

Learn **Python** in Ten Minutes

The tutorial: The book



Stavros Korokithakis

Learn Python in Ten Minutes: The tutorial: The book

Stavros Korokithakis



This is a Leanpub book which is for sale at <http://leanpub.com>. Leanpub helps you connect with readers and sell your ebook, while you're writing it and after it's done.

TABLE OF CONTENTS

Learn Python in Ten Minutes 1

Preliminary fluff 1

Properties 1

Getting help 1

Syntax 2

Learn Python in Ten Minutes

Preliminary fluff

So, you want to learn the Python programming language but can't find a concise and yet full-featured tutorial. This tutorial will attempt to teach you Python in 10 minutes. It's probably not so much a tutorial as it is a cross between a tutorial and a cheatsheet, so it will just show you some basic concepts to start you off. Obviously, if you want to really learn a language you need to program in it for a while. I will assume that you are already familiar with programming and will, therefore, skip most of the non-language-specific stuff. Also, pay attention because, due to the terseness of this tutorial, some things will be introduced directly in code and only briefly commented on.

Properties

Python is strongly typed (i.e. types are enforced), dynamically, implicitly typed (i.e. you don't have to declare variables), case sensitive (i.e. `var` and `VAR` are two different variables) and object-oriented (i.e. everything is an object).

Getting help

Help in Python is always available right in the interpreter. If you want to know how an object works, all you have to do is call `help(<object>)`!

Also useful are `dir()`, which shows you all the object's methods, and `<object>.__doc__`, which shows you its documentation string:

```
>>> help(5)
Help on int object:
(etc etc)

>>> dir(5)
['_abs_', '__add__', ...]

>>> abs.__doc__
'abs(number) -> number

Return the absolute value of the argument.'
```

Syntax

Python has no mandatory statement termination characters and blocks are specified by indentation. Indent to begin a block, dedent to end one. Statements that expect an indentation level end in a colon (:). Comments start with the pound (#) sign and are single-line, multi-line strings are used for multi-line comments. Values are assigned (in fact, objects are bound to names) with the `_equals_` sign ("`=`"), and equality testing is done using two `_equals_` signs ("`==`"). You can increment/decrement values using the `+=` and `-=` operators respectively by the right-hand amount. This works on many datatypes, strings included. You can also use multiple variables on one line. For example:

```
myvar = 3
>>> myvar += 2
>>> myvar
5
>>> myvar -= 1
>>> myvar
4
"""This is a multiline comment.
The following lines concatenate the two strings."""
>>> mystring = "Hello"
>>> mystring += " world."
>>> print mystring
```

```
Hello world.  
# This swaps the variables in one line(!).  
# It doesn't violate strong typing because values aren't  
# actually being assigned, but new objects are bound to  
# the old names.  
>>> myvar, mystring = mystring, myvar
```

