PYTHON: A LINGUA FRANCA

USING PYTHON TO BUILD BRIDGES BETWEEN TECHNOLOGIES, BUSINESSES AND PEOPLE

ActiveState®

PYTHON LINGUA FRANCA

There are a multitude of programming languages in use today: dozens of very popular, general-purpose languages with wide user bases; hundreds of specialized languages in niche applications, and more emerging every year. The desire to optimize code for specific kinds of business or technical problems is one of the key factors that has led to the enormous proliferation of specialized programming languages. [1]

While each language has its own inherent strengths and weaknesses, these are of secondary consideration for any business starting a new project. Their first concern must be weighing the advantages of using a specialized language against the difficulty in finding developers with proficiency in a potentially uncommon language.

Programmers have a similar choice: invest time in learning a popular, general-purpose programming language to open up a wider range of employment opportunities, or specialize in a niche language which has fewer, but possibly better paying opportunities.

It's not an all-or-nothing game for either party though. One path out of the conundrum is to use a specialized language where appropriate, and a general-purpose language to connect with other software and communicate with developers in the industry at large. Ideally, this general-purpose programming language would be easy to learn, readily comprehensible, and able to interact with as many specialized languages and technologies as possible. In other words, a common language, or lingua franca.

> ***"There are some technical advantages for having programming languages match the vocabulary of various problem domains. For one thing, such languages are easy to learn for programmers who are familiar with the various domains."***

This paper presents the case that Python is the language best suited to becoming a programmer's lingua franca.

[1] Software Engineering Best Practices: Lessons from Successful Projects in the Top Companies, Capers Jones

PYTHON LINGUA FRANCA

WHAT IS A LINGUA FRANCA?

A lingua franca is a common language used for communications between people who do not share the same native language. The historical "Lingua Franca" was used throughout the Mediterranean for commerce during the Renaissance. Though made up mostly of Italian, its vocabulary included words from Arabic, French, Greek, Portuguese, Spanish and Turkish. If you wanted to do business across the Mediterranean in the 1600s, you used lingua franca or you dealt with a lot of translators.

Today, English is considered the lingua franca for international business, aviation, science, and high tech. Attempts have been made to formalize this type of English usage with several "simplified English" models, such as Basic English. Its limited vocabulary makes it easier to learn for non-native speakers by cutting out most of the notoriously large vocabulary that standard English has accumulated.

Programming languages don't operate in the same environment as human ones, and they don't evolve the same way. They are artificial constructs that need to be "understood" by interpreters and compilers as well as humans. But the need for a common working language is the same.

OTHER CONTENDERS

The need for a common programming language predates Python, and there are a few other languages which have been ubiquitous enough to fulfil the role.

BASIC

Though not now in widespread use, many programmers working today started with BASIC during the first wave of home and kit computers. For many years, it was the de facto first language of hobbyist programmers, and many would argue that the micro-computer revolution could not have happened without it. [2]

> ***"BASIC was derived from academic research tools like beloved old FORTRAN ... It was crude. It was dry. It was unsuitable for the world of the graphic user interface. BASIC had a lot of nasty habits. But it liberated several million bright minds to poke and explore and aspire as never before." [2]***

But BASIC is no longer the common language of programmers. Even though language descendants like VisualBasic .NET

[2] Why Johnny Can't Code, David Brin

are still extremely popular, they do not play the same important part in programming culture that BASIC did in its day.

YOU ALL KNOW C RIGHT?

It's hard to find fault with C as a lingua franca candidate: it's the most widely used programming language of all time; compilers are available for almost all platforms and operating systems, and it has been an important part of most computer science curriculums for decades.

As acknowledged by Joel Spolsky, "Although C is becoming increasingly rare, it is still the lingua franca of working programmers. It is the language they use to communicate with one another, and, more importantly, it is much closer to the machine than "modern" languages that you'll be taught in college like ML, Java, Python, whatever trendy junk they teach these days."

What Spolsky refers to as "trendy junk" is actually a reality for current students, recent graduates, and anyone working in web development. Though C might be an important part of a well-rounded and complete computer science education, for programmers without a background in C, it's syntax can seem idiosyncratic. This is not a serious problem for C programmers, or those willing to invest the time to become C programmers, but for those simply trying to understand a code sample it can be a significant barrier.

JAVA

Java is widely used as a computing lingua franca for some of the same reasons as C. Java is extremely popular – vying with C for top spot depending on which index you consult – widely taught in computer science programs, and cross-platform. [3]

> ***"When I travel I speak English. When I teach I speak Java, and for the same reason: it lets me be understood."*** [3]

Unfortunately, Java is a verbose and repetitive language. It often takes a lot of code to describe even simple algorithms or concepts. Writing the code can be manageable if you have a good IDE to help fill in class declarations and boilerplate, but it's a pain to read, especially for the uninitiated. If your application or API only has a Java programming interface there will be a large body of people who can use it. Unfortunately, for those who don't know the language learning

[3] Java as Lingua Franca, Elliotte Rusty Harold

PYTHON LINGUA FRANCA

enough Java to do anything useful with it is a lot to ask.

What's actually needed is a language that's not just popular but also easy to understand for those that don't already know it.

PYTHON FOR USABILITY

Python was designed to be easy to learn, easy to understand, and most of all easy to read. With readability and re-usability at the core of the language's design, one would expect that becoming proficient in the language would be easier. Though it's not something that's easy to quantify, the experiences of veteran programmers moving to Python from other languages seem to indicate that this is the case.

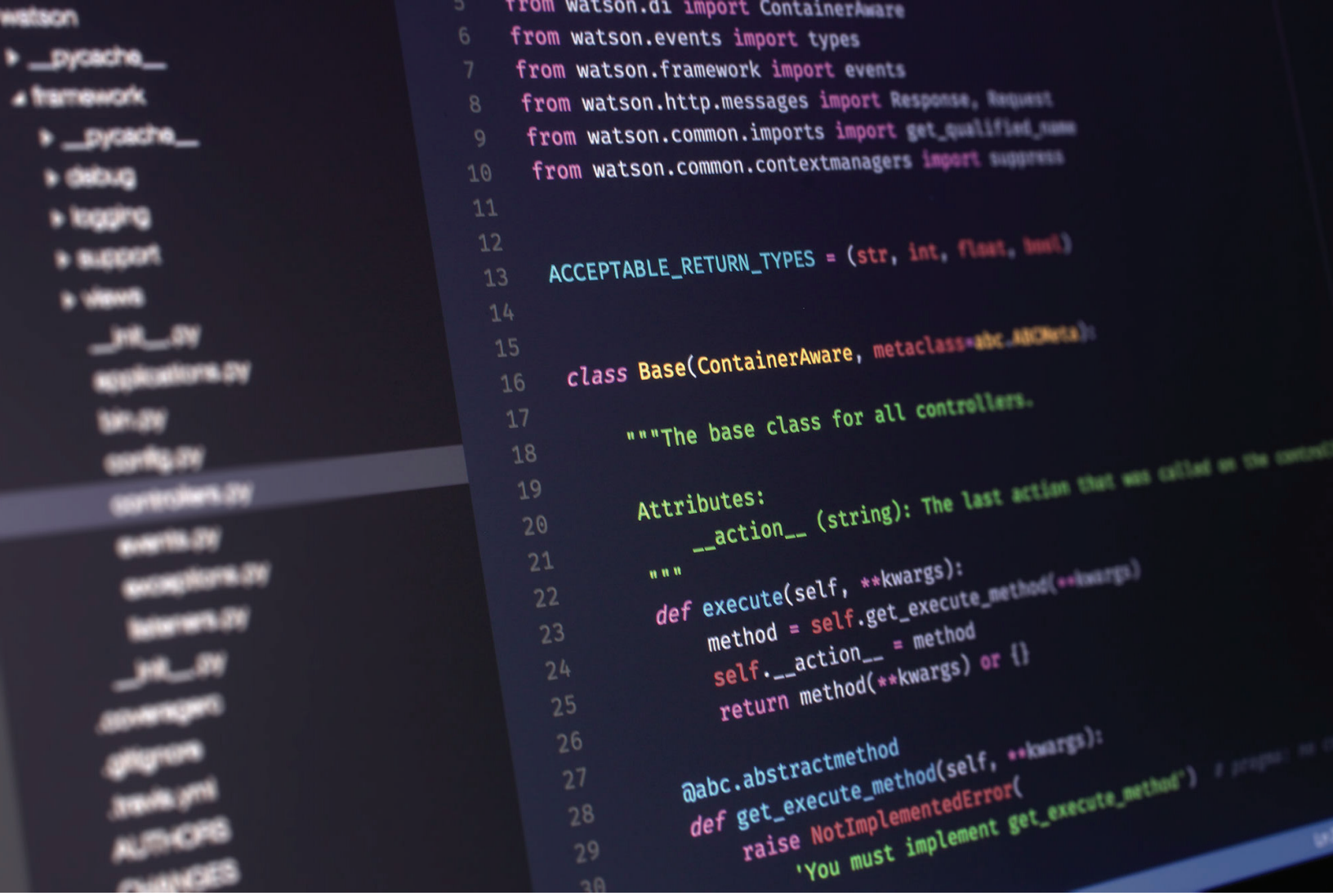
"... I was generating working code nearly as fast as I could type. When I realized this, I was quite startled. An important measure of effort in coding is the frequency with which you write something that doesn't actually match your mental representation of the problem, and have to backtrack on realizing that what you just typed won't actually tell the language to do what you're thinking. An important measure of good language design is how rapidly the percentage of missteps of this kind falls as you gain experience with the language.

When you're writing working code nearly as fast as you can type and your misstep rate is near zero, it generally means you've achieved mastery of the language. But that didn't make sense, because it was still day one and I was regularly pausing to look up new language and library features!

This was my first clue that, in Python, I was actually dealing with an exceptionally good design. Most languages have so much friction and awkwardness built into their design that you learn most of their feature set long before your misstep rate drops anywhere near zero. Python was the first general-purpose language I'd ever used that reversed this process." [4]

> ***"This emphasis on readability is no accident. As an object-oriented language, Python aims to encourage the creation of reusable code. Even if we all wrote perfect documentation all of the time, code can hardly be considered reusable if it's not readable. Many of Python's features, in addition to its use of indentation, conspire to make Python code highly readable." [5]***

[4] Why Python? Eric Raymond
[5] Introduction to Programming Python, Guido Van Rossam



WHAT DOES GOOD LANGUAGE DESIGN LOOK LIKE?

The following three code samples compare a minimal class in C++, Java and Python. The code was pared down from working examples implementing a tree algorithm. The following elements are common to all three languages:

- Comment blocks
- Class definition
- Constructor method definition
- Public methods with arguments
- Class level attributes
- Object instance attributes
- Attribute get and set operations
- Assignment, conditional and return statements
- Object pointers, integers and null values

The first major difference is overall size. Python is half the size of Java and about 40% the size of C++. This is common.

The next thing you'll notice in Python is the absence of setup and typing declarations. The algorithms here are all the same, but Python gets right to the problem at hand.

It is conceivable that the Java and C++ code could actually be generated from the Python code. This approach is the essence of Jython. Using the Python language doesn't necessarily mean using a specific Python interpreter or compiler.

PYTHON LINGUA FRANCA

CODE EXAMPLE 1: C++

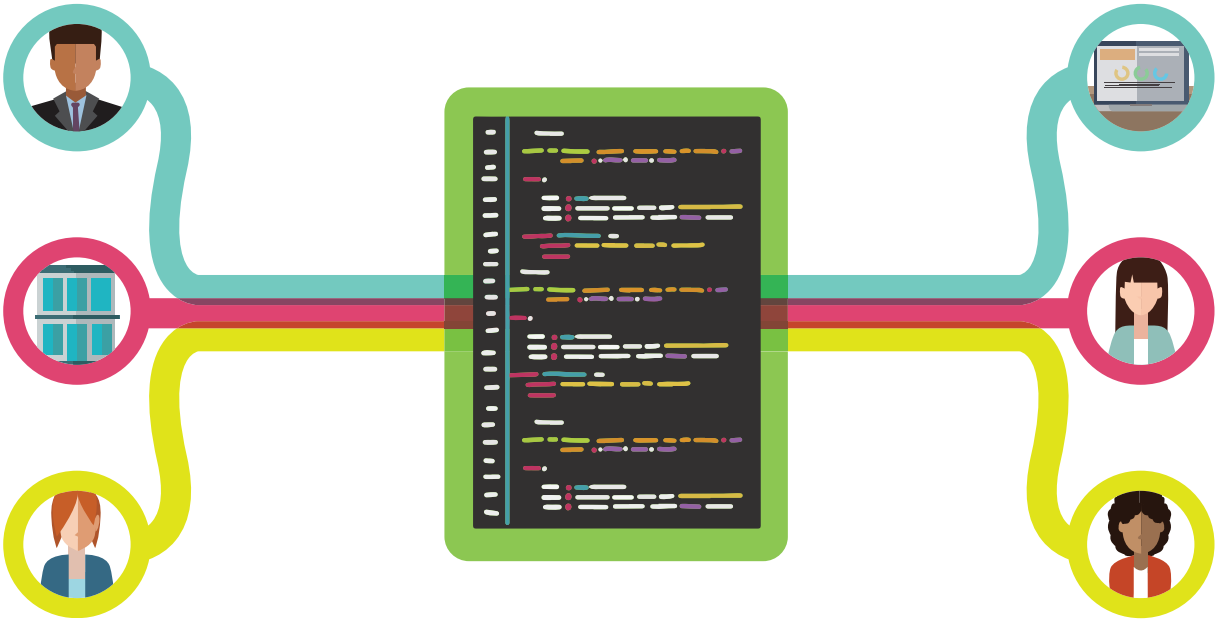
```
#include <iostream>
#include <sstream>
#include <string>
#include <vector>
using namespace std;
template<class t> class redblacktree {
private:
static const int red = 0;
static const int black = 1;

int m_color;
t m_val;
redblacktree *m_left;
redblacktree *m_right;

redblacktree(redblacktree *b) {
        m_val = b->m_val;
        m_left = b->m_left;
        m_right = b->m_right;
        m_color = red;
}

public:
        redblacktree(t x) {
                m_val = x;
                m_left = 0;
                m_right = 0;
                m_color = red;
        }
        const redblacktree *find(const t &key)
const {
                const redblacktree *result = 0;
                if (key == m_val) {
                        result = this;
                }
                else if (key < m_val) {
                        if (m_left != 0) {
                                result = m_left->find(key);
                        }
                }
                else {
                        if (m_right != 0) {
                                result = m_right->find(key);
                }
        }
        return result;
        }
};
```

CODE EXAMPLE 2: JAVA

```
import java.util.*;

public class redblacktree<t extends
comparable<t>> {

        public static final int red = 0;
        public static final int black = 1;

        private int _ _color;
        private t _ _val;
        private redblacktree<t> _ _left;
        private redblacktree<t> _ _right;

private redblacktree(redblacktree<t> b)
{
        _ _val = b._ _val;
        _ _left = b._ _left;
        _ _right = b._ _right;
        _ _color = red;
}
        public redblacktree(t x) {

                _ _val = x;
                _ _left = null;
                _ _right = null;
                _ _color = red;
        }

        public redblacktree<t> find(t key) {

                redblacktree<t> result = null;
                if (key == _ _val) {
                        result = this;
                }
                else if (key.compareto(_ _val) < 0) {
                        if (_ _left != null) {
                                result = _ _left.find(key);
                        }
                }
                else {
                        if (_ _right != null) {
                                result = _ _right.find(key);
                        }
                }
                return result;
        }
}
```

CODE EXAMPLE 3: PYTHON

```
class redblacktree:
        red, black = range(2)

        def _ _init_ _(self, val=none):
                self.left = none
                self.right = none
                self.val = val
                self.color = redblacktree.red

        def find(self, key):
                result = none
                if (key == self.val):
                        result = self
                elif (key < self.val):
                        if (self.left != none):
                                result = self.
                        left.find(key)
        else:
                if (self.right != none):
                        result = self.right.find(key)
        return result
```



DATA SCIENCE PROGRAMMING LANGUAGE OF CHOICE

Python has interoperated well with the statistical language R for over a decade. But in recent years Python has displaced R to become the programming language of choice for data science. There are now hundreds of contributed packages available from the Python Package Index (PyPI) that provide data scientists with:

- Database connectivity to Hadoop, SQL Server, MySQL, MongoDB, Redis, HDF5, etc.
- Data preparation and batching via Pandas, Luigi, Dask, Airflow, etc.
- Algorithm and model building using fundamental packages like NumPy, SciPy, Scikit-learn, as well as machine learning packages like TensorFlow, Theano, NLTK, etc.
- Data visualization using Matplotlib, scikit-image, Bokeh, Seaborn, etc.

While those in the data science realm can benefit from using Python for both the specialized job of creating machine learning models (for example) and the generalized job of operationalizing those models in applications and services, other domains can benefit from Python's flexibility and interoperability, as well.

For example, Python allows programmers to share ideas, techniques and algorithms no matter what technology stack they're using. The Python community has many, many programmers with backgrounds in other languages that have created technical bridges back to those other languages. These bridges can take a few forms:

- Python interpreters implemented in the target language or technology - e.g. Jython is a Python interpreter written in Java; IronPython is written for the .NET framework.
- CPython embedded in a C/C++ application - used by many applications to provide a scripting interface.
- Modules which provide language parsers/interpreters or connectivity to the target language – JPype gives Python programs access to Java class libraries; RPy provides an interface to the R programming language.

PYTHON LINGUA FRANCA

LEARNING PYTHON

The creators of Python and the community of users are clearly proud of its design and actively promote it. This translates into a wealth of resources for learning the language, many of which are available for free. Google has made their own internal Python training program available online. Python itself has excellent documentation and tutorials, and the Python wiki has a comprehensive list of books, tutorials and other learning material in a wide range of (human) languages. Not speaking English is not a barrier to learning Python.

More than any other language, Python also excels at introducing itself to people who are already programmers in other languages. This is a testament to the number of people who have moved to Python from other languages, and created helpful guides along the way.

HOW TO MAKE PYTHON WORK FOR YOU

If you have a new project which could benefit from the simplicity, openness and extensibility of Python, you have a clear starting point. But even if you are currently working with another language, there are still a number of opportunities to use Python in conjunction with your existing software or to expand your personal knowledge of the language for future projects.

In an environment where software providers often try to herd customers into a single technology stack, Python excels at bridging the gaps between technologies, keeping options open to use whichever solution is right for the task at hand.

ACTIVESTATE AND PYTHON

ActiveState has been helping enterprises leverage open source dynamic languages for over 20 years by removing barriers to adoption. As a founding member of the Python Foundation, and the first commercial provider of Python, ActiveState provides standard, enterprise-grade Python distributions for everyone from your data science to application development to IT and administration teams.

Contact ActiveState for a complimentary consultation with ActiveState's open source language experts.

[6] https://www.kdnuggets.com/2015/05/r-vs- python-data- science.html

ActiveState®

website: www.activestate.com
Toll-free in NA: 1.866.631.4581
email: solutions@activestate.com

ABOUT ACTIVESTATE

ActiveState is a leader in providing commercial level open source language distributions. It provides commercial versions of Python, Tcl, Perl, Ruby and Go. More than two million developers and 97% of Fortune 1000 companies use ActiveState open source language builds including CA, Cisco, Pepsi, Lockheed Martin and NASA. To learn more, visit activestate.com