Intro to Python 2020

What makes Python Awesome ?

by Raymond Hettinger
@raymondh

whoami id -un

PSF board member

Python core developer since 2001

Python consultant, speaker, and trainer

Author of the itertools module, set objects, sorting key functions, and many tools that you use every day

@raymondh on twitter

Who uses Python? ps -aux

Google uses C++, Java, Go, and Python interchangeably

YouTube is pure Python

Dropbox:

- Started 2007 with 5 people writing in Python
- Now valued at several billion with several hundred employees

Disney

- At least 800 Python programmers
- Used in all facets of production including animation

JPL and Cisco, Counsyl, HFT, SauceLabs, Django shops, SciPy …

Name an Awesome Language

Why is it awesome?

Surely, something makes it great?

Which languages are Awesome?

- Assembler
- Cobol and Fortran
- Lisp and Scheme
- Forth
- Prolog
- AWK, Lua, Perl and Ruby
- C and Java
- Smalltalk
- Javascript
- Haskell and Scala

What factors made Python Awesome?

- Many languages to choose from
- Many are good in their own way
- What were Python's success factors?

License

- Python was created in the early 1990s by Guido van Rossum
- All Python releases are Open Source
- Most, but not all, Python releases have also been GPL-compatible
- This LICENSE AGREEMENT is between the Python Software Foundation ("PSF"), and the Individual or Organization ("Licensee")
- PSF is making Python available to Licensee on an "AS IS" basis.
- The PSF grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use Python alone or in any derivative version,

Commercial Distributions

ActiveState

Enthought

Anaconda

Zen!

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.

More Zen!

In the face of ambiguity, refuse the temptation to guess.
There should be one -- and preferably only one
-- obvious way to do it.
Although that way may not be obvious at first
unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain,
it's a bad idea.
If the implementation is easy to explain,
it may be a good idea.
Namespaces are one honking great idea
-- let's do more of those!

Community

Newsgroups:

comp.lang.python comp.lang.python.announce

Maillist:

python-dev
python-help
python-tutor

Pycons everywhere

PIGs everywhere

PyPi

- Repository of software for the Python programming language
- There are currently 233,000 packages
- Directly accessible by installers:

```
pip install requests
```

KillerApps and Success Stories

- Zope, Django, Pylons
- Numpy and Scipy
- Bittorrent and Twisted
- YouTube and Google
- Blender
- many, many more

Win32

- Thanks to Mark Hammond

- A huge base of users was opened up

Books

Learning Python

Dive into Python

Python Cookbook

Python in a Nutshell

Python Essential Reference

and many more!

Upcoming book: Python Swallowed Whole

Summary: Context for Success

- License
- Commercial Distributions
- Zen
- Community
- Repository of Modules (PyPi)
- Killer Apps and Success Stores
- Win32
- Books

Besides context, What are we doing right?

What qualities of the language itself make it awesome?

Ease of Learning

What can be learned in three hours:

types: int, str, float, lists, tuples, sets, dict
control: for-loops, while-loops, try/except, if-elif-else
organization: functions, classes, modules

What can be learned in a day:

python idioms
methods on common objects
tour of the library
special methods

Rapid Development Cycle

- Scripting languages are hard to beat for development speed
- Programs are grown organically
- Interactive testing lets people work with their tools early
- Interpreted languages save the edit-compile-run cycle

Economy of Expression

```
import collections, glob, gzip, re, pprint

counter = collections.Counter()
for filename in glob.glob('/Data/logs/*.gz'):
    for line in gzip.open(filename):
        mo = re.search('GET (.*) HTTP', line)
        if mo is not None:
            url = mo.group(1)
            counter[url] += 1

result = counter.most_common(20)
pprint.pprint(result)
```

Example: a five minute script

```
# Search a directory tree for all files with duplicate content

hashmap = {}      # content signature -> list of matching filenames

for path, dirs, files in os.walk('.'):
    for filename in files:
        fullname = os.path.join(path, filename)
        d = open(fullname).read()
        h = hashlib.md5(d).hexdigest()
        hashmap.setdefault(h, []).append(fullname)

pprint.pprint(hashmap)
```

Beauty Counts

- Readability is the #1 mentioned characteristic of why organizations choose Python
- The beautiful appearance on the page directly affects a programmer's sense of joy
- Readability translates directly into maintainability
  (Fixing someone else's code sucks less
  if it is written in Python)

One way to do it

- Most frequently mentioned advantage over Perl
- When a language has too many styles to choose from (each with their own idioms), it becomes a write-only language.
- The standard library provides a common core

Interactive Prompt

- Python experts don’t memorize Python
- They use the interactive prompt often
- It is a killer feature that runs circles around compiled languages
- With VPython or iPython, the interactive prompt becomes an application unto itself
- Embedding interactive prompt is a killer feature for tools like Blender which use Python for direct control

Batteries Included

- Internet protocols: FTP, SMTP, NNTP, …
- Servers, Clients, Parsers
- Compression
- XMLRPC
- Data transport/interchange: JSON, XML, CSV, …
- SqlLite3
- Persistence
- Collections, Itertools, Functools
- Bisect, Heapq
- Datetime, Time, Calendar
- Logging, Unittest, Doctest, Profiling, Tracing, Pdb

Behind the Scenes

- Conservative growth
- We read Knuth so you don’t have to
- Aim for a simple implementation
- No segfaulting – considered a critical bug
- BDFL

Protocols

- Database API specification
  - Common interface to many varieties of SQL
- Hashlib common interface(md5, sha1, etc)
- Compression interface (compress/decompress)
- Conversion protocols (pickle, json, marshal):
  - loads() and dumps()
- WSGI
  - one ring to bind them all
  - connects web servers to web frameworks
- File API (real files, stringIO, etc):
  - open, write, read, readline

Summary:
High level qualities of Python

- Ease of Learning
- Rapid Development Cycle
- Economy of Expression
- Readability and Beauty
- One way to do it
- Interactive Prompt
- Batteries Included
- Behind the Scenes Work
- Common Protocols

Specifics of Python: The Foundation

- Dictionaries and Lists
- Automatic memory management (GC and/or RefCnt)
- Exceptions
- First class functions and classes
- Overridable syntax:

[ ]	__getitem__
.	__getattribute__
+	__add__

Winning Language Feature: Iterator Protocol

- High level glue that holds the language together
- Iterables: strings, lists, sets, dicts, collections, files, open urls, csv readers, itertools, etc
- Things that consume iterators: for-loops, min, max, sorted, sum, set, list, tuple, dict, itertools
- Can be chained together like Unix pipes and filters

Winning Language Feature: List Comprehensions

- Arguably, one of the most loved language features
- Very popular addition to Python
- Derived from notation used in mathematics
- Clean and beautiful
- Much more flexible and expressive than map, filter, and reduce

Generators

Easiest way to write an Iterator

Simple syntax, only adds the YIELD keyword

Remembers state between invocations:

- the stack including open loops and try-statements; the execution pointer; and local variables

Winning Language Features: Genexps, Set comps, and Dict comps

Logical extensions

to list comprehensions and generators
to unify the language

Winning Language Feature: Decorators

- Expressive
- Easy on the eyes
- Works for functions, methods, and classes
- Adds powerful layer of composable tools

Winning Language Feature: Introspection

Adds a capability that is off-limits to static languages

Lets us build tools like inspect, pydoc, help, metaclasses, doctests, unittest discovery, debuggers, etc.

Winning Language Feature: exec, eval, type

Lisp was the king of being able to evaluate itself

Makes it possible to generate new code on the fly (timeit and namedtuple both use this technique)

type() lets code create new classes on the fly

eval() and exec accept dict-like arguments for locals (the popular spreadsheet recipe uses this technique)

Winning Language Feature: With-statement

- Clean, elegant resource management: threads, locks, etc.
- More importantly, it is a tool for factoring code
- Factors-out common setup and teardown code
- Few languages have a counterpart to the with-statement

Abstract Base Classes

Uniform definition of what it means to be a sequence, mapping, etc

Mix-in capability

Ability to override isinstance() and issubclass()

- The new duck-typing, “if it says it’s a duck …”

Winning Language Feature: Indentation

This is how we write our pseudo code

It contributes to Python's clean, uncluttered appearance

It was an audacious move

One of the secrets to debugging C is to run it through a beautifier so the indentation will reflect the actual logic, instead of the programmers intended logic

With Python, the indentation is executable so the visual appearance and actual execution always match.

Winning Language Features: Summary

- Solid foundation: lists and dicts, memory management, exceptions, first-class feautures, and overridable syntax
- Iterator Protocol
- Generators
- List comps, set comps, dict comps, and genexps
- Decorators
- Introspection and Abstract Base Classes
- Exec, eval(), and type()
- With-statement
- Indentation

Anything Else?

Is that all that makes Python awesome?

Time for your thoughts and questions