How Python is Winning New Friends

Steve Holden
CTO, Global Stress Index Limited
sh@felix.com
Introductions

• Programmer since 1967
• Computational scientist by training
• Engineer at heart
• Python user since Python 1.4 (c. 1995)
• Enjoy helping people to learn
I’ve Written about Python
Any Python users out there?
Developments in Computing

SOME HISTORY
Programming Was Hard

- No operating system
- No libraries
- No compilers
- No assemblers
- The painful process of abstraction layering began
1977
Easier to Program

• Assemblers/compilers available
• UNIX starting to emerge as a common base
  – Microprogramming handled hardware complexity
• Storage flexibly handled by the OS
• Networking heading to ubiquity
2020

Whatever it is, it will be complex!
And so to Python

“BUT IT’S [JUST] A SCRIPTING LANGUAGE ...”
The problem with taking offense is that it's really hard to figure out what to do with it after you're done using it. Better to just leave it on the table and walk away. Umbrage untaken quietly disappears.

— Seth Godin —
What’s a “Scripting Language”?

• “First they ignore you; then they abuse you; then they crack down on you and then you win.” – not Mahatma Ghandi
What’s a “Scripting Language”?

• “First they ignore you; then they abuse you; then they crack down on you and then you win.” – *not* Mahatma Ghandi

• “Ridicule is like repression. Both give place to respect when they fail to produce the intended effect.” – *Mahatma Ghandi*
Note to Purists

• *Learners* do not have complex needs
  – Simplicity and consistency are important
  – Execution speed mostly isn’t
• Direct hands-on experience *enables*
• Large resources not required
  – Wide availability and ease of access are critical
The Programming Audience

• Professional software engineers
• Scientists
• Lab technicians
• Teachers and students
• Self-guided learners
• Anyone who wants to control the billions of IoT devices
• ...

WHY DO PEOPLE USE PYTHON?
Easy for Beginners

- Simple Object Model
  - Abstracts memory allocation away
- *Everything* is an object
- Names are references to objects
  - Names live in *namespaces*
  - Objects live in the *heap*
Simple Assignment Semantics

• References keep objects alive
  – Object lifetime management is a non-problem
  – Dangling references therefore impossible

• Data is never copied on assignment
  – Python instead “binds names to values”
The REPL

• Interactively manipulate objects – live!
• Allows *direct* learning
  – Answer your own questions authoritatively
The Ecosystem

HOW MANY PYTHONs?
Jupyter Notebook/Lab

- Heading towards “literate programming”
- Integrates graphical and other outputs with code and commentary in Markdown
- Great way to communicate executable code solutions
Open a CSV file using Pandas

In [5]:
1. import pandas
2. df = pandas.read_csv(' ../data/iris.csv')
3. df.head(20)

Out[5]:
<table>
<thead>
<tr>
<th>sepal_length</th>
<th>sepal_width</th>
<th>petal_length</th>
<th>petal_width</th>
<th>species</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
<td>0.2</td>
<td>setosa</td>
</tr>
<tr>
<td>4.0</td>
<td>3.0</td>
<td>1.4</td>
<td>0.2</td>
<td>setosa</td>
</tr>
<tr>
<td>4.7</td>
<td>3.2</td>
<td>1.3</td>
<td>0.2</td>
<td>setosa</td>
</tr>
</tbody>
</table>

JupyterLab Demo

JupyterLab: The next generation user interface for Project Jupyter

https://github.com/jupyter/jupyterlab

It has been a collaboration between:
- Project Jupyter
- Bloomberg
- Anaconda
Custom computation: Tree summation

As an example of a non-trivial algorithm, consider the classic tree reduction. We accomplish this with a nested for loop and a bit of normal Python logic:

```python
In [8]: total.result()
Out[8]: 9569

def finish(total, single_output):
    c1 / \ c2
    | / \ neighbors merge
    a1 a2 a3 a4 a5 a6 a7 a9
    start
    a0

def total:
    l = L
    while len(l) > 1:
        new_L = []
        for i in range(0, len(l), 2):  # add neighbors
            future = c.submit(add, l[i], l[i + 1])
            new_L.append(future)
        l = new_L

    return l

progress(l)
```

Visualization

- View progress: total, in-memory: 261, processing: 0, ready: 3, waiting: 0, failed: 0
- View memory usage: 515/515
- View tasks: add, double, sort
PyPy

• “Python written in Python”
• Implementation based on Rpython
  – Restricted, compilable language subset
• Gives C-like speeds on regular Python code
  – Retains Python-like clarity
Cython

• Optimising static compiler
• Compiles Python (with C typing information) into C
• Great for wrapping existing C/C++ code in Python
MicroPython

- The *entire* Python 3.4 syntax, including
  - Exceptions
  - with, yield from, etc.
- Also adds 3.5’s `async` and `await`
- Optional machine code!

- Types include `str`, `bytes`, `bytearray`, `tuple`, `list`, `dict`, `set`, `frozenset`, `array.array`, `collections.namedtuple`
- Classes and instances
- And the REPL!
class Crafty:
    def __init__(self, **kws):
        self.__dict__.update(kws)
    c1 = Crafty(a=1, b="two", three=[1, 2, 3])
    print(c1.a, c1.b)
    c2 = Crafty(x="axe", y="sword", z="depth")
    print(c2.a, c2.b)

  Frames
   Global frame
      Crafty
      c1   c2

  Objects
   Crafty class
      hide attributes
   Crafty instance
      a 1
      b "two"
      three set
        1 2 3
   Crafty instance
      x "axe"
      y "sword"
      z "depth"
Summary

MOST OF ALL
Python is FREE and FUN!

• Direct interaction with complex objects
• Ability to hook DIY classes into standard language syntax
• Easy for the motivated student to learn
Possibilities ...

- Robot control
- Toys and games
- Weather stations
- Light patterns
- Science instrumentation/data collection
- Home automation
Final Thoughts

• Computers *don’t* just belong in mathematics
  – Computer programming is *not* computer science
Final Thoughts

• Computers *don’t* just belong in mathematics
  – Computer programming is *not* computer science
• Python gives learners *direct, hands-on* experience
  – *Puts them in control*
Final Thoughts

• Computers *don’t* just belong in mathematics
  – Computer programming is *not* computer science
• Python gives learners *direct, hands-on* experience
  – *Puts them in control*
• Let people find their *own uses* for computers
Questions?

sh@felix.com

@holdenweb

Slides available (soon, promise) at
http://github.com/holdenweb/ACCU2018