



## Exercise 36: Designing and Debugging

Now that you know `if-statements`, I'm going to give you some rules for `for-loops` and `while-loops` that will keep you out of trouble. I'm also going to give you some tips on debugging so that you can figure out problems with your program. Finally, you will design a little game similar to the last exercise, but with a slight twist.

### Rules for If-Statements

1. Every `if-statement` must have an `else`.
2. If this `else` should never run because it doesn't make sense, then you must use a `die` function in the `else` that prints out an error message and dies, just like we did in the last exercise. This will find *many* errors.
3. Never nest `if-statements` more than two deep and always try to do them one deep.
4. Treat `if-statements` like paragraphs, where each `if-elif-else` grouping is like a set of sentences. Put blank lines before and after.
5. Your boolean tests should be simple. If they are complex, move their calculations to variables earlier in your function and use a good name for the variable.

If you follow these simple rules, you will start writing better code than most programmers. Go back to the last exercise and see if I followed all of these rules. If not, fix my mistakes.

#### WARNING

Never be a slave to the rules in real life. For training purposes you need to follow these rules to make your mind strong, but in real life sometimes these rules are just stupid. If you think a rule is stupid, try not using it.

### Rules for Loops

1. Use a `while-loop` only to loop forever, and that means probably never. This only applies to Python; other languages are different.
2. Use a `for-loop` for all other kinds of looping, especially if there is a fixed or limited number of things to loop over.

1  
2  
3  
4  
MAIN



Follow

## Tips for Debugging

1. Do not use a "debugger." A debugger is like doing a full-body scan on a sick person. You do not get any specific useful information, and you find a whole lot of information that doesn't help and is just confusing.
2. The best way to debug a program is to use `print` to print out the values of variables at points in the program to see where they go wrong.
3. Make sure parts of your programs work as you work on them. Do not write massive files of code before you try to run them. Code a little, run a little, fix a little.

## Homework

Now write a similar game to the one that I created in the last exercise. It can be any kind of game you want in the same flavor. Spend a week on it making it as interesting as possible. For Study Drills, use lists, functions, and modules (remember those from Exercise 13?) as much as possible, and find as many new pieces of Python as you can to make the game work.

Before you start coding you must draw a map for your game. Create the rooms, monsters, and traps that the player must go through on paper before you code.

Once you have your map, try to code it up. If you find problems with the map then adjust it and make the code match.

The best way to work on a piece of software is in small chunks like this:

1. On a sheet of paper or an index card, write a list of tasks you need to complete to finish the software. This is your to do list.
2. Pick the easiest thing you can do from your list.
3. Write out English comments in your source file as a guide for how you would accomplish this task in your code.
4. Write some of the code under the English comments.
5. Quickly run your script so see if that code worked.
6. Keep working in a cycle of writing some code, running it to test it, and fixing it until it works.
7. Cross this task off your list, then pick your next easiest task and repeat.

This process will help you work on software in a methodical and consistent manner. As you work, update your list by removing tasks you don't really need and adding ones you do.

## Video

**Purchase The Videos For \$29.59**

For just \$29.59 you can get access to all the videos for Learn Python The

**Buying Is Easy**

Buying is easy. Just fill out the fc

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!





below and we'll get started.

Full Name

Full Name

Email Address

Email Address

☒    ☐  US

Pay With Credit Card (by Stripe™)      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>Amazon</b> Kindle <b>\$17.27</b>	<b>B&amp;N</b> Paper + DVD <b>\$22.96</b>	<b>B&amp;N</b> Nook (No Video) <b>\$17.27</b>
	<b>InformIT</b> eBook + Paper <b>\$43.19</b>	<b>Interested In Ruby?</b> Ruby is also a great language. <b>Learn Ruby The Hard Way</b>		