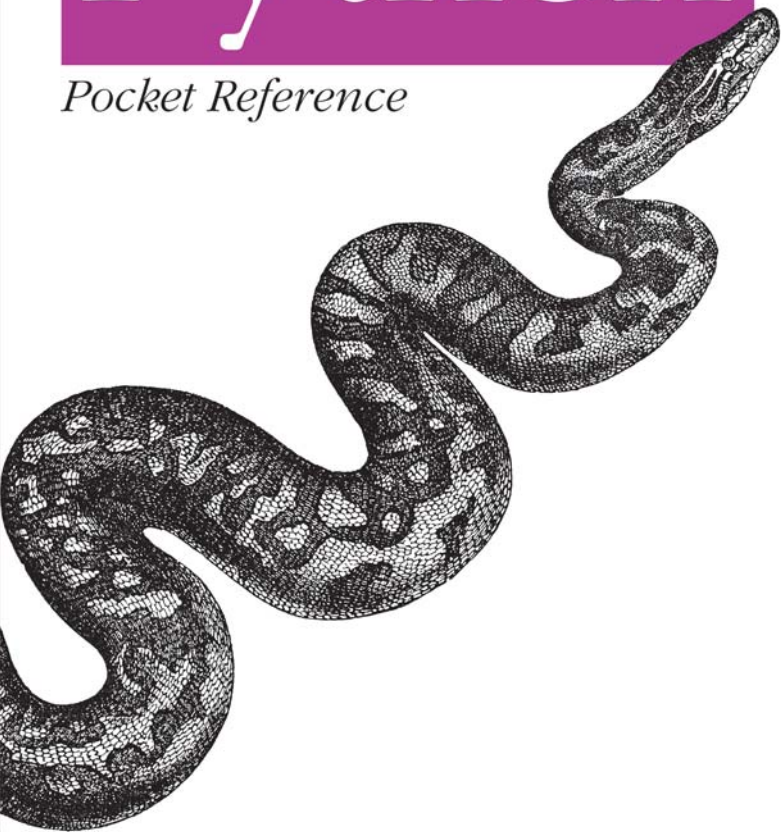


Python In Your Pocket

3rd Edition
Covers Python 2.4

Python

Pocket Reference



O'REILLY®

Mark Lutz

Python Pocket Reference



A python is a muscular snake that kills its prey by squeezing it to the point of suffocation. But the *Python Pocket Reference* is not a book about snakes.

This is a book about Python, the language, which is one of the most elegant, muscular, and widely used scripting languages available these days. With this book, you'll be the one doing the squeezing, as you apply the power of Python to boost the productivity of your programming work.

And speaking of squeezing, that's exactly what expert Python programmer Mark Lutz has done by condensing a ton of important reference information into the *Python Pocket Reference*. In this small but essential book, you'll find syntax for all Python statements, descriptions of built-in types and operators, call specifications for built-in functions and methods, and much, much more. Updated to cover the latest developments in Python 2.4, this is a handy and concise reference you won't want to be without.

Mark Lutz is a software developer, and a Python writer and trainer. He is also the author of *Programming Python* and coauthor of *Learning Python*, both from O'Reilly. Mark has programmed a variety of Python systems, teaches courses about the language, and has been involved with Python since 1992.

"The best little volume on Python that money can buy."

—Guido van Rossum, creator of Python

www.oreilly.com

ISBN 0-596-00940-2

US \$9.95

CAN \$13.95



THIRD EDITION

Python

Pocket Reference

Mark Lutz

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

Python Pocket Reference

by Mark Lutz

Copyright © 2005, 2002, 1998 O'Reilly Media, Inc. All rights reserved.
Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North,
Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales
promotional use. Online editions are also available for most titles
(*safari.oreilly.com*). For more information, contact our corporate/
institutional sales department: (800) 998-9938 or *corporate@oreilly.com*.

Editor: Jonathan Gennick

Production Editor: Claire Cloutier

Cover Designer: Edie Freedman

Interior Designer: David Futato

Printing History:

October 1998:	First Edition.
January 2002:	Second Edition.
February 2005:	Third Edition.

Nutshell Handbook, the Nutshell Handbook logo, and the O'Reilly logo are
registered trademarks of O'Reilly Media, Inc. The *Pocket Reference* series
designations, *Python Pocket Reference*, the image of a rock python, and
related trade dress are trademarks of O'Reilly Media, Inc.

Many of the designations used by manufacturers and sellers to distinguish
their products are claimed as trademarks. Where those designations appear
in this book, and O'Reilly Media, Inc. was aware of a trademark claim, the
designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the
publisher and author assume no responsibility for errors or omissions, or for
damages resulting from the use of the information contained herein.

Contents

Introduction	1
Conventions	2
Command-Line Options	2
Python Options	2
Program Specification	4
Environment Variables	5
Built-in Types and Operators	6
Operators and Precedence	6
Operations by Category	7
Sequence Operation Notes	10
Specific Built-in Types	11
Numbers	11
Strings	12
Unicode Strings	21
Lists	22
Dictionaries	26
Tuples	28
Files	29
Other Common Types	32
Type Conversions	34

Statements and Syntax	35
Syntax Rules	35
Name Rules	36
Specific Statements	38
Assignment	38
Expressions	39
The print Statement	40
The if Statement	41
The while Statement	41
The for Statement	42
The pass Statement	42
The break Statement	42
The continue Statement	43
The del Statement	43
The exec Statement	43
The def Statement	43
The return Statement	46
The yield Statement	46
The global Statement	47
The import Statement	47
The from Statement	49
The class Statement	49
The try Statement	50
The raise Statement	51
The assert Statement	53
Namespace and Scope Rules	53
Qualified Names: Object Namespaces	54
Unqualified Names: Lexical Scopes	54
Statically Nested Scopes	55

Object-Oriented Programming	56
Classes and Instances	57
Pseudo-Private Attributes	58
New Style Classes	58
Operator Overloading Methods	59
For All Types	59
For Collections (Sequences, Mappings)	62
For Numbers (Binary Operators)	63
For Numbers (Other Operations)	66
Built-in Functions	67
Built-in Exceptions	79
Base Classes (Categories)	79
Specific Exceptions Raised	80
Warning Category Exceptions	82
Warnings Framework	82
Built-in Attributes	83
Built-in Modules	84
The sys Module	85
The string Module	91
Module Functions	91
Constants	92
The os System Module	93
Administrative Tools	93
Portability Constants	94
Shell Commands	95
Environment Tools	97
File Descriptor Tools	98

File Pathname Tools	100
Process Control	103
The os.path Module	107
The re Pattern-Matching Module	110
Module Functions	110
Regular Expression Objects	112
Match Objects	112
Pattern Syntax	114
Object Persistence Modules	116
anydbm and shelve Interfaces	117
pickle Interface	118
Tkinter GUI Module and Tools	119
Tkinter Example	120
Tkinter Core Widgets	120
Common Dialog Calls	121
Additional Tkinter Classes and Tools	122
Tcl/Tk-to-Python/Tkinter Mappings	123
Internet Modules and Tools	124
Commonly Used Library Modules	124
Other Built-in Modules	127
The math Module	127
The time Module	128
The datetime Module	129
Threading Modules	129
Binary Data Parsing	130
Python Portable SQL Database API	130
API Usage Example	131
Module Interface	131

Connection Objects	132
Cursor Objects	132
Type Objects and Constructors	133
Python Idioms and Hints	134
Core Language Hints	134
Environment Hints	135
Usage Hints	136
Assorted Hints	137
Index	139

Python Pocket Reference

Introduction

Python is a general-purpose, object-oriented, and open source computer programming language. It is commonly used for both standalone programs and scripting applications in a wide variety of domains, by hundreds of thousands of developers.

Python is designed to optimize developer productivity, software quality, program portability, and component integration. Python programs run on most platforms in common use, including mainframes and supercomputers, Unix and Linux, Windows and Macintosh, Palm OS and Pocket PC, Java and .NET, and more.

This pocket reference summarizes Python statements and types, built-in functions, commonly used library modules, and other prominent Python tools. It is intended to serve as a concise reference tool for developers and is designed to be a companion to other books that provide tutorials, code examples, and other learning materials.

This third edition covers Python Version 2.4 and later. It has been thoroughly updated for recent language and library changes and expanded for new topics. Most of it applies to earlier releases as well, with the exception of recent language extensions.

Conventions

The following conventions are used in this book:

[]

Items in brackets are usually optional. The exceptions are those cases where brackets are part of Python's syntax.

*

Something followed by an asterisk can be repeated zero or more times.

a | b

Items separated by a bar are often alternatives.

Italic

Used for filenames and URLs and to highlight new terms.

Constant width

Used for code, commands, and command-line options, and to indicate the names of modules, functions, attributes, variables, and methods.

Constant width italic

Used for replaceable parameter names in command syntax.

Command-Line Options

```
python [option*]  
[ scriptfilename | -c command | -m module | - ] [arg*]
```

Python Options

-d

Turns on parser debugging output (for developers of the Python core).

-E

Ignores environment variables (such as PYTHONPATH).

- h
Prints help message and exit.
- i
Enters interactive mode after executing a script, without reading the PYTHONSTARTUP file. Useful for postmortem debugging.
- O
Optimizes generated byte-code (create and use .pyo byte-code files). Currently yields a minor performance improvement.
- OO
Operates like -O, the previous option, but also removes docstrings from byte-code.
- Q arg
Division options: -Qold (default), -Qwarn, -Qwarnall, -Qnew.
- S
Doesn't imply "import site" on initialization.
- t
Issues warnings about inconsistent tab usage (-tt issues error instead).
- u
Forces *stdout* and *stderr* to be unbuffered and binary.
- v
Prints a message each time a module is initialized, showing the place from which it is loaded; repeats this flag for more verbose output.
- V
Prints Python version number and exit.

-W *arg*

Functions as warning control; *arg* is *action:message:category:module:lineno*. See warnings module documentation in the Python Library Reference (<http://www.python.org/doc/>).

-x

Skips first line of source, allowing use of non-Unix forms of `#!/cmd`.

Program Specification

scriptfilename

Denotes the name of a Python scriptfile to execute; the main, topmost file of a program, made available in `sys.argv[0]`.

-c *command*

Specifies a Python command (as a string) to execute; `sys.argv[0]` is set to `-c`.

-m *module*

Runs library module as a script: searches for module on `sys.path`, and runs it as a top-level file (e.g., `python -m profile` runs the Python profiler).

-

Reads Python commands from *stdin* (the default); enters interactive mode if *stdin* is a tty (interactive device).

*arg**

Indicates that anything else on the command line is passed to the scriptfile or command (and appears in the built-in list of strings `sys.argv[1:]`).

If no *scriptfilename*, *command*, or *module* is given, Python enters interactive mode, reading commands from *stdin* (and using GNU readline, if installed, for input).

Besides using traditional command lines, you can also generally start Python programs by clicking their filenames in a file

explorer GUI, by calling functions in the Python/C API, by using program launch menu options in IDEs such as IDLE and Komodo, and so on.

Environment Variables

PYTHONPATH

Augments the default search path for imported module files. The format is the same as the shell's PATH setting: directory pathnames separated by colons (semicolons on DOS). On module imports, Python searches for the corresponding file or directory in each listed directory, from left to right. Merged into `sys.path`.

PYTHONSTARTUP

If set to the name of a readable file, the Python commands in that file are executed before the first prompt is displayed in interactive mode.

PYTHONHOME

If set, the value is used as an alternate prefix directory for library modules (or `sys.prefix`, `sys.exec_prefix`). The default module search path uses `sys.prefix/lib`.

PYTHONCASEOK

If set, ignores case in import statements (on Windows).

PYTHONDEBUG

If nonempty, same as `-d` option.

PYTHONINSPECT

If nonempty, same as `-i` option.

PYTHONOPTIMIZE

If nonempty, same as `-O` option.

PYTHONUNBUFFERED

If nonempty, same as `-u` option.

PYTHONVERBOSE

If nonempty, same as `-v` option.

Built-in Types and Operators

Operators and Precedence

Table 1 lists Python’s expression operators. Operators in the lower cells of this table have higher precedence (i.e., bind tighter) when used in mixed-operator expressions without parentheses.

Table 1. Expression operators and precedence

Operator	Description
lambda args: expr	Anonymous function maker.
X or Y	Logical OR: Y is evaluated only if X is false.
X and Y	Logical AND: Y is evaluated only if X is true.
not X	Logical negation.
X < Y, X <= Y, X > Y, X >= Y	Comparison operators ^a .
X == Y, X <> Y, X != Y	Equality operators.
X is Y, X is not Y	Object identity tests.
X in S, X not in S	Sequence membership.
X Y	Bitwise OR.
X ^ Y	Bitwise exclusive OR.
X & Y	Bitwise AND.
X << Y, X >> Y	Shift X left, right by Y bits.
X + Y, X - Y	Addition/concatenation, subtraction.
X * Y, X % Y, X / Y, X // Y	Multiply/repetition, remainder/format, division, floor division ^b .
-X, +X, ~X, X ** Y	Unary negation, identity, bitwise complement, power.
X[i], X[i:j], X.attr, X(...)	Indexing, slicing, attribute references, function calls.
(...), [...], {...}, `...`	Tuple ^c , list ^d , dictionary, conversion to string ^e .