Spreading the Adoption of Python in India: the FOSSEE Python Project

Prabhu Ramachandran[‡§*]

https://youtu.be/6UnuPhTPdnM

Abstract—The FOSSEE (Free Open Source Software for Science and Engineering Education) project (http://fossee.in) is funded by the Ministry of Human Resources and Development, MHRD, (http://mhrd.gov.in) of the Government of India. The FOSSEE project is based out of IIT Bombay and the goal of the project is to eliminate the use of proprietary tools in the college curriculum. FOSSEE promotes various open source packages. Python is one of them.

In this paper, the Python-related activities and initiatives of FOSSEE are discussed. The group focuses on promoting the use of Python in the college curriculum. The important activities of this group include the creation of spoken-tutorials on Python, the creation of 400+ IPython-based textbook companions, an online testing tool for a variety of programming languages, a course akin to software carpentry at IIT Bombay, the organization of the SciPy India conference, and finally spreading the adoption of Python in schools and colleges. The paper discusses how these tools may be used to teach Python in the context of collegiate education and computational science.

Introduction

The FOSSEE project (http://fossee.in) started in 2009 with the goal of helping minimize the use of proprietary software in the college curriculum in India. The project is funded by the Ministry of Human Resources and Development, MHRD (http://mhrd.gov.in) of the Government of India. FOSSEE is part of the MHRD's National Mission on Education through ICT (NMEICT). NMEICT started in 2009 as an initiative to improve the quality of education in India. As part of this mission there have been several initiatives. One important example is the NPTEL project (http://nptel.ac.in) which provides content for over 900 courses at the undergraduate and graduate level (400 web-based and 500 video-based) online. These are proving to be extremely useful all over the country. Other projects include the Spoken Tutorial project (http://spoken-tutorial.org) which has also been previously presented at SciPy 2014 [kmm14]. FOSSEE is one such project that is the outcome of the NMEICT funding.

The FOSSEE project is based out of IIT Bombay and promotes the use of various open source packages in order to help eliminate the use of proprietary packages in the curriculum. A large number of colleges tend to unnecessarily purchase commercial licenses when they really do not need it. The difficulty with using commercial packages to teach basic concepts and computational techniques is well known:

- The packages are typically expensive, the money could be better spent on equipment. This is especially relevant in India.
- Students cannot legally take the software with them home or after they complete their course.
- Academic licenses are not enough as the students end up becoming dependent on the packages after the leave the institution. Furthermore, the packages are even more expensive when used in a commercial setting.

In order to help reduce the dependence on commercial packages, the FOSSEE project's efforts are focused towards training students and teachers to use FOSS tools for their curricular activities. This also requires development efforts in order to either enhance existing projects or fill in any areas where FOSS tools are lacking. There are around ten PIs actively involved in various sub-projects. Some of the most active projects are Scilab, Python, eSim (an Electronic Design Automation tool), OpenFOAM, and Osdag (open source design of steel structures).

After the initial efforts in 2009 and 2010 it was found that some of the initiatives worked and scaled up well whereas others did not. As a result, all of the FOSSEE sub-projects follow a similar structure. Typically each sub-project produces the following output:

- Generates "spoken-tutorials" that new users can use to self-learn a particular software package.
- Organize a crowd-sourced development of "textbook companions" for popular textbooks used in the curriculum. A textbook companion is created by coding every solved example in a text using a particular open source software package like Scilab or Python.
- Support user questions on a forum for the packages that are promoted.
- Develop new software that is useful in a particular domain.
- Support hardware interfacing to encourage open experimentation.
- Migrate labs that use proprietary packages and help them switch to a FOSS equivalent.
- Conduct workshops and conferences to spread the word and teach students and teachers.

Some of these are project specific. For example, the Scilab project is able to perform lab migrations as Scilab is a close

* *Corresponding author: prabhu@aero.iitb.ac.in*
‡ *Department of Aerospace Engineering*
§ *IIT Bombay, Mumbai, India*

equivalent to Matlab and this makes it easier for people to switch to it from Matlab. Kannnan Moudgalya's paper in 2014 [kmm14] discusses in detail the approach and design decisions made by the FOSSEE and spoken-tutorials projects. In particular the paper discusses spoken tutorials, textbook companions, and lab migrations.

The focus of the present paper is to elucidate some the Python-specific activities [FOSSEE-Python] that are of potential direct interest to the SciPy community. The overarching goal of the Python related activities is to help spread the use of Python in the curriculum. In 2016, the scope has expanded to help spread the use of Python in high-schools as well. The focus of this paper is to discuss the approaches that have been taken by the FOSSEE-Python group towards these goals. The lessons and approaches taken by this project are potentially of benefit for similar projects around the world.

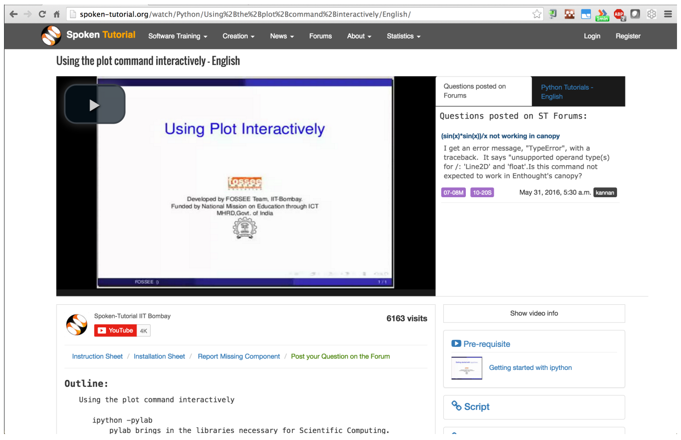
The paper starts by discussing spoken tutorials, which are carefully created screencasts that are well suited for self learning. The paper then discusses a variant of the software carpentry course that has been offered at IIT Bombay. The subsequent section discusses a convenient tool called Yaksh for assessing programming skills of students. The approach taken to create a large amount of user-created documentation in the form of "textbook companions" is then discussed. The paper concludes with some brief information about the SciPy India conference series which is organized by FOSSEE and how that fits in with the overarching theme.

Spoken-tutorials

When the project started in 2009, many live workshops were conducted to teach Python but this proved to be too time consuming and did not scale. There are more than 3000 colleges in the country and live workshops cannot reach all of these institutions. At this time it was felt that preparing self-learning material that students can learn on their own would be much more effective and scalable. A sister project, the spoken-tutorial project (http://spoken-tutorial.org) pioneered the generation and dissemination of spoken-tutorials. A spoken tutorial is basically a carefully designed screencast for a roughly 10 minute duration or less. Any screencast cannot qualify as a spoken-tutorial. A spoken tutorial requires a carefully written script. Notably, a spoken tutorial should be made such that a novice can understand it. The spoken-tutorial project ensures that all new tutorials undergo a novice check to make sure that this is indeed the case. This involves asking a novice to go over the script and ensure that they are able to reproduce the entire script and follow it. This carefully written script allows a spoken tutorial to be dubbed into multiple languages. A series of spoken tutorials can thus be used to effectively teach a programming language or software package. As such, a spoken tutorial is not a substitute for classroom instruction of the traditional kind. It has been most effectively used to teach a programming language or introduce a software package.

The major advantage of the spoken tutorial is that it retains a high quality of instruction, can be used for self-learning, and scales extremely well. In addition, these tutorials can be dubbed into various local languages. The spoken tutorial project has trained over a million students and teachers on a variety of software packages. The project hosts over 700 individual spoken-tutorials. Over 20 different Indian languages are supported.

As part of the Python initiative the FOSSEE Python group has created about 40 spoken tutorials to teach non-CSE undergraduate


spoken-tutorial.org/watch/Python/Using%2Bthe%2Bplot%2Bcommand%2Binteractively/English/
Spoken Tutorial
Software Training
Creation
News
Forums
About
Statistics
Login
Register
Using Plot Interactively
Developed by FOSSEE Team, IIT-Bombay.
Funded by National Mission on Education through ICT
MHRD,Govt. of India
Questions posted on Forums
Python Tutorials - English
Spoken-Tutorial IIT Bombay
YouTube
4K
6163 visits
Instruction Sheet / Installation Sheet / Report Missing Component / Post your Question on the Forum
Show video info
Pre-requisite
Getting started with ipython
Script

Using the plot command interactively - English

Questions posted on ST Forums:

(sin(x)*sin(x))/x not working in canopy

I get an error message, "TypeError", with a traceback. It says "unsupported operand type(s) for /: 'Line2D' and 'float'.Is this command not expected to work in Enthought's canopy?

07-08M 10-20S May 31, 2016, 5:30 a.m. kannan

Outline:

```
  Using the plot command interactively

     ipython -pylab
        pylab brings in the libraries necessary for Scientific Computing.
```

Fig. 1: An example of a Python spoken tutorial. The video can be viewed, an outline of the material is available below the video. An instruction sheet and installation sheet is also available. Prerequisite videos are listed and users can also post questions on a forum.

students how to use Python for their curricular computational tasks. A new set of around 50 tutorials is currently being recorded. The spoken tutorials include tutorials on starting with IPython, plotting with matplotlib, etc. Currently these are only available in English.

Fig. 1 shows a typical Python spoken tutorial as hosted on the spoken-tutorial website. It shows the main screencast video. Below the video is an outline of the tutorial. Information on installation and other instructions is also listed. Users can easily navigate to prerequisite tutorials. In addition, users can post their questions on the forum.

These spoken tutorials can be accessed by anyone and can also be downloaded into a self-contained CD by users. Around 40000 users have gone over this material. Detailed statistics for the various tutorials are available here: http://spoken-tutorial.org/statistics/training/

The FOSSEE team generates the spoken tutorials and the spoken tutorial team coordinates the conduct of workshops where students use this material to teach themselves Python. FOSSEE staff members support these workshops by attending to user questions that may arise.

Spoken tutorials have thus become an effective way to scale up training on open source packages. For a motivated and skilled user, spoken-tutorials and documentation alone are often enough to self-learn. However, this is not enough for the average user. There are many software packages, tools, web sites and books related to computational science. It is never easy for a student (undergraduate or graduate) to choose the right set of packages or practices they should follow. The next section discusses a course that is designed and run by the FOSSEE group at IIT Bombay that helps address this.

The SDES course

SDES is an acronym that stands for Software Development Techniques for Engineers and Scientists. As discussed earlier, the Python group initially focused on teaching Python at various colleges. It was soon felt that students needed to learn how to use Unix shells effectively, use version control, basic knowledge of LaTeX, good software development practices in addition to Python. Students are often unaware of the right set of tools to learn. Most students undergo a basic computer programming course in their first year but this is rarely enough for them to perform their curricular tasks.

In order to fill this need, a course was designed in late 2009. The course is titled Software Development techniques for Engineers and Scientists (SDES). This course takes inspiration from the Software Carpentry Course material [SWC]. However, the course is tailored for undergraduate students. The course is offered at IIT Bombay so students at the undergraduate and graduate levels could take this as part of their course-work. Students can certainly learn this material from several online resources, however, the existence of this course allows students to credit this as part of their course requirements.

The course starts with teaching students on how to use Unix command line tools to carry out common (mostly text processing) tasks. The course then goes on to teach students how to automate typical tasks using basic shell-scripting. The students are then taught version control. The course originally used mercurial, however, this has changed to git. The students are then taught basic and advanced Python. The emphasis is on typical engineering/numerical computations such as those that involve (basic) manipulation of large arrays in an efficient manner. Good programming style is discussed along with debugging and test driven development. They also learn LaTeX and document creation with reStructuredText. The course material is available from github, at http://github.com/FOSSEE/sees.

As part of the evaluation, students pick a software project and attempt to apply all that they have learned. Students are also given many programming assignments to test their ability to program. We have built a convenient online testing tool called Yaksh that is discussed in a subsequent section for this task. This makes the examinations interesting for students and is helpful for instructors to assess student's understanding.

The course has been offered twice thus far and will be offered again in the fall of 2016. The course has been well received by students and is quite popular. The number of students is restricted to about 60 each time. During the last delivery it was felt that the student projects were not done well enough. A more aggressive and systematic approach is needed to push students to work consistently over the duration of the course, rather than in the last minute. It was also found that it is difficult for students and instructors to pick meaningful projects that are neither too trivial or too difficult. For the next delivery, the plan is to encourage students to work systematically on their projects. Studying the git logs of the student project repositories to assess team contribution and systematic work is one approach that is being considered. Instead of always picking new projects, one possibility is to give them an existing project and ask them to improve it.

The SDES course was offered as part of a 1000 teacher training course offered in 2011 at IIT Bombay. This course had over 600 participants who took the course and was well received. Unfortunately, it is not clear how well this course eventually helped teachers and if the teachers went on to teach this material in their colleges.

Teaching the course has generally been enjoyable and rewarding. Students seem to find the course useful and generally continue to use the tools that they have learned. The course is rather demanding from the perspective of assessment and a good team of TAs is necessary. Fortunately, the FOSSEE Python team helps in this regard.

Online test tool: Yaksh

Assessing the programming skills of students is a very important task during training. This is necessary both from the perspective of effective teaching as well as learning. For an instructor, testing early and often is helpful because it provides immediate feedback on which students need help and which of them are doing well. For students, doing well in a test gives them confidence and doing poorly teaches them where they should concentrate harder or get help. Unfortunately, assessment is not usually a pleasant task. Assessment is doubly important when learning a programming language as in India there are students who learn how to program but never write more than a few lines of code. Programming requires practice and encouraging students to program is very important.

For FOSSEE this is also important from the perspective of being able to certify students. The Spoken Tutorial team conducts a large number of workshops all over the country and it would be good if the tests required that students be able to write simple programs at least.

In 2011, the author saw Chris Boesch run a programming contest at PyCon APAC 2011. The contest was entirely online, and users could submit their code and obtained instant feedback. The system was built on top of Google App Engine. This made testing programming lively and enjoyable. The author along with the FOSSEE team have built a Django application to do something similar. The package is called Yaksh, is Open Source, and the sources are available at http://github.com/FOSSEE/online_test. The initial version of Yaksh was used to administer programming quizzes for the online teacher training course based on the SDES course in late 2011. More than 600 simultaneous users took their tests on this interface. This work was presented at SciPy India 2011 [PR11].

Yaksh provides a simple interface for an instructor to create a question paper with mutiple-choice questions (MCQ) as well as full-fledged programming questions. A programming question consists of a problem statement and the user writes the code on the interface. This code is immediately checked against several test cases and any failures are reported directly to the user by providing a suitable traceback. By design, a programming question can be answered many times until the user gets it completely correct. This encourages students to try and submit their answers. An MCQ can only be answered once for obvious reasons.

It was found that the approach of allowing multiple submissions and providing instant feedback instead of the traditional approach where a student would upload the answers on an interface and obtain the grades later to be much more effective. Instant feedback makes the process lively and entertaining for the student. The ability to submit multiple times gives them comfort in that they know that they can gradually fix their code. This makes students less anxious. They also immediately know that their answer is correct if they get it right. This makes a significant difference. Clearly this is not enough to teach all aspects of programming, however, this is a very useful aid.

Yaksh provides a convenient monitoring interface for the instructor which provides, at a glance, information on the students' performance. Each submission of a student is logged and can be seen by the moderator. This is useful for an instructor.

Yaksh works best with Python since it has been used mostly for Python tests but does support multiple other programming languages like C, C++, Java, Bash, and Scilab.

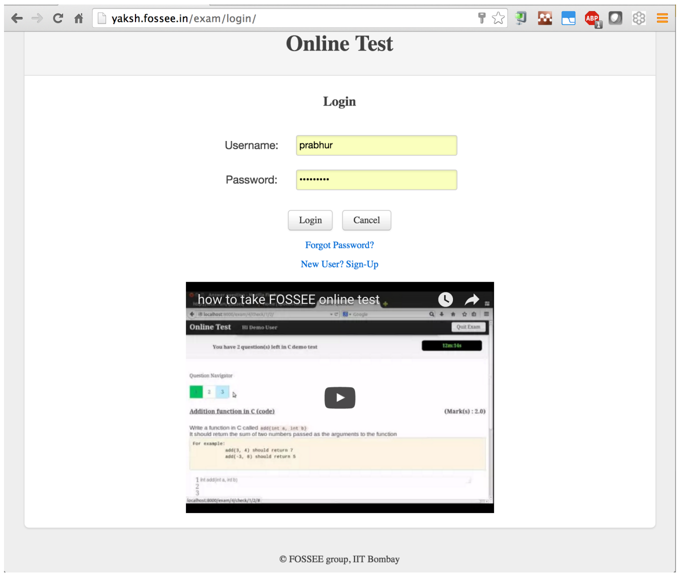
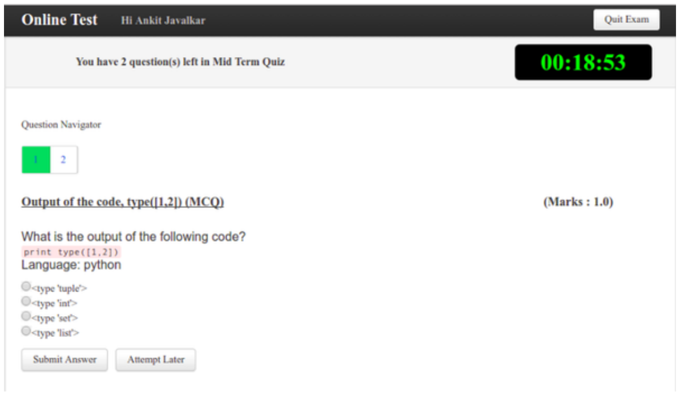


Fig. 2: *The Yaksh application login screen with a video on how one can use it.*



You have 2 question(s) left in Mid Term Quiz

00:18:53

Output of the code, type([1,2]) (MCQ) **(Marks : 1.0)**

What is the output of the following code?

```
print type([1,2])
```

Language: python

- <type 'tuple'>
- <type 'int'>
- <type 'set'>
- <type 'list'>

Fig. 3: *The interface for a multiple-choice question on yaksh.*

Yaksh sandboxes the user code and runs the code as "nobody" when configured to do so. The code execution can also be performed in a docker container. This minimizes any damage a student can do. Since all answers are logged before execution, it is easy to find out if a student has been malicious -- this has never happened in the current usage Yaksh.

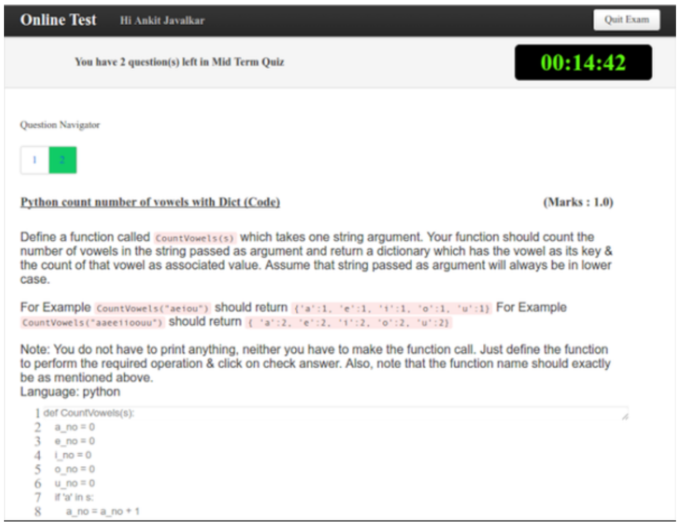
Fig. 2 shows the login screen for Yaksh, which features a small video that demonstrates how the interface can be used. Fig. 3 shows the interface for an MCQ and Fig. 4 shows the interface for a programming question. The top bar shows the time remaining to take the question. A question navigator is provided for students to quickly move between questions.

Fig. 5 shows a typical moderator interface while monitoring a running quiz. The interface shows the number of questions each student has completed. On clicking on a user, all the answers they have submitted are visible.

Installation and running a demo

Yaksh is a Python package and is distributed on PyPI. Yaksh can be installed with pip. When installed, an executable script `yaksh` is created. To setup a demo instance on can run

```
$ yaksh create_demo
```



You have 2 question(s) left in Mid Term Quiz

00:14:42

Python count number of vowels with Dict (Code) **(Marks : 1.0)**

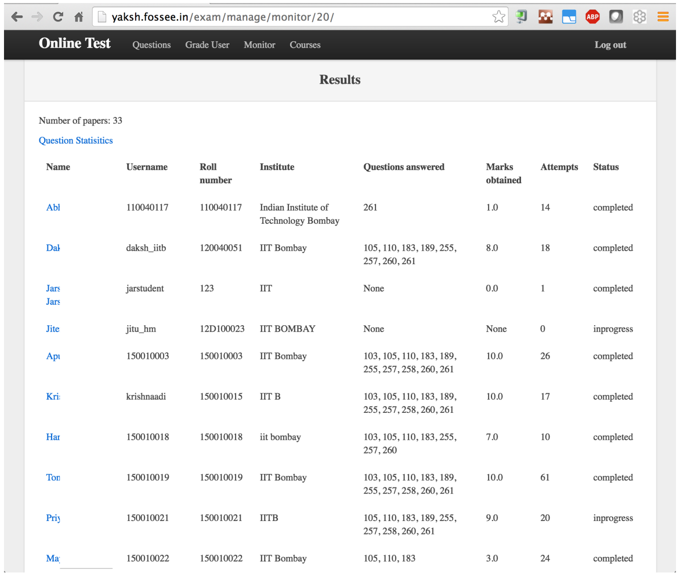
Define a function called `CountVowels(s)` which takes one string argument. Your function should count the number of vowels in the string passed as argument and return a dictionary which has the vowel as its key & the count of that vowel as associated value. Assume that string passed as argument will always be in lower case.

For Example `CountVowels("aeiou")` should return `{'a':1, 'e':1, 'i':1, 'o':1, 'u':1}` For Example `CountVowels("aaeeiioouu")` should return `{ 'a':2, 'e':2, 'i':2, 'o':2, 'u':2}`

Note: You do not have to print anything, neither you have to make the function call. Just define the function to perform the required operation & click on check answer. Also, note that the function name should exactly be as mentioned above.
Language: python

```
def CountVowels(s):
  a_no = 0
  e_no = 0
  i_no = 0
  o_no = 0
  u_no = 0
  if 'a' in s:
    a_no = a_no + 1
```

Fig. 4: *The interface for a programming question on yaksh.*



Results

Number of papers: 33

Name	Username	Roll number	Institute	Questions answered	Marks obtained	Attempts	Status
Abl	110040117	110040117	Indian Institute of Technology Bombay	261	1.0	14	completed
Dak	daksh_iitb	120040051	IIT Bombay	105, 110, 183, 189, 255, 257, 260, 261	8.0	18	completed
Jars Jars	jarstudent	123	IIT	None	0.0	1	completed
Jite	jitu_hm	12D100023	IIT BOMBAY	None	None	0	inprogress
Apu	150010003	150010003	IIT Bombay	103, 105, 110, 183, 189, 255, 257, 258, 260, 261	10.0	26	completed
Kri	krishnaadi	150010015	IIT B	103, 105, 110, 183, 189, 255, 257, 258, 260, 261	10.0	17	completed
Har	150010018	150010018	iit bombay	103, 105, 110, 183, 255, 257, 260	7.0	10	completed
Ton	150010019	150010019	IIT Bombay	103, 105, 110, 183, 189, 255, 257, 258, 260, 261	10.0	61	completed
Priy	150010021	150010021	IITB	105, 110, 183, 189, 255, 257, 258, 260, 261	9.0	20	inprogress
Ma	150010022	150010022	IIT Bombay	105, 110, 183	3.0	24	completed

Fig. 5: *The moderator interface for monitoring student progress during an exam on yaksh.*

This creates a new demo Django project called `yaksh_demo` with a demo database and a couple of users added. One is a moderator and other is an examinee. It also loads a few simple demo questions and a quiz. One can then simply run:

```
$ yaksh run_demo
$ sudo yaksh run_code_server
```

This starts up a server on the `localhost` and also runs the code evaluator as nobody. The server is tested to work on Linux and OS X but not on Windows although technically it should not be difficult to do this. Note that a malicious user could fork bomb the machine in this case as the service is still running on the machine. Resource limiting is possible but not currently implemented.

The above instructions are only for a demo and are not suitable for a production installation as a sqlite database is used in the demo case. More detailed instructions for a production installation are available in the repository.

Design overview

In order to create a quiz the teacher/instructor (also called the moderator) must first create a course. Users can login and register for the course with the instructor's approval. The moderator can add any number of questions to yaksh through the online interface. These can be either MCQ questions or programming questions. The programming questions will require a set of test cases. In the case of a Python programming question, a simple question could be of the form:

```
Write a function called factorial(n) which takes
a single integer argument and returns the
factorial of the number given.
```

The question will also be accompanied with a few test cases of the form:

```
assert factorial(0) == 1
assert factorial(1) == 1
assert factorial(5) == 120
```

As many questions as desired may be created. For other languages assertions are not easily possible but standard input/output based questions are easily handled. More sophisticated test support is also possible (for example one could easily support some form of assertions for C/C++ if a template were used to generate the files). The architecture of yaksh supports this fairly easily.

Questions could also be imported from a Python script. The interface lets users export and import questions. The moderator then creates a quiz and an associated question paper. A quiz may have a pre-requisite quiz and can have a passing criterion. Quizzes have active durations and each question paper will have a particular time within which it must be completed. For example one could conduct a 15 minute quiz with a 30 minute activity window. The students can be allowed to attempt the quiz either once or multiple times as desired. This is often useful when teaching new users. Questions are automatically graded. A user either gets the full marks or zero if the tests fail. In the future yaksh will also support partial grading depending on the number of test cases the code passes.

In terms of the internal design, yaksh is fairly simple.

- The Django app manages the questions, quizzes, users etc.
- A separate code-server process runs as "nobody" to limit the amount of damage malicious code could have. This process runs an XML/RPC server. The Django app creates an XML/RPC `ServerProxy` instance and invokes the code server with the user code and any additional data (like the test cases etc.). This is executed by the server process.
- Unfortunately, XML/RPC can only handle 2 simultaneous connections. Therefore, a pool of these servers is created and managed. The Django app then connects to any available server and executes the code.
- In order to prevent issues with infinite loops, we use the `signal` module to send `SIGALRM` in a finite amount of time. The default is 2 seconds but this can be easily configured.

The code server can be easily run within a docker container and this is also supported by Yaksh. Some documentation for this is also provided in the production README.

In addition to these features yaksh also has an experimental web-API that allows an instructor to utilize yaksh from their own web sites or HTML documents. An instructor could create questions and a question paper from the yaksh interface but have users take the test on say an Jupyter notebook interface. This is still being developed but a proof of concept is available. In order to do this, a user could simply add `yaksh.js` to their HTML and call a few API methods to fetch as well as submit user answers.

Some experiences using yaksh

Yaksh has been used while delivering the SDES course at IIT Bombay. This has worked quite well and is well received by students. As mentioned before, Yaksh has also been used for the online course with over 600 participants and worked quite well. This was however done in 2011 and thereafter has only been used for smaller classes.

Recently, Yaksh was used by the author to teach first year undergraduate students Python as part of a data analysis and interpretation course. Many students were new to programming and a lot was learned about how well this could work.

Yaksh definitely made it much easier to assess the understanding of students. Initially the students were not given tests but were given Jupyter notebooks as well as exercises to solve at home. The assumption was that the students would follow the material since it was done slowly in class. This was not the case. A take-home assignment was given using Yaksh where students would solve simple problems (many taken from the exercise problems that were already given). Surprisingly, many of the students were struggled badly. Even the best students were not able to finish all problems. This showed that a lot more practice was needed. As a result, 7 different quizzes with a few problems each were conducted. After about 5 such quizzes it was found that some students were still having difficulties understanding basic concepts. These were students who were completely new to programming. Around 20 poorly performing students were identified. These students came to a special class and solved 10 problems using yaksh over the course of 2 hours. The monitoring facility was immensely useful as one could walk over to a struggling student and provide assistance or point a TA in their direction. The students all seemed to like the experience and understood the importance of actually programming versus learning the language syntax. Their performance in the subsequent quizzes and assignments improved significantly.

One major lesson learned was that one should ensure that students are tested from the get-go rather than towards the end. This would result in a much smoother experience. Based on the overall experience, it is clear that Yaksh is an effective tool for students and teachers alike.

Plans

Yaksh will continue to be improved based on the needs of the FOSSEE team and that of others. It is hoped that this is also of use to the community. The future goals for the yaksh project are to:

- Clean up and come up with a stable web-API.
- Support the use of Jupyter notebooks for tests.
- Support more programming languages.
- Integrate Yaksh into the spoken-tutorial website in order to help them test students.

Textbook companions

Spoken-tutorials allow FOSSEE to reach out to a larger audience and train students and teachers on the use of FOSS tools and packages. The SDES course is similar to the Software Carpentry effort and offers a full-fledged course that readies students for computational science. Yaksh facilitates both of these by making it easier to test students on their programming skills.

While Python in general and the SciPy project in particular have plenty of good online documentation, this may not always be adequate from the perspective of a beginner. Good quality documentation is not easy to write and requires both expertise as well as the ability to explain things at the level of the user. This is often difficult for a developer who knows almost everything about the package. On the other hand it is not always easy for an inexperienced user to write documentation either.

Students are often interested in taking internships and desire to participate in software projects that are relevant to their area of interest. Is it possible to engage these students in a way where they are able to contribute meaningful documentation in an area of their interest?

Textbook companions offer an interesting approach in this context. As discussed in detail in [kmm14], textbook companions are created by writing Python code for every solved example in a textbook. Students create these textbook companions which are then reviewed by either teachers or reviewers at FOSSEE. This task scales very well as students are eager to take up the task. They already know the subject matter as the textbook is part of their curriculum. The examples are already solved, so they have to convert the solved example into appropriate Python code. Students are given an honorarium and a certificate after their textbooks pass a review. Currently, there are over 530 Scilab textbook companions [STC] created. The Python project has 416 completed books with over 200 textbooks in progress. The Python textbook companions are hosted online at http://tbc-python.fossee.in

The Python Textbook Companions (PTC's) are submitted in the form of IPython notebooks. This is important for several reasons:

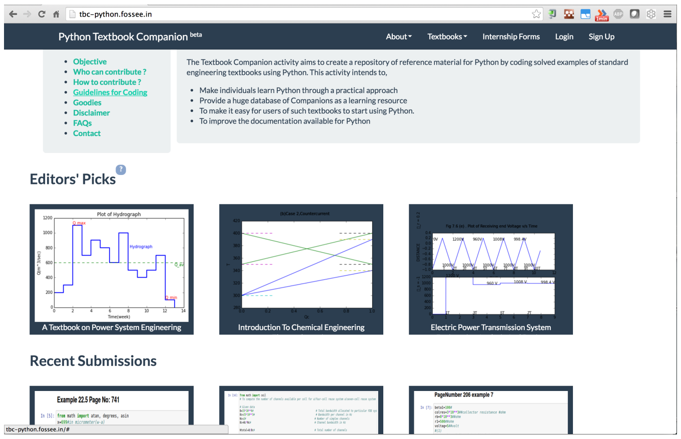
- IPython notebooks allow one to put together formatted HTML, code, and the results in one self-contained file.
- IPython notebooks are easy to render and a HTML listing can be generated.
- The file can also be hosted online and interactively used.
- The huge popularity of the notebook makes this a very useful resource.

The FOSSEE group has also customized the generated HTML such that users can leave comments on the IPython notebooks. This is done by linking disqus comments to each rendered notebook. The disqus API is then queried for any new comments each day and contributors are sent a consolidated email about any potential comments for them to address. This feature is relatively new and needs more user testing.

The submission process and hosting of the IPython notebooks is done using a Django web application that can be seen at http://tbc-python.fossee.in. The code for the interface is also available from github (https://github.com/FOSSEE/Python-TBC-Interface). Once a textbook is reviewed it is also committed to a git repository on github: https://github.com/FOSSEE/Python-Textbook-Companions.

The process works as follows:

1) The student picks a few possible textbooks that have not been completed and informs the textbook companion coordinator.



The Textbook Companion activity aims to create a repository of reference material for Python by coding solved examples of standard engineering textbooks using Python. This activity intends to,

- Make individuals learn Python through a practical approach
- Provide a huge database of Companions as a learning resource
- To make it easy for users of such textbooks to start using Python.
- To improve the documentation available for Python

Fig. 6: *The Django application which hosts the Python textbook companions.*

2) Once a particular book is assigned to the contributor, the student submits one sample chapter which is reviewed by the coordinator.
3) The student then completes the entire book. Each chapter is submitted as a separate IPython notebook.
4) The student also uploads a few screenshots of their favorite notebooks that are displayed on the interface.
5) The submitted code is reviewed and any corrections are made by the contributor.
6) The notebooks are then committed to the git repository.
7) The completed notebooks are hosted by the TBC web application.

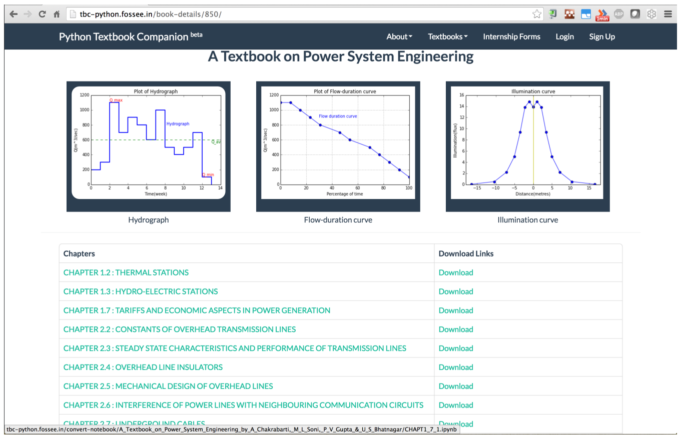
After the textbook is reviewed and accepted the student is sent an honorarium for their work. Fig. 6 shows the main Python TBC interface with information about the project and the editor's picks.

Approximately 3 proposals for new textbooks are submitted each week. Of these, around one is rejected if the book is either a programming language book or it is already completed. Initially many proposals were C or C++ programming books which were being converted to Python. This has since been discontinued and such books are no longer accepted. Of the submissions, around 70% of the submissions are from males, 40% of the submissions are by students, another 40% from teachers, and the remaining 20% from working professionals.

Fig. 7 shows a typical textbook. The IPython notebooks for each chapter can be viewed or downloaded. More information on the book itself can be seen including an ISBN search link for the student to learn more about a book, a link to the actual IPython notebook on github and other details are also available. The entire book can be downloaded as a ZIP file.

Upon clicking a chapter, a typical rendered HTML file is seen. This is seen in Fig. 8. A button to edit the chapter is seen, this will fire up a tmpnb instance which allows users to easily modify and run the code. This makes it convenient to view, modify, and learn the created content. In the figure, one can see an icon for entering comments. This links to a disqus comment field at the bottom of the page. This lists all current comments and allows users to submit new comments on the particular chapter.

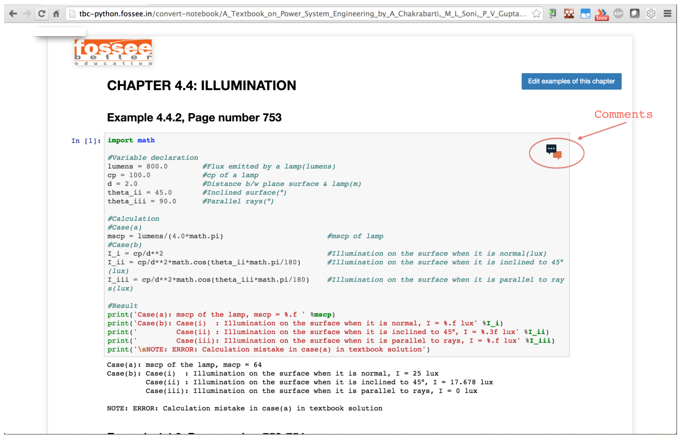
A large number of solved examples are indeed quite simple but there are several that are fairly involved. Some of the nicer textbooks are highlighted in the editor's pick section.



A Textbook on Power System Engineering

Chapters	Download Links
CHAPTER 1.2 : THERMAL STATIONS	Download
CHAPTER 1.3 : HYDRO-ELECTRIC STATIONS	Download
CHAPTER 1.7 : TARIFFS AND ECONOMIC ASPECTS IN POWER GENERATION	Download
CHAPTER 2.2 : CONSTANTS OF OVERHEAD TRANSMISSION LINES	Download
CHAPTER 2.3 : STEADY STATE CHARACTERISTICS AND PERFORMANCE OF TRANSMISSION LINES	Download
CHAPTER 2.4 : OVERHEAD LINE INSULATORS	Download
CHAPTER 2.5 : MECHANICAL DESIGN OF OVERHEAD LINES	Download
CHAPTER 2.6 : INTERFERENCE OF POWER LINES WITH NEIGHBOURING COMMUNICATION CIRCUITS	Download
CHAPTER 2.7 : UNDERGROUND CABLES	Download

Fig. 7: *A typical textbook is shown. The figure shows some screenshots to pique the interest of the casual reader. The Jupyter notebook corresponding to each chapter is listed and can be viewed or downloaded.*


tbc-python.fossee.in/convert-notebook/A_Textbook_on_Power_System_Engineering_by_A_Chakrabarti,_M_L_Soni,_P_V_Gupta...
Edit examples of this chapter
Comments

CHAPTER 4.4: ILLUMINATION

Example 4.4.2, Page number 753

```
In [1]: import math

        #Variable declaration
        lumens = 800.0          #Flux emitted by a lamp(lumens)
        cp = 100.0              #cp of a lamp
        d = 2.0                 #Distance b/w plane surface & lamp(m)
        theta_ii = 45.0         #Inclined surface(°)
        theta_iii = 90.0        #Parallel rays(°)

        #Calculation
        #Case(a)
        mscp = lumens/(4.0*math.pi)                         #mscp of lamp
        #Case(b)
        I_i = cp/d**2                                       #Illumination on the surface when it is normal(lux)
        I_ii = cp/d**2*math.cos(theta_ii*math.pi/180)       #Illumination on the surface when it is inclined to 45°
        (lux)
        I_iii = cp/d**2*math.cos(theta_iii*math.pi/180)     #Illumination on the surface when it is parallel to ray
        s(lux)

        #Result
        print('Case(a): mscp of the lamp, mscp = %.f ' %mscp)
        print('Case(b): Case(i)  : Illumination on the surface when it is normal, I = %.f lux' %I_i)
        print('         Case(ii) : Illumination on the surface when it is inclined to 45°, I = %.3f lux' %I_ii)
        print('         Case(iii): Illumination on the surface when it is parallel to rays, I = %.f lux' %I_iii)
        print('\nNOTE: ERROR: Calculation mistake in case(a) in textbook solution')

        Case(a): mscp of the lamp, mscp = 64
        Case(b): Case(i)  : Illumination on the surface when it is normal, I = 25 lux
                 Case(ii) : Illumination on the surface when it is inclined to 45°, I = 17.678 lux
                 Case(iii): Illumination on the surface when it is parallel to rays, I = 0 lux

        NOTE: ERROR: Calculation mistake in case(a) in textbook solution
```

Fig. 8: *A typical textbook chapter being rendered. The button to edit examples of the chapter fires up a tmpnb instance so users can edit the code and try their changes.*

The Python textbook companion effort of FOSSEE has not been formally announced and advertised in the wider SciPy community. Once announced, the plan is to start to analyze the usage and popularity of this resource. It is still unclear as to how different people are using the notebooks. Some good feedback has been received from the contributors [testimonials] to the project. Many of them have enjoyed creating these notebooks and have benefited by this effort. Some contributor comments are quoted in [kmm14].

In summary, the textbook companions are of interest because:

1) They provide ready-to-use examples of how to apply a given software package or set of tools to a particular problem.
2) They scale well and can be easily crowd-sourced.
3) The scale of the current effort allows one to ask interesting questions, for example "what are the different uses of the FFT in science and engineering?".
4) It provides an interesting alternative to internships and projects for undergraduate students looking to learn and contribute something meaningful.

The texbook companions thus complement the other initiatives of the FOSSEE-Python group.

Scipy India

The SciPy India conference provides an opportunity for those interested in Python to learn of new developments, talk about how they have used Python, meet other interested users/developers and participate in the community.

The Python FOSSEE group has been organizing the SciPy India conference since 2009. Seven conferences have been organized thus far. The conferences have traditionally been held in December. They are largely funded by the FOSSEE project. The project staff manage the local organization almost completely. The conference website is at http://scipy.in

There is an attendance of about 200 people each year. A large number of these are new users. The conference is typically well received and many people are aware of the SciPy community through these efforts. Each year a leading expert in the community is invited to keynote at the conference. The first conference had Travis Oliphant keynote and the conference in 2015 had Andreas Kloeckner as the keynote. Several other important members of the extended SciPy community from India and abroad have spoken at the conference.

Originally, sprints were conducted but this did not prove very effective. The conference now focuses on high-quality tutorials for two days and a single day for the conference itself. Many college professors attend the conference and many go back and encourage their students to use the tools and participate in the future.

Plans for the future

The Python group plans to build on the existing work. The team will continue to generate textbook companions, provide support for the workshops conducted by the spoken-tutorial team, and continue to work on the Yaksh interface. The existing Python spoken tutorials will be updated and new ones will be created as required. These spoken tutorials will also be dubbed to other Indian languages.

In addition the Python group plans to promote the use of Python in the CBSE (Central Board of Secondary Education) school curriculum. The CBSE board has already included Python as an alternative to C++ in the 11th and 12th grade exams. Unfortunately, there is quite a bit of resistance towards this as many teachers are unfamiliar with Python. The plan is to support schools in this initiative over the next year. Textbook companions will be prepared for the school initiative. Spoken-tutorials tailor-made to the school curriculum will also be generated. This is an exciting new development but a significant amount of work is still necessary.

Conclusions

As discussed in this paper, the FOSSEE project has used several interesting approaches to spread Python in India. Spoken tutorials help deliver good-quality self-learning training material to a large audience. The SDES course allows students to learn effective computational skills as part of their curriculum. Yaksh is an open source tool that can be used to effectively test the programming skills of a student. Together, these tools and materials maybe be effectively used by instructors to teach computational tools and programming to a large number of students. The author's experience with using Yaksh while teaching students at different levels has also been shared. It seems that testing students often on their programming is an effective way to have them practice their programming skills and provide quick feedback to the instructor.

Textbook companions offer an interesting alternative to documentation and scales well. The very fact that FOSSEE has helped facilitate around 500+ textbook companions shows that this activity scales and has potential to make a difference.

The FOSSEE Python group has helped spread the use of Python in India. The group has also helped the other sister FOSSEE groups with respect to any Python related support when possible. It is hoped that the code and other material that has been generated is of use to the wider community across the world.

Acknowledgments

FOSSEE would not exist but for the continued support of MHRD and we are grateful to them for this. The project would not be a success without the efforts of the many PIs of the FOSSEE project especially Prof. Kannan Moudgalya of IIT Bombay who also leads the spoken-tutorial project. The author wishes to thank Asokan Pichai who helped shape the FOSSEE project over the first few years. This work would not be possible without the efforts of the many FOSSEE staff members. The past and present members of the project are listed here: http://python.fossee.in/about/ the author wishes to thank them all. The author wishes to thank the reviewers of this manuscript for their suggestions that have made this manuscript better.

REFERENCES

[kmm14] Kannan Moudgalya, Campaign for IT literacy through FOSS and Spoken Tutorials, Proceedings of the 13th Python in Science Conference, SciPy, July 2014.

[FOSSEE-Python] FOSSEE Python group website. http://python.fossee.in, last seen on June 2nd 2016.

[STC] Scilab Team at FOSSEE, Scilab textbook companions, http://scilab.in/Textbook_Companion_Project, May 2016.

[SWC] Greg Wilson. Software Carpentry, http://software-carpentry.org, Seen on May 2016.

[PR11] Prabhu Ramachandran. FOSSEE: Python and Education, Python for science and education, Scipy India 2011, 4th-11th December 2011, Mumbai India.

[testimonials] Python texbook companion testimonials. http://python.fossee.in/testimonials/1/ Seen on Jun 1, 2016