result. You can think of a function as taking input through its arguments, and returning output through `return`. The `print` is *completely* unrelated to this, and only deals with printing output to the terminal.

# Video

## Purchase The Videos For \$29.59

For just \$29.59 you can get access to all the videos for Learn Python The Hard Way, **plus** a PDF of the book and no more popups all in this one location. For \$29.59 you get:

- All 52 videos, 1 per exercise, almost 2G of video.
- A PDF of the book.
- Email help from the author.
- See a list of everything you get before you buy.

When you buy the videos they will immediately show up **right here** without any hassles.

Already Paid? [Reactivate Your Purchase Right Now!](#)

## Buying Is Easy

Buying is easy. Just fill out the form below and we'll get started.


Full Name

Full Name

Email Address

Email Address

☒    Pay With Credit Card (by Stripe™)

☐  Use your PayPal™ account.

Buy Learn Python The Hard Way, 3rd Edition

<b>Zed Shaw</b> PDF + Videos + Updates <b>\$29.95</b>	<b>Amazon</b> Paper + DVD <b>\$22.74</b> <b>InformIT</b> eBook + Paper <b>\$43.19</b>	<b>Amazon</b> Kindle <b>\$17.27</b> <b>Interested In Ruby?</b> Ruby is also a great language. <b>Learn Ruby The Hard Way</b>	<b>B&amp;N</b> Paper + DVD <b>\$22.96</b>	<b>B&amp;N</b> Nook (No Video) <b>\$17.27</b>
---	--	---	---	---