Using other code bases with Python

Gibson Bennett

A Brief History

- Began implementation in 1989 by Guido van Rossum
- Inspired by ABC, a language similar to BASIC/COBOL
- One of the most popular languages
  - #5 TIOBE rankings - beat by Java, C, C++, C#
  - #3 IEEE rankings - beat by C,Java
  - #2 most common beginner language - beat by Java

https://commons.wikimedia.org/wiki/File:Guido_van_Rossum_OSCON_2006.jpg

Where is Python used?

- Web and internet development
- Scientific and Numerical Programming
- Education
- General software development

What is Python to Programmers?

- Multi-paradigm programming language
  - Object-oriented
  - Structured
  - Functional
  - Aspect-oriented
- Dynamic (Duck) typing
- High-level, general purpose

Different Python Implementations

- CPython
  - The standard reference implementation using a C bytecode interpreter
- IronPython
  - C#, .NET Framework
- Jython
  - Java Virtual Machine (JVM)
- PyPy
  - Restricted Python (RPython) with Just-In-Time(JIT) compiling

Why not Python?

- Slower than lower-level languages like C/C++, Java
- Lack of static and user-defined variable types
- Lack of scope for variables
  - E.G. a global variable that can be used in all functions
  - This includes lack of differentiating declaration and usage of a variable
- Parallelism is not very elegant
- Pass-by-reference unless stated otherwise

Speed

N-Body simulation using a symplectic integrator, size 50,000,000

Language	Time (sec)	Memory use (MB)
Fortran	19.96	.512
C++	19.37	1.056
C	21.15	1.028
Python3	923.74	8,040

Why Python?

- Short!
  - 3-5 times shorter than Java, 5-10 times shorter than C++
- Simple and readable
- General purpose and flexible
- Development is ~5-10 times faster than C++
- Vast documentation and package support
- Excels as ‘glue’

What is Glue?

- Python code is used to piece together various bits of high-performance code
- Write performance-critical code in C/C++, and Python for higher-level control and customization
- Done through creation of an extension to convert Python data to C/C++ compatible code that allows function calls
- Python serves as a flexible input method for linking codes together

Python and C/C++

- The most popular integration
- Requires writing a C-extension module
  - Difficult, compilation required, No forward compatibility
  - Low-level control, no additional libraries
- This is (partially) how numpy is implemented
- ex) Making a cosine function

```c
/*  Example of wrapping cos function from math.h with the Python-C-API. */

#include <Python.h>
#include <math.h>

/*  wrapped cosine function */
static PyObject* cos_func(PyObject* self, PyObject* args)
{
    double value;
    double answer;

    /*  parse the input, from python float to c double */
    if (!PyArg_ParseTuple(args, "d", &value))
        return NULL;
    /* if the above function returns -1, an appropriate Python exception will
     * have been set, and the function simply returns NULL
     */

    /* call cos from libm */
    answer = cos(value);

    /*  construct the output from cos, from c double to python float */
    return Py_BuildValue("f", answer);
}

/*  define functions in module */
static PyMethodDef CosMethods[] =
{
     {"cos_func", cos_func, METH_VARARGS, "evaluate the cosine"},
     {NULL, NULL, 0, NULL}
};

/* module initialization */
PyMODINIT_FUNC

initcos_module(void)
{
     (void) Py_InitModule("cos_module", CosMethods);
}
```

```python
from distutils.core import setup, Extension

# define the extension module
cos_module = Extension('cos_module', sources=['cos_module.c'])

# run the setup
setup(ext_modules=[cos_module])
```

Cython

- Superset of Python that supports calling of C functions and C types
- Allows seamless Python and C/C++ integration
- Often 100-1000 time faster for numerically-heavy code

```
def f(x):
    return sin(x**2)

def integrate_f(a, b, N):
    s = 0
    dx = (b-a)/N
    for i in range(N):
        s += f(a+i*dx)
    return s * dx
```

```
def f(double x):
    return sin(x**2)

def integrate_f(double a, double b, int N):
    cdef int i
    cdef double s, dx
    s = 0
    dx = (b-a)/N
    for i in range(N):
        s += f(a+i*dx)
    return s * dx
```

Computing Standard Deviation of 1 Million floats

Method	Time (ms)	Compared to Python	Compared to Numpy
Pure Python	183	x1	x0.03
Numpy	5.97	x31	x1
Naive Cython	7.76	x24	x0.8
Optimised Cython	2.18	x84	x2.7
Cython calling C	2.22	x82	x2.7

Cython Wrapping

```
""" Example of wrapping cos function from math.h using Cython. """

cdef extern from "math.h":
    double cos(double arg)

def cos_func(arg):
    return cos(arg)
```

```
from distutils.core import setup, Extension
from Cython.Distutils import build_ext

setup(
    cmdclass={'build_ext': build_ext},
    ext_modules=[Extension("cos_module", ["cos_module.pyx"])]
)
```

Python and R

- R is (arguably) more powerful for statistics and data analytics
- Use rpy2 to allow Python to create and use an R instance, or import individual R packages and routines
  - Also can be done the other direction with rPython
- Use a Jupyter notebook with IR kernel to use R and Python code
  - Not ideal for development
- Use Beaker notebook integrate many languages
  - Similar idea to Jupyter notebook

Python and MATLAB

- Mathworks provides an API to allow Python to call MATLAB functions
- MATLAB is used as a computational engine
- All MATLAB functions, including user-defined functions can be used

```
import matlab.engine
eng = matlab.engine.start_matlab()
tf = eng.isprime(37)
print(tf)
```

```
True
```

Python and Fortran

- Use f2py to create wrappers
- Process is almost identical to C/C++ method

Python and SQL

- Large variety of packages to allow SQL databases to interact with Python
  - Varies depending on the host of the server
  - e.g. pymssql for Microsoft Azure databases
- Allows for modifications and database access

References

https://www.python.org/doc/essays/comparisons/

https://wiki.python.org/moin/IntegratingPythonWithOtherLanguages

https://www.stat.washington.edu/~hoytak/blog/whypython.html

http://www.tiobe.com/tiobe-index/

http://raid6.com.au/~onlyjob/posts/arena/

https://docs.scipy.org/doc/numpy-1.10.0/user/c-info.python-as-glue.html

https://www.toptal.com/python/why-are-there-so-many-pythons

https://benchmarksgame.alioth.debian.org/u64q/measurements.php?lang=python3

https://softwareengineering.stackexchange.com/questions/15468/what-are-the-drawbacks-of-python

https://www.python.org/doc/essays/comparisons/

http://mikelev.in/2011/01/python-programming-language-advantages/

https://notes-on-cython.readthedocs.io/en/latest/std_dev.html

http://cython.org/

https://stackoverflow.com/questions/27464039/why-the-performance-difference-between-numpy-zeros-and-numpy-zeros-like

http://www.kdnuggets.com/2015/12/using-python-r-together.html