Learn Python The Hard Way
(http://learnpythonthehardway.org)

Take The Video Course For $29
(http://inculcate.me/school/courses/2/)

Ruby
(http://ruby.learncodethehardway.org

Exercise 37: Symbol Review

It's time to review the symbols and Python words you know, and to try to pick up a few more for the next few lessons. What I've done here is written out all the Python symbols and keywords that are important to know.

In this lesson take each keyword, and first try to write out what it does from memory. Next, search online for it and see what it really does. It may be hard because some of these are going to be impossible to search for, but keep trying.

If you get one of these wrong from memory, write up an index card with the correct definition and try to "correct" your memory. If you just didn't know about it, write it down, and save it for later.

Finally, use each of these in a small Python program, or as many as you can get done. The key here is to find out what the symbol does, make sure you got it right, correct it if you do not, then use it to lock it in.

Keywords

- and
- del
- from
- not
- while
- as
- elif
- global
- or
- with
- assert
- else
- if
- pass
- yield
- break
- except
- import
- print
- class
- exec
- in
- raise
- continue
- finally
- is
- return
- def
- for
- lambda
- try

Data Types

For data types, write out what makes up each one. For example, with strings write out how you create a string. For numbers write out a few numbers.

- True
- False
- None
- strings

- numbers
- floats
- lists

String Escapes Sequences

For string escape sequences, use them in strings to make sure they do what you think they do.

- `\\`
- `\'`
- `\"`
- `\a`
- `\b`
- `\f`
- `\n`
- `\r`
- `\t`
- `\v`

String Formats

Same thing for string formats: use them in some strings to know what they do.

- `%d`
- `%i`
- `%o`
- `%u`
- `%x`
- `%X`
- `%e`
- `%E`
- `%f`
- `%F`
- `%g`
- `%G`
- `%c`
- `%r`
- `%s`
- `%%`

Operators

Some of these may be unfamiliar to you, but look them up anyway. Find out what they do, and if you still can't figure it out, save it for later.

- `+`
- `-`
- `*`
- `**`
- `/`
- `//`
- `%`
- `<`
- `>`
- `<=`
- `>=`
- `==`
- `!=`
- `<>`
- `( )`

- `[ ]`
- `{ }`
- `@`
- `,`
- `:`
- `.`
- `=`
- `;`
- `+=`
- `-=`
- `*=`
- `/=`
- `//=`
- `%=`
- `**=`

Spend about a week on this, but if you finish faster that's great. The point is to try to get coverage on all these symbols and make sure they are locked in your head. What's also important is to find out what you *do not* know so you can fix it later.

Reading Code

Now go find some Python code to read. You should be reading any Python code you can and trying to steal ideas that you find. You actually should have enough knowledge to be able to read, but maybe not understand what the code does. What I'm going to teach you in this lesson is how to apply things you have learned to understand other people's code.

First, print out the code you want to understand. Yes, print it out, because your eyes and brain are more used to reading paper than computer screens. Make sure you only print a few pages at a time.

Second, go through your printout and take notes of the following:

1. Functions and what they do.
2. Where each variable is first given a value.
3. Any variables with the same names in different parts of the program. These may be trouble later.
4. Any `if-statements` without else clauses. Are they right?
5. Any `while-loops` that might not end.
6. Finally, any parts of code that you can't understand for whatever reason.

Third, once you have all of this marked up, try to explain it to yourself by writing comments as you go. Explain the functions, how they are used, what variables are involved, anything you can to figure this code out.

Lastly, on all of the difficult parts, trace the values of each variable line by line, function by function. In fact, do another printout and write in the margin the value of each variable that you need to "trace".

Once you have a good idea of what the code does, go back to the computer and read it again to see if you find new things. Keep finding more code and doing this until you do not need the printouts anymore.

Study Drills

1. Find out what a "flow chart" is and write a few.
2. If you find errors in code you are reading, try to fix them and send the author your changes.
3. Another technique for when you are not using paper is to put `#` comments with your notes in the code. Sometimes, these could become the actual comments to help the next person.

Common Student Questions

What's the difference between `%d` and `%i` formatting?
Shouldn't be any difference, other than people use `%d` more due to historical reasons.

How would I search for these things online?
Simply put "python" before anything you want to find. For example, to find `yield` do `python yield`.