# Building Pythonic Pyramids in Nigeria

*Christopher Thron*

**The International Mathematicians' Union (IMU) began its Visiting Lecturer Program** (VLP) in 2008, inspired by similar programs from the Centre International de Mathématiques Pures et Appliquées (CIMPA); London Mathematical Society; and Norwegian Program for Development, Research and Education. In ten years the VLP has supported lecturers' visits to seventeen different countries in Africa, Central America, the Middle East, and Southeast Asia. Lecturers offer intensive 3–4 week courses that are part of a regular undergraduate or Master's degree program (see bit.ly/2qF8ccs for more information).

## The First Iteration

In Spring 2016, the VLP supported a "Mathematical Software" course which I delivered to second-year students at the University of Ilorin, Nigeria. Materials were adapted (with permission) from the University of Edinburgh's "Interactive Introduction to MATLAB" course (MATLAB.eng.ed.ac.uk/): these included a pdf textbook and YouTube videos (see bit.ly/2J57zk2). Knowing that pirated software is common in Nigeria (and other African countries), I taught the course using Octave (gnu.org/software/octave/), an open-source

*Photo above: Post exam group photo*
*Right: Thron with professors Bamigbola (course facilitator and host) and Adeniyi (head of department)*

software that can run generic MATLAB programs without any modification. Only a few students (maybe 20 percent) had their own laptops or access to computers outside of class, so I resorted to an active learning approach, where the bulk of class time was occupied with hands-on activities. Between the desktops in the lab and students' personal laptops, we ended up with about 3–4 students per computer.

The course was a mixed success. Some students got in lots of programming practice, but many others watched passively. Students often got stuck on the in-class exercises, and would wait for me to come around to their group and help out—but since there were close to 20 different groups, this meant that progress was slow. The final exam scores had a rather low mean and huge standard deviation.

## The Second Iteration

In 2017, the IMU approved my application to return to the University of Ilorin to offer a similar course. I was determined to address some of the previous deficiencies. First was the choice of software. Over the previous few months I had become increasingly aware of the relative benefits of Python, which holds the #1 spot on the 2017 IEEE list of programming languages (bit.ly/2wWgUaB), reflecting Python's widespread use across disciplinary boundaries. Students that do not become mathematicians should be equipped with technical skills that are useful outside of mathematics (and especially in data science). Python is only slightly more difficult to learn than MATLAB/Octave—for example there are some idiosyncrasies with variable types, which are handled automatically by MATLAB. All features and packages necessary for a course in mathematical programming are included within Anaconda (bit.ly/2JVHQMc), a free Python distribution that is available in Windows, Mac, and Linux.

I wanted to keep the active learning format, but to get the course moving at a reasonable pace, I would have to do something about the "stuck students" issue. The best way would be to get class helpers. Inspired by pyramid sales schemes, I envisioned a "Pythonic pyramid": train a small group of advanced students in Python so they could effectively share their knowledge with other students with the hope that this knowledge sharing could become a chain reaction, producing a larger and larger base of Python users. I tried get the pyramid started by running a workshop for selected graduate students who would then help out with my undergraduate class.

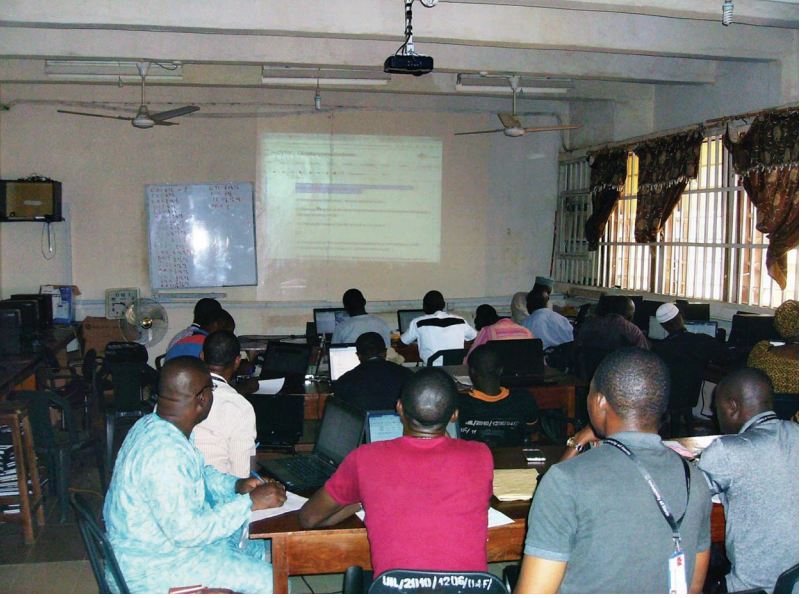
With only three months to develop my curriculum, I adapted Hans Fangohr's online Python resource (bit.ly/2qL8jmX), which had material in a variety of formats including Jupyter notebook. Although the list of topics was good, I ended up completely rewriting and reorganizing the material for my own target audience to cover at least one mathematical application. For this, I used a chapter by Brian Storey on numerical solution of ODEs (bit.ly/2HtAiBH) that is organized around successively more sophisticated MATLAB programs. My final list of topics was:

- Python user interface (Anaconda and Jupyter)
- Algebraic and mathematical operations
- Data types (integer, real, complex) and structures (string, list, tuple)
- Arrays: vectors, and matrices with numpy (Python's numerical math package)
- Visualizing data (plotting) using matplotlib (Python's MATLAB-clone plotting package)
- Control flow (loops and conditional statements)
- Functions
- Application: Numerical solution of differential equations

The hands-on approach which I planned demanded that I provide numerous exercises for the students. Fangohr had labs for his students in addition to class material, but I had only three weeks (and three different sections), so any student programming would have to take place during class time. Designing suitable exercises was a challenging task. If the exercises were too easy, students wouldn't be challenged to think—but if they were too hard, students would sit idle until an instructor came to assist them. Some students were much slower than others, so for each topic I put together a minimal set of exercises to establish basic competency, as well as further exercises for faster students who would otherwise be idle while the rest of the class finished.

## Building the Pyramid

I arrived in Ilorin two days before my graduate seminar was scheduled to begin. The next day was occupied with arranging the schedule and setting up the lab. The lab computers were not networked and had no internet connection, so software was installed by flash drive. This created two main problems. First, my flash drives didn't always get back to me when I loaned them to students to install Anaconda on their personal computers. Second, and more seriously, many computers in Africa are infected with a pernicious "short-



cut" virus that spreads via USB. This virus (which I'd never encountered in the US) is pervasive in west central Africa, probably because so much software is shared via USB. Out of five student laptops, chances are that at least one is infected with the virus. Fortunately, there is a free program called Smadav (www.smadav.net/?lang=en) which provides effective protection—but even Smadav is unable to clean a computer that is already infected.

Besides logistical issues, there were pedagogical challenges as well. My past experience showed that many Nigerian students (not too unlike American students) are content to sit through classes without absorbing anything, then cram furiously before the exam. I had to come up with some strong motivation for students to keep up with and actively participate in the class. I also had to combat a generally lackadaisical attitude towards attendance, and especially towards punctuality—I'd found previously that it was not unusual for the class to start out almost empty, and have students trickle in gradually during the class period. To deal with this, I started each class session with a short 2- or 3-question quiz of about 10 minutes.

The preliminary workshop with graduate students went well with about 30 participants, three times the number that I originally envisioned. Interest level and motivation was high, and the students responded well to personal interactions during periods of hands-on activity. The biggest problem seemed to be that quizzes were too hard, although they were very closely modeled on exercises that students had previously done. It seems there just wasn't enough time for the material to sink in.

## The Implementation

The undergraduate class began the following Monday. The original roster had about 90 students, although attendance at first was somewhat less than this, and by the end of the course the number enrolled surpassed 100. Students were divided into three sections, and each section came two days a week for two 3-hour sessions (morning and afternoon).

Lectures consisted of my showing the Jupyter notebooks on the screen, with code cell outputs removed, revealing the code cell-by-cell, and then showing the output. I was fairly successful at keeping direct presentations at 15 minutes, then having students work on exercises to reinforce their familiarity with features I had just introduced. Acoustics were bad, and the air conditioner was loud, so I had to insist on no talking whatsoever when I was talking (at times I felt like a drill sergeant). I also had to consciously resist my lazy American tendency to blur syllables together. By referring constantly to the material displayed on the screen, it seems I was able to achieve fairly efficient information transfer.

Usually three or four students worked together on a single computer. For the most part, exercises were solved through collaborative effort. Free discussion among students was facilitated by the fact that they had already taken several classes together. As the course progressed, I was encouraged to see students going around to look at other groups' progress, rather than waiting for me to come around to help. Some groups sailed through the exercises on their own, while others needed help at virtually every step. I circulated constantly as they worked, and explained error messages to the students so that they would be able to diagnose errors for themselves. I also emphasized that students needed to think like the computer, and follow the program line by line—this was quite difficult for many students, particularly those who were new to computers.

Adding to the excitement was the perpetual threat of power failure. Fortunately, the projector and lab computers were connected to UPS, so outages didn't cause an immediate standstill. The mathematics department had a (rather noisy) generator, but it was not working for most of the duration of my stay. I had also brought a portable projector with a battery that was good for about an hour. In the end, I only had to cancel part of one class due to power problems.

## Results

Unfortunately, the "pyramid" I had envisioned did not quite materialize. Graduate students were scheduled to assist me in each undergraduate class session, but often no one showed up. Those that did show up didn't always take the task seriously, either talking among themselves, or leaving after a short time. It would have been better if the graduate seminar were smaller and consisted only of students that were committed to building the pyramid. Those times that graduate students came and persisted, I took pains to explain the exercises thoroughly so that they could circulate and provide effective help. I am hopeful that their positive experiences will prepare the ground for future pyramid construction.

As planned, short quizzes were administered at the begin-

ning of each session. Participation in the first quiz was about 40 percent, but reached over 70 percent by the end of the course.

Since I wanted to highlight the mathematical capabilities of Python, many of the quiz questions involved basic trigonometry, complex numbers, modular arithmetic, and so on. Many students seemed unaware of basic mathematical facts such as the relationship between trigonometric functions and the unit circle. The departmental faculty told me that indeed students were having trouble retaining material from earlier classes (not unlike the situation in my home institution). This may be due to poor preparation in the lower grades—students' technical skills in algebra are poor, so they are still struggling with algebra while they should be focusing on new concepts.

Altogether about 10 undergraduate students (besides 20 or so graduate students) installed Anaconda on their own laptops. One student put Python on his Android tablet, but was unable to install the 'numpy' module which enables vector and matrix operations. If an easy installation of Python with 'numpy' and 'matplotlib' (plotting routines) were made available for Android, Python could reach a far wider audience among African students since there are online Python servers that can be accessed via Android, but extensive internet use is too expensive for many students.

All instructional materials (including examples, exercises, and in-class quizzes) were uploaded to the class' Whatsapp group in pdf and Jupyter notebook format. One day was dedicated for review—three review sessions were held, and were well attended. Scores on the final examination ranged from 4 to 100.

In all, the seminar and course went a long way towards increasing awareness of Python among students and faculty, which had been virtually nil before I arrived. The collaborative learning approach took positive advantage of Nigerian students' lively social dynamics, and enabled close personal interaction between instructor and students. As a result, I was able to get a much clearer idea of students' progress, and to address their misunderstandings directly. I also had many opportunities to talk with students about their opinions and aspirations (besides appearing in dozens of Facebook selfies). I have even maintained email contact with some students. Although no course evaluations were distributed for students to share their feedback, some students have expressed their appreciation of my methods. One student emailed me saying "It was one of the best experiences I've had."

Several general areas for improvement also became evident. The Pythonic pyramid did not reach the second floor: no local instructors were sufficiently equipped to teach a Python class on their own. The graduate seminar contained too many students who lacked either the interest or the capability of passing their knowledge on to others. If the pyramid is to gain significant altitude, intensive mentoring should be provided to a handful of instructors who see the value of Python and are willing to commit both to learn and "evangelize" for themselves.

Students need regular access to computing devices that can run Python and preferably devices they own themselves. Python is not too resource-intensive, so if old second-hand laptops were made available to students at low cost it could have a big positive impact. The University of Ilorin has a program for distributing Android tablets to students (bit.ly/2qESPAE); but as mentioned above, there is no straightforward procedure for installing Python on Android with the necessary modules. The problem could also be solved with fast, universal internet (or intranet) access on campus, through which students could access Python servers on their tablets or even smart phones.

In today's world, a mathematically and computationally savvy workforce is a necessary prerequisite for a competitive national economy. Programs like this one can improve students' deficient experience with computers, improve their attitudes towards academics, closing the technology gap between Nigeria and more advanced countries. I am hopeful that persistent pyramid-building efforts may eventually pay off, and students may catch a vision for the possibilities of Python (and mathematical programming in general) to bring Nigeria to new levels of development. ■

*Christopher Thron is associate professor and chair of the Department of Science and Mathematics at Texas A&M University-Central Texas. He has taught courses and workshops in mathematics, statistics, and software in Cameroon, Chad, Nigeria, Sudan, and the People's Republic of China.*