Bash Vs Python

by H. Fosdick

For those who know one of these languages and want to learn about the other.

The Basics

	Bash	Python
Easy to learn, use, and maintain	No (unfriendly syntax)	Yes
Very powerful	Yes	Yes
Open source	Yes	Yes
Portable	Yes	Yes
Runs on all platforms	Yes	Yes
Runs as the OS Shell	Yes	No
Interfaces to tons of tools	Yes	Yes
ANSI or ISO Standard	Yes (POSIX compliant with extensions)	No (de facto standard by PEPs)

Profiles

	Bash	Python
Dialects	Bash (a superset of the Bourne shell)	Version 2.x, Version 3.x, specialized implementations
Unique Usage	* Default scripting language for Linux (including on Windows and mainframe Linux systems) * Sometimes the default for BSD, Oracle Solaris, older Apples, other systems	* Most popular programming language in the world * Most widely taught language * Ships with many OS's
Programming paradigms	Procedural, functional, scripting	Procedural, functional, scripting, object-oriented
OOP: classes, objects, multi-inheritance, polymorphism, encapsulation	Unsupported	Yes
User Group	Free Software Foundation	Python community
Quick Online Lookup	DevHints, LinuxTutorials	QuickRef
Cheat Sheet (printable PDF)	LinuxSimplify, Cheatography	Python 3, Cheatography
Forum	LinuxQuestions.org, Linux.org	Python forums
Further information	Free Software Foundation: GNU Bash	Python.org

Language Comparison

	Bash	Python
Formatting	Free form	Based on indentation rules
Case-sensitive	Yes	Yes
Comments	Start comment with: # Or, use a Here document for commenting out multiple lines	Start line with # or enclose inside triple quotes (""")
Line Continuation	\ (backslash)	\ (backslash) or implicit
Statement Separator	; (semi-colon)	; (semi-colon)
Code Blocks	Define by do - done, if - fi, or case - esac	Define by : (colon) plus indentation
Undefined Variables	Allowed. To error on undefined variables, code: set -u	Undefined variable raises NameError. Circumvent this by assigning special constant None.
Assignment Operators	= += -= *= /= %= Compound Bitwise: &= |= ^= <<= >>=	= += -= *= /= %= //= **= := Compound Bitwise: &= |= ^= >>= <<=
Arithmetic Operators	+ - * / % ** Compound: += -= *= /= %= Evaluate arithmetic expression: $((expression)) Increment/decrement a value: i++ i-- ++i --i	+ - * / % ** // Compound: += -= *= /= %= //= **=
Comparison Operators	Integers: -eq -ne -gt -ge -lt -le Strings: = == != > >= < <= =~	== != > < >= <= Object Identity: is is not
Logical Operators	&& || ! (prefix)	and or not
Concatenation Operators	+= Or, use abuttal: VAR3="$VAR1$VAR2"	+ += * (multiple copies of a string) Concatenate literal strings with blank beween. Use f-strings. Use join or format methods
Bitwise Operators	& | ^ << >> ~ Compound: &= |= ^= <<= >>=	& | ^ << >> ~ Compound: &= |= ^= <<= >>=
Membership Operators	Unsupported	in not in
Regular Expressions	Yes	Use the re module
Built-in Functions	None in the traditional sense, designed to issue shell and line commands	About 70 functions
Data Types	By default, variables are untyped. Or use "declare" to explicitly type them as: -a, -A, -i, -l, -n, -r, -t, -u, -x	A dynamically typed language with these data types: text (str), numeric (int, float, complex), sequence (list, tuple, range), mapping (dict), set (set, frozenset), boolean (bool), binary (bytes, bytearray, memoryview)

Function to Check Data Type	declare -p	type
Collections of Variables	One-dimensional arrays: declare -a array_name	list, tuple, range, set, frozenset, dict, bytearray
Associative Arrays	declare -A array_name	Use a dictionary (dict)
Multidimensional Arrays	Unsupported	Use NumPy library
Stack & Queue Operations	No generalized facility (offers a directory stack with pushd, popd, and dirs)	Yes (queue module and deque from collections)
Decimal Arithmetic	Install and use: bc	Use the decimal module
Flow of Control	until, while, for, break, continue, return, exit	if, for, while, break, continue, pass, return, exit, quit, match-case
Trace Script Execution	Use the -x option. Run the script: bash -x or inside the script: set -x	Use the trace or pdb modules
Terminate Process	exit	exit()
Get User Input	echo -n "Enter your name: " read name	name = input("Enter your name: ")
Exception Handling	trap	try-except-else-finally, raise, with
Standard Exceptions	trap covers 64 signal specs (list them by: trap -l)	SyntaxError, TypeError, NameError, IndexError, KeyError, ValueError, AttributeError, IOError, ZeroDivisionError, ImportError
Run an Operating System Command	Just issue the command string	Use the subprocess module (or older os module)

From: *Bash Reference Manual* version 5.2., *Python Tutorial*, & *Python Language Reference (v 3.12.3)*.