Paper ID #36227

Python for chemical engineers: an efficient approach to teach non-programmers to program

Prof. Gennady Gor, New Jersey Institute of Technology

Dr. Gennady Gor received Ph.D. in theoretical physics from St. Petersburg State University, Russia in 2009. He continued his postdoctoral research in the United States, at Rutgers University, Princeton University and Naval Research Laboratory. In 2016 he joined the Chemical and Materials Engineering department at NJIT as an assistant professor. He authored more than 60 peer-reviewed publications, and is the recipient of the National Research Council Associateship (2014) and the NSF CAREER Award (2020). Dr. Gor's Computational Laboratory for Porous Materials employs a set of modeling techniques, such as Monte Carlo and molecular dynamics simulations, density functional theory and finite element methods, to study materials ranging from nanoporous adsorbents to macroporous polymers and geological porous media.

Python for chemical engineers: an efficient approach to teach non-programmers to program

Gennady Y. Gor
gor@njit.edu
Otto H. York Department of Chemical and Materials Engineering
New Jersey Institute of Technology

Abstract

Modern engineering calculations are hard to imagine without a flexible and efficient programming language. Python is such a language. Python is open source, free, easy to learn, and simple to use. These factors make Python one of the most popular programming languages in the world, highly demanded by employers. However, most undergraduate programming courses for engineers focus on languages other than Python.

I developed and taught a new course “Python for Chemical Engineering Calculations”, which was offered in Spring 2021 as a 3-credit undergraduate elective. The goal of this course was to introduce undergraduate chemical engineering students to Python and demonstrate how it can be used for solving a spectrum of chemical engineering problems. The example problems were taken from the undergraduate chemical engineering curriculum, e.g., from courses such as Chemical Engineering Thermodynamics, Fluid Flow, Kinetics and Reactor Design, etc. Lectures and practical sessions were complemented by six guest lectures delivered by engineers working in industry who demonstrated the use of Python in their jobs.

Not only did the course content differ from a conventional programming course, but the course delivery method was also unconventional. The course was offered in Spring 2021, when all of the classes, including this one, were taught in the synchronous online mode. I used the “flipped classroom” approach, where the students watched the short tutorial videos before each class. Classes typically started from short quizzes based on the videos, after which the class time was utilized for hands-on activities. An online setting with the “break-out rooms” provided a perfect environment to give feedback to students on their code. In addition to quizzes on syntax and in-class coding activities, assessments included a midterm and a final exam. Each of the exams had two parts: a “take-home” coding assignment, and an oral defense of the resulting code.

As seen from the official course evaluations, the course was very well received by the students. They spoke highly of the strong connection between programming and the chemical engineering curriculum, which was impossible to see from taking a generic programming course. Guest lectures was another aspect enjoyed by many. Finally, the flipped classroom which provided a lot of time for in-class activities, appealed to the students a lot. Based on this success, I will utilize

the course materials and the overall approach to revamp the core undergraduate course "Chemical Engineering Computing".

Introduction

Modern engineering calculations are hard to imagine without a flexible and efficient programming language. Python is such a language. Python is open source, free, easy to learn, and simple to use. These factors make Python one of the most popular programming languages in the world, highly demanded by employers. However, most undergraduate programming courses for engineers focus on languages other than Python.

I developed and taught a new course "Python for Chemical Engineering calculations", which was offered in Spring 2021 as a 3-credit undergraduate elective[1]. The goal of this course is to introduce undergraduate chemical engineering students to Python (including the numerical and scientific modules, NumPy and SciPy) and demonstrate how it can be used for solving a spectrum of chemical engineering problems. The example problems were taken from the undergraduate chemical engineering curriculum, e.g., from such courses as Chemical Engineering Thermodynamics, Fluid Flow, Kinetics and Reactor Design, etc. Lectures and practical sessions were complemented by six guest lectures delivered by engineers working in industry who demonstrated the use of Python in their jobs. The guest lectures included such topics as application of Python for data science in the context of chemical engineering jobs.

Not only did the course content differ from a conventional programming course, but the course delivery method was also unconventional. The course was offered in Spring 2021, when all the classes, including this one, were taught in the synchronous online mode. I used the "flipped classroom" approach, where the students watched the short tutorial videos before each class. Classes typically started from short quizzes based on the videos, after which the class time was utilized for hands-on activities. An online setting with the "break-out rooms" provided a perfect environment to give feedback to students on their code. In addition to quizzes on syntax and in-class coding activities, the assessment included a midterm and a final exam. Each of the exams had two parts: a "take-home" coding assignment, and an oral defense of the resulting code. Oral assessments were introduced by the instructor during COVID to help maintaining academic integrity, and demonstrated its efficiency in a different course[2,3].

This paper describes in detail the course structure and the logistics of the course. It discusses the outcome of the first time teaching it, and the thoughts on further development of this course, including potentially adapting it as a required course. The discussion is based on the instructor's observations, as well as on the feedback provided by the students in the official course evaluations.

Methodology: Course Structure

This section presents the course description, similar to what is summarized in a syllabus[1]: topics covered in the course, pre-requisites, learning materials, assessment, etc.

The course covered the following topics:

- Why do we need Python: Motivational Examples.
- Python Environment: Anaconda and IPython. Basics: Variables, Types, Arithmetics.
- More Math, Complex numbers, Strings, Boolean type, Conditional Statements.
- Lists, Sorting and Slicing, Loops. Scripts and Spyder.
- Functions, Recursive Functions, Modules, Math module.
- More on functions, Mutable and unmutable objects, Reading/Writing Files.
- Classes in Python.
- Unit Testing.
- Practice on classes (van der Waals equation of state).
- NumPy: Array Math, Reading and writing data.
- NumPy: Reshaping, Stacking, Slicing.
- Matplotlib: Basic plotting.
- NumPy: Polynomials, Polyfit.
- Matplotlib: more on plotting (scatter plot options, saving files, etc.)
- Calculating fugacity from the van der Waals equation of state.
- Working with data. Tuples, Dictionaries.
- Working with data. Pandas Series and Dataframes.
- SciPy: Physical Constants. Special Functions. Numerical integration.
- SciPy: ODE. Examples from Fluid Flow.
- SciPy: ODE. Examples from Kinetics and Reactor Design.
- CoolProp Calculations for Chemical Engineering Thermodynamics.
- SciPy: Optimization. Examples from Chemical Engineering Thermodynamics.
- Versions. Collaboration. Git.

Note that the details of numerical methods were not covered in this course, as chemical engineering students take a required course "Chemical Engineering Computing", which discusses numerical methods in detail. On the other hand, in order to bring the industry perspective to this course, six guest lectures were given by chemical and software engineers, most of whom work in industry:

- Guest Lecture 1: Speaker – Mr. Michael Pliskin (MS, Computer Science, Engineering Manager at Google) Software Engineering Perspective on Python. Classes in Python. Unit Testing.
- Guest Lecture 2: Speaker – Dr. Jonghun Lee (Ph.D. Chemistry, Research Scientist at Colgate-Palmolive) Materials Science Perspective on Python. Use of Python for Image Processing
- Guest Lecture 3: Speaker – Dr. Christopher Rasmussen (Ph.D. Chemical Engineering, Senior Scientist at Zymergen, Inc.) Chemical Engineering Perspective on Python. Use of Python for New Polymers Development.
- Guest Lecture 4: Speaker – Mr. Taylor Kvist (BS Chemical Engineering, MS Computer Science, Process Contact Engineer at Infineum), Data Science Perspective on Python. Basics of Pandas and Application for Process Calculations.
- Guest Lecture 5: Speaker – Mr. Nicholas Corrente (BS Chemical Engineering, PhD student at Rutgers University), Use of Python for calculation of thermodynamic properties of fluids. CoolProp package.
- Guest Lecture 6: Speaker – Dr. Patrick Robinson (Ph.D. Chemical Engineering, Operations

Business Team Leader at Phillips 66), Chemical/Petroleum Engineering Perspective. Working with data (example of gasoline consumption).

When offering an elective course, choosing the right pre-requisites is a challenge. The course did not have formal pre-requisites, however, it was expected that the students had strong chemical engineering fundamentals (Chemical Process Calculations, Chemical Engineering Thermodynamics, Fluid Flow). Additionally, a strong math background was required (Calculus, Differential Equations).

The following learning materials were provided to the students. For each class which covered new material, the video lectures were recorded by the instructor and made available on Youtube[4]. Each video was made 10-20 minutes long for students' convenience. Additionally, slides with the video lectures materials, example scripts, and solutions of in-class activities were uploaded. The course did not have a required textbook, and there was no required reading other than materials provided in the slides. The aforementioned guest lectures were given live and the recordings were made available to the students, along with the materials provided by the guests.

Students' assessment was based on 19 in-class activities, 12 homework assignments, 9 quizzes (each worth 1% or 1.5% of the total grade) – 45% total, and 2 exams/projects, midterm project worth 25%, and final project worth 30%.

In-class activities included a number of Python programming exercises, varying in complexity. Activities followed the lecture (video) materials and started with basic arithmetic, examples on use of logic and working with strings, lists and loops, etc. After the NumPy and SciPy modules were introduced, in-class activities included more advanced calculations, such as the calculation of thermodynamic properties from various cubic equations of state (Chemical Engineering Thermodynamics), solving ordinary differential equations to describe the processes in bio-reactors (Kinetics and Reactor Design), reading data and performing data reduction (Chemical Engineering Thermodynamics and/or Physical Chemistry), using image processing libraries, etc.

The examples of in-class activities of various complexity are given below.

Activity 4. (Week 3 of the semester)
1. Write a function that solves a quadratic equation analytically (using the general formula).
2. Using the function from 1, solve all equations $x^2+bx+c=0$, where b and c are integer numbers in the range $[-3, 3]$. Print out the results in a readable form.
3. Modify Part 2, so that it prints only the solution when the roots are integer are well.
Hint: use float.is_integer() method.

Activity 9. Problem 2. (Week 6 of the semester).
A problem to practice with reading and writing files and performing manipulations with NumPy arrays.
1. Read the steam table data from the file (txt file provided). Please do not change the file name! (10%)
2. Using the data for enthalpy and for entropy calculate the following thermodynamic properties for liquid and vapor phases:
$G_l = H_l - TS_l$ and $G_v = H_v - TS_v$ (10%)
3. Save the new data into a file with the columns T, P, G_l, G_v (20%)

Activity 15. (Week 12 of the semester).
Write a Python script to plot the solutions for the bioreactor problem (Example from Kinetics and Reactor Design course).
The production of a product P from a particular gram-negative bacteria follows the Monod growth law

$$-r_s = \mu_{\max} \frac{C_c C_s}{K_s + C_s} \tag{1}$$

with $\mu_{\max} = 1\,\mathrm{h}^{-1}$ (maximum specific reaction rate), $K_s = 0.2\,\mathrm{g/l}$ (constant of saturation, which provides 1/2 of the maximum specific growth rate), $Y_{c/s} = 0.5\,\mathrm{g/g}$ (yield coefficient).

$$Y_{c/s} = \frac{\text{mass of new cells formed}}{\text{substrate consumed}} \tag{2}$$

The reaction is to be carried out in a batch reactor with the initial cell concentration of $C_{c0} = 0.1\,\mathrm{g/l}$ and substrate concentration of $C_{s0} = 20\,\mathrm{g/l}$.

$$C_c = C_{c0} + Y_{c/s}(C_{s0} - C_s) \tag{3}$$

Solve the problem numerically and plot $-r_s$, $-r_c$, C_s, C_c as a function of time.

- Rate of change of substrate
$$r_s = \frac{\mathrm{d}C_s}{\mathrm{d}t} \tag{4}$$
- Rate of growth of cells
$$r_c = -Y_{c/s} r_s \tag{5}$$
- Concentration of cells
$$C_c = C_{c0} + Y_{c/s}(C_{s0} - C_s) \tag{6}$$

Homework assignments included two types: independent and tutorial exercises. The former involved writing Python scripts and submitting a solution, which were graded (12 assignments mentioned above). The latter was equivalent to typical reading; the students were asked to watch several tutorial videos, and repeat the exercises from those videos[4]. These assignments were assessed through the in-class quizzes (9 quizzes). The quizzes were multiple choice with most questions asking "what will the following Python code will do?". The quizzes were closed notes, and were proctored using the Respondus Lockdown Browser tool. Overall, the course used the flipped classroom approach[5], where most of the new materials were left to the students to study at home (mostly through tutorial videos) while the in-class time was utilized for discussion of homework and in-class activities. During every class, the instructor used the WebEx break-out rooms to conference with students who needed help or provide feedback on their in-class activities. These break-out rooms included both one-on-one sessions with the instructor and group sessions amongst the students. Some of the time-consuming in-class activities were left to students as homework assignments, so that students earned points for performing part of the work in-class, and additional points for completing the activities at home.

Both the midterm and final exams were in the form of projects. Here is the description of the exam, as provided to the students:

Assignment: You will have to implement a Python code which calculates thermodynamic properties from the cubic EOS. The properties and specific EOS will differ for different students. Irrespective of the specific properties you need to calculate, your code should include the implementation of the following functions:
compr_factor(V, T) – gives the compressibility factor
cubic_form_Z(P, T) – gives the coefficients of the cubic EOS
fug_coef(V, T) – gives the fugacity coefficient
pressure(V, T) – gives the pressure
residual_H(Z, T) – gives the residual enthalpy
roots_Z(P,T) – gives the roots of the cubic polynomial
Also your script should calculate the vapor pressure for the given compound at the given temperature.
Each of you will receive an email with:
1. Type of the EOS,
2. Compound,
3. Temperature for the vapor pressure calculation,
4. Instruction on which thermodynamic property to calculate and to plot.
The script should be implemented, documented, (docstrings for each function/method) and tested. The final script with all the necessary files (additional custom modules, data files for testing, etc.) should be uploaded on Canvas by the deadline. Late submissions will not be accepted, and you will receive zero for the exam!
Grading: The exam will be graded based on the two components:
1. The code itself. The code will be graded for (1) correctness, (2) readability, (3) efficiency.
2. The code presentation. Each student will meet with me (or a proctor) on WebEx to present the project (15-20 minutes). The presentation will be graded based on the answers to the following questions:
(1) I will ask you to explain me what certain parts of your code do
(2) I will ask questions related to the physical meaning of the calculated values
(3) I will ask to implement minor changes in the code, or explain how to modify the code to implement some additional features
(4) I will ask to predict changes in the code behavior upon some hypothetical changes

Exam format: This is an "open book" exam. You can use any resources on the Internet to help you solving your problem. However, must summarize the list of resources you used in the comments to your script (in the header of your script). Furthermore, you cannot copy the code from existing sources, except for the examples I provided. Copied parts of the code will be considered as plagiarism.
You are allowed to discuss your project with no more than two people from the class. However, you must list the names of those students in the comments to your code. Furthermore, you are not allowed to show or share the code, and the assignment itself. This will considered as cheating.
Presentation scheduling: For scheduling your oral slot, please select as many options as you can.

The final exam was given in the same format, but on a different topic – use of data reduction for experimental data on vapor-liquid equilibrium of a binary mixture. It included the use of Pandas, NumPy, SciPy and Matplotlib.

Results

The results of this experimental course can be assessed by two criteria: student performance and course evaluations. The course started with 18 students; two students dropped the course after the first two weeks, while two more students stopped attending towards the end of the semester, missed a significant number of assignments, and received F grades as a result. One more student failed the course because he violated the academic integrity policy on the final exam. This number of withdrawals and fails is not atypical for undergraduate students in our program, and does not exceed the percentage of Ws and Fs in the other undergraduate courses taught by the instructor.

Of the remaining 13 students who received passing grades in the course, 1 student received C+, 2 students received B, and 10 students received A. This number of excellent grades (more than half of the entire class size) noticeably exceeds the number of A grades in other courses taught by the instructor, and suggests that most of the students mastered the material successfully.

Twelve students completed the course evaluations, which allows the instructor to make conclusions regarding the success of the experiment. The students assessed the quality of the course materials as excellent (8) or good (4), and the overall educational value of the course as excellent (10) or good (1), 1 student skipped this question. Most of the students considered the course neither easy nor difficult (8), while 1 student considered it easy, and 3 as difficult. All of the assessments of the instructor were either good or excellent as well.

The students were informed that this course was an experiment, and was taught for the first time, and encouraged to provide detailed narrative comments to help further improve the course. In line with the quantitative assessment of the course, there were no negative comments. Some of these comments, addressing the question "What are the best features of this course?" are provided here. The alphanumeric codes of the courses are replaced with their full titles.

- *The flipped classroom is great. I wouldn't enjoy it as much for a heavy class like Thermo or Fluid Flow but for a lighter coding course with connections to chemical engineering, the flipped style works great. This course has far more educational value than Introduction to Computer Science in C++ (which was required for ChemE's when I took it).*
- *Style of teaching is very good, the flipped classroom works well since the videos are well prepared. Rarely were examples given that were overly difficult or did not make sense to me.*
- *Actually learning industry applicable skills. I also really appreciate the smaller class size. We switched back and forth between video lectures and synchronous lectures, but I personally preferred learning on my own time and bringing my questions to class when we worked on the activities. Also the guest lectures helped break up the semester and keep me engaged. This was a great class.*
- *Learning the basics of programming and using it to solve chemical engineering problems. The guest lecturers have been extremely helpful as they have introduced us to the application of programming in the industry. Also, the activities associated with the lecturers were a good way to implement what was learned.*
- *The structure of the course is good. Videos already recorded allow students to go at*

whatever pace they would like

- *This course has allowed me to learn about a new programming language without any prior knowledge. Unlike other programming courses I have taken, the beginning pace was slow enough so you could grasp the fundamentals from the start.*
- *The overall tone of this course is to solve practical chemical engineering problems with a powerful computer language to supplement ALL of the chemical engineering core courses and provide another skill to add to a resume. In the most recent chemical engineering curriculum, I have taken the required Introduction to Computer Science in C++ course and Chemical Engineering Computing. The issue with the Introduction to Computer Science course is that there is almost zero direct references to chemical engineering and its application. Perhaps this course is still in the curriculum for ABET accreditation reasons, but it truly did not significantly impact my toolbox in problem solving. The Chemical Engineering Computing course was quite relevant because we solved more engineering related problems and learned much more about the numerical methods. However, this course, Python for Chemical Engineering Calculations has provided me with direct applications to common ChE problems seen in Chemical Engineering Thermodynamics and Fluid Flow.*
- *I very much enjoyed the guest lectures; it was one of the few times that I learned about how the course material was actually being used in industry. Also, I think the flipped classroom worked well for this course as with programming, questions tend to arise when you are working on an activity rather than learning the syntax.*
- *The guest lectures and seeing industry applications of Python in chemical engineering are very enjoyable. I was doubtful about the structure where videos were posted for watching outside of class and activities were given during class time, but upon reflection I think it was probably the best way to learn the material.*
- *Flipped classroom and in class assignments that we can ask for help directly.*

Discussion

Although the course was a great success overall, it is worth discussing its potential further improvements, so that in the future it could serve to engineering education even better. The need for some of these changes was identified by the instructor, plus some of recommendations were provided by the students in the course evaluations.

The first modification which is required is a natural transition from the online format of delivery (imposed by COVID-19 constraints) to the in-person format. Even moving to the in-person mode, the class should stay “flipped”. The students will watch the video lectures at home and come to class to work on practical assignments. The hands-on exercises in class can be done and discussed with the instructor and teaching assistants in the in-person format as efficiently as in the online setting.

One of the students, when answering the question “What aspects of the course would you want to see improved?” in the course evaluations, provided the following detailed comment:

I would like to see this course in some shape be required in the curriculum. If it was up to me, I would make this course a required course in the curriculum. I have met countless students in my ChE classes that fear the use of computing software to solve engineering problems that can quickly be solved because of poor experiences in Introduction to Computer Science. This course has given several powerful tools (graphing, curve fitting, optimization, etc) that can be applied in all ChE courses. Most students that I encounter are solving these engineering problems by hand or Excel instead of utilizing the powerful resources that Python offers. This course might sounds similar to Chemical Engineering Computing, but that course only involved barely relevant chemical engineering problems, except for the project at the end of the semester. This course is very relevant to most ChE courses. I believe that students could better perform in the core ChE classes if they took this course.

This comment certainly resonates with the vision of the instructor: in the ideal case, this (or similar) course should be available to all chemical engineering students, rather than to those who chose it as an elective. The question then is where in the curriculum should it be placed? Currently, the chemical engineering curriculum at NJIT, and many others colleges includes Introduction to Computer Science (first year, second semester) and Chemical Engineering Computing (third year, second semester). Unfortunately, the current course cannot serve as a replacement for neither the former, which is just general programming, nor for the latter, which is focused on numerical methods and their implementation.

If the course is left unchanged, pre-requisites and co-requisites have to be introduced. Since the course is heavily based on Chemical Engineering Thermodynamics material, it should be a pre-requisite. The following courses taken by Chemical Engineering students can be co-requisites for the course: Fluid Flow, Physical Chemistry for Chemical Engineers, and Differential Equations. Therefore, in the current form the course cannot be offered earlier than the fourth semester of a typical chemical engineering curriculum. These constraints do not allow the course to serve as a replacement to the required programming course taken by freshman engineering students. However, it can be offered in the second year instead of the existing freshman programming course.

Alternatively, it can be offered in the first year, but will require substantial simplifications. If the current sequence of Introduction to Computer Science and Chemical Engineering Computing courses remain unchanged, much of the material of this course can be utilized for teaching Chemical Engineering Computing, including the Python basics and some of the practical examples. That would still leave time for covering the basics of numerical methods.

Finally, another aspect which can be changed in the course is the use of a textbook, especially if the course is to be adapted as a required course. While a number of Python textbooks exist, none of them appear well-suited. The "Introduction to software for chemical engineers" books[6] includes just one chapter on Python, which is not sufficient. A number of textbooks on Python have appeared in recent years, e.g. "A student's guide to Python for physical modeling"[7] provides a basic introduction, which could serve as a textbook for the freshman programming course, but lacks more advanced examples. A recent book "Learning scientific programming with Python"[8,9] is a very strong and well-structured book, with the main disadvantage that it is tailored for physicists rather than for chemical engineers. Also, the level of presentation is more suitable for graduate students rather than undergraduates. Therefore, there is a demand for a Python programming book written specifically for chemical engineers.

Conclusion

As seen from both the students' performance and the official course evaluations, the course was a big success. Both key features of the course, the focus on chemical engineering applications and flipped classroom, appealed to the majority of the students. Therefore, the course can be offered as an elective in the future. However, the course cannot be included as a core course in the current chemical engineering curriculum, as the typical curriculum already includes two programming/modeling courses which do not leave room for a third one. Some elements of the course, including the Python video lectures and examples, can be utilized for both Introduction to Computer Science and for Chemical Engineering Computing courses. Alternatively, the curriculum can be reorganized so that the course in the current form is offered in the second year instead of Introduction to Computer Science in the first year.

Acknowledgements

The author would like to thank the guest speakers, Michael Pliskin, Dr. Jonghun Lee, Dr. Christopher Rasmussen, Taylor Kvist, Nicholas Corrente, and Dr. Patrick Robinson, for their lectures; and thank all ChE 490 students for providing detailed feedback in their course evaluations and beyond, especially Manisha Kannan, who also helped with proof-reading the manuscript. This work was supported by the National Science Foundation (CBET-1944495).

References

[1] G. Y. Gor. ChE 490: Special Topic – Python Programming for Chemical Engineers. Chemical and Materials Engineering Syllabi., 2021. URL `https://digitalcommons.njit.edu/cme-syllabi/160`.

[2] G. Y. Gor. ChE 342: Chemical Engineering Thermodynamics II. Chemical and Materials Engineering Syllabi., 2020. URL `https://digitalcommons.njit.edu/cme-syllabi/131`.

[3] G. Y. Gor and A. Emelianova. Oral assessments in the online classroom: Example of chemical engineering thermodynamics. *submitted*, 2022.

[4] G. Y. Gor. ChE490: Python for Chemical Engineering Calculations. Video Lectures., 2021. URL `https://www.youtube.com/playlist?list=PLWx-kn1Xf12a77HKLCtZn4E2cjjjoL8N8`.

[5] Jacob Bishop and Matthew A Verleger. The flipped classroom: A survey of the research. In *2013 ASEE Annual Conference & Exposition*, pages 23–1200, 2013.

[6] Mariano Martín Martín. *Introduction to software for chemical engineers*. CRC Press, second edition, 2014.

[7] Jesse M Kinder and Philip Nelson. *A student's guide to Python for physical modeling*. Princeton University Press, 2021.

[8] Christian Hill. *Learning scientific programming with Python*. Cambridge University Press, 2016.

[9] Christian Hill. *Learning scientific programming with Python*. Cambridge University Press, second edition, 2020.