Python Essential Reference: Chapter 2

Lexical Conventions & Syntax

White Space Matters: Lines

- Each statement is ended with a newline
- You can continue a long statement over multiple lines with the \ character
- Or if the statement is a """triple quoted string""", (args), {dictionaries}, [lists]

White Space Matters: Indentation

- Indentation is used to show blocks of code such as functions, classes, conditionals, and loops.
- Tabs or spaces are acceptable. This is controversial.
- If you're starting a new project, use 4 spaces.

Identifiers

- How you name variables, functions, classes, modules, etc.
- Includes letters, numbers, and underscores.
- Can't start with a number.
- Case sensitive (hello != Hello)
- Special symbols not allowed (sorry $jquery hackers)
- In general don't start identifiers with underscores.

Reserved Words

- Don't use these as identifiers

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

Numeric Literals

- Boolean Values (`True`, `False`, caps matter)
- Integers (`1, 2, 3, 4`, `199382749230` etc)
- You don't have to worry about long numbers, how big the number is, etc. This isn't C.
- Floating-point numbers (`3.1415926535`, `5.2917`, etc)
- Complex Numbers (`1j, 2j`, etc)

String Literals

- Defined by single ('), double ("), or triple ('''', """") quotes. No difference between single and double.
- Triple quotes span multiple lines
- Backslash (\) escapes characters
- List of escape codes in book.
- You'll use \n pretty often, gives you a line feed.

Encoding, Headaches, Python 2 & 3

- Strings are ASCII in python 2 by default.
- This is pretty limiting
- Strings are unicode in python 3.
- This is pretty awesome :)
- You can make strings unicode in python 2 by adding a `u` before the string

Operators, Delimiters, Special Symbols

```
+, -, *, **, /, //, %, <<, >>, &,
|, ^, ~, <, >, <=, >=, ==, !=, <>,
+=, -=, *=, /=, //=, %=, **=, &=,
|=, ^=, >>=, <<=, (), [], {}, , , :
, . , ` , = , ; , ' , " , # , \ , @
```

Documentation Strings

- AKA docstrings
- Explains what a module, class, or function does.
- The book's example is actually wrong
- Use triple quotes """ around strings.

```
def foo():
    """This describes what foo does"""
    print "bar"

print foo.__doc__
```