"Python First": A Lab-Based Digital Introduction to Computer Science

Atanas Radenski
Chapman University
Orange, CA 92869, USA
1-714-744-7657
radenski@chapman.edu

ABSTRACT
The emphasis on Java and other commercial languages in CS1 has established the perception of computer science as a dry and technically difficult discipline among undecided students who are still seeking careers. This may not be a big problem during an enrolment boom, but in times of decreased enrolment such negative perception may have a devastating effect on computer science programs and therefore should not be ignored. We have made our CS1 course offerings more attractive to students (1) by introducing an easy to learn yet effective scripting language - Python, (2) by making all course resources available in a comprehensive online study pack, and (3) by offering an extensive set of detailed and easy to follow self-guided labs. Our custom-designed online study pack comprises a wealth of new, original learning modules: extensive e-texts, detailed self-guided labs, numerous sample programs, quizzes, and slides. Our recent student survey demonstrates that students like and prefer Python as a first language and that they also percept the online study pack as very beneficial. Our "Python First" course, originally required for computer science majors, has been so well received that it has been recently approved as a general education science elective, thus opening new recruitment opportunities for the computer science major. Our "Python First" digital pack is published online at http://studypack.com.

Categories and Subject Descriptors
K.3.2 **[Computers & Education]**: Computer and Information Science Education - *computer science education, curriculum*; Computer Uses in Education - *distance learning*

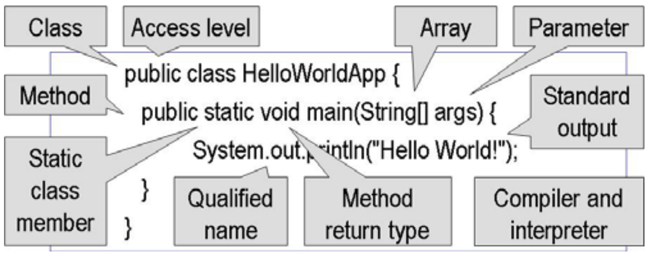
General Terms
Languages, Human Factors

Keywords
CS1, Python, OOP, CS2, Java, online study pack, self-guided lab

1. THE NEED
Today, the majority of introductory computer science courses are based on a commercial language, such as Java or C++. While students with good preliminary background who have already committed themselves to a computing career usually succeed in such courses, many others remain disappointed or even completely fail. One significant problem is that purely commercial languages are not designed for education but are intended to be used for large-scale software development. Commercial applications are complex and so are the programming languages designed exclusively to support them. For beginner programmers, purely commercial languages offer very steep learning curves. Retention rate from Java-based introductory computing courses can be as low as 50%; this has motivated instructors to explore non-traditional approaches to CS1 [3].

The complexity of commercial languages and the high dropout rates from introductory computer science courses have contributed to the perception of computer science as a dry and technically difficult discipline among undecided students who are still seeking careers. In times of decreased enrolment such negative perception may have a devastating effect on the enrolment numbers in computer science programs. In many colleges and universities in the USA, undecided students constitute a very large pool which should be used for internal recruitment in order to maintain sufficient enrollment in computer science programs.

In Java-based courses, beginners inevitably start their study with the top-most concept, the *class*. The problem is that the class concept requires the knowledge of numerous underlying concepts, such as *method, type, parameter, array, access level*. These non-trivial concepts cannot be avoided even in trivial programs, as shown in Fig. 1. In CS1 courses, it is common to introduce new Java concepts in terms of other concepts that are actually defined much later in the course. This awkward necessity is imposed by the complex design of Java and can be confusing to instructors and students alike.



```
public class HelloWorldApp {
  public static void main(String[] args) {
    System.out.println("Hello World!");
  }
}
```

Figure 1. A minimal Java program.

In the not so distant past, introductory computer science education was dominated by languages that were specifically designed for education, such as Pascal, Basic, and Logo. Purely educational languages were simpler and smaller than purely commercial languages and were therefore manageable by everyone who was interested in learning how to program. These languages were designed for student success, and success meant happiness. In the age of the educational languages, learning how to program was fun.

While purely educational languages were appropriate for beginning programming education, they were not particularly suitable for commercial programming. Indeed, educational languages were not meant at all for large-scale software development but were exclusively intended to be used for educational programming. Purely educational languages were unattractive to commercial employers and thus were gradually dropped from most institutions' curricula. While the benefits of some modern educational languages and platforms, such as Karel++, Alice 2, and BlueJ for example are still being explored [1], CS1/CS2 courses are in reality dominated by purely commercial languages.

Apparently, any single programming language that is to be used for introductory computer science education is subject to two *conflicting* requirements. First, such a language should be simple enough so that it can be handled by beginners. Second, such a language is expected to be commercial and object-oriented, which implies significant complexity. At present, these contradictory requirements cannot be fully satisfied by any single language.

2. LANGUAGE CHOICE RATIONALE

2.1 Python's Advantages

Fortunately, there are some interesting languages, such as PhP, JavaScript, and Python, that offer a reasonable compromise between simplicity and practical applicability. Among those, Python has the advantages of being a general purpose language that can be studied interactively and that is actively maintained by a single organization. In contrast to Python, which is a general purpose language, PhP is specialized for server-side Web programming, while JavaScript is specialized for client-side Web programming. Both PhP and JavaScript are not really designed with interactive programming in mind. Also, JavaScript is distributed in non-standard versions. Debugging PhP and JavaScript is difficult.

A unique quality of Python is that it is neither a purely educational language, nor a purely commercial language. Python was originally designed for education but soon gained popularity among practical programmers who became the driving force in its evolution. The Python’s creator, Guido van Rossum, rightfully claims that while Python is a good first choice for teaching, it also serves well as a language for more serious application development [6]. Unlike other languages proposed for teaching to novices, Python is not just a teaching language. It is a language that is suitable for developing real-world applications. Nowadays, the popularity of Python is growing at some high-profile organizations, such as Yahoo, Google, Disney, Nokia, and the US Government. Python regularly ranks among the ten most popular languages [9].

With Python, it is possible to start a CS1 course with simple yet interesting programs that actually do some useful computing and produce some meaningful results from the very beginning. Our preferred first CS1 program is a temperature converter, a program that defines two functions to convert between Fahrenheit and Celsius temperatures (Fig. 2). Such a program can be used to introduce some fundamental yet manageable concepts, such as *program*, *variable*, *expression*, *assignment statement*, and *function*. These concepts are not difficult to explain, partly because students usually have good intuition for some of them, and partly because these concepts are stand-alone and do not use other complex linguistic structures.

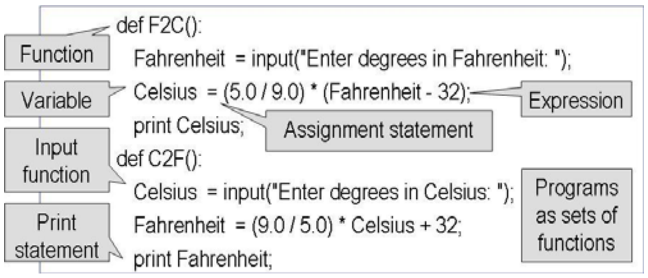

```
def F2C():
    Fahrenheit = input("Enter degrees in Fahrenheit: ");
    Celsius = (5.0 / 9.0) * (Fahrenheit - 32);
    print Celsius;
def C2F():
    Celsius = input("Enter degrees in Celsius: ");
    Fahrenheit = (9.0 / 5.0) * Celsius + 32;
    print Fahrenheit;
```

Figure 2. A feasible first Python program

By design, Python supports a highly interactive programming style that gives students a chance to learn by interactive experiments and exploration. The process of interactive program execution is based on concepts that are intuitively clear and easy to explain (Fig. 3).

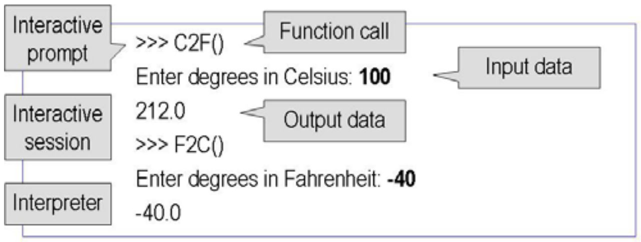

```
>>> C2F()
Enter degrees in Celsius: 100
212.0
>>> F2C()
Enter degrees in Fahrenheit: -40
-40.0
```

Figure 3. Interactive execution in Python

Last but not least, Python is a multi-paradigm language that adequately supports various programming styles: *imperative*, *functional*, and *object-oriented*. Those who learn how to program with Python are well prepared to learn to program in a variety of other languages in subsequent courses.

The Scheme language is also a good choice for a CS1 language because, like Python, it is simple, interactive, allows the discussion of various programming styles, and is really well supported [8]. However, Scheme can be perceived as an educational language that is quite different from the typical commercial languages, such as Java and C++. In comparison to Scheme, Python has the advantage of being easily seen as a practical general purpose language that is not so different from the mainstream commercial languages.

2.2 Middle Way to CS1

The *objects-first* approach in CS1 has gained popularity among educators and has perhaps become a fashion. The idea is to start teaching introductory programming with objects and classes first. Our experience makes us believe that the *objects-first* approach can be difficult for instructors and students alike. Classes do not seem a feasible choice in the beginning of a CS1 course, because they are the most complex structure in a programming language, a structure that builds on knowledge of virtually anything else. Also, starting with classes can significantly delay the study of simpler yet fundamental language elements. We have seen *objects-first* CS1 textbooks that postpone the simple yet indispensable if-statements and while-loops until the second half of the book! Postponing basic control structures until late in a CS1 course may deprive students from writing self-contained, interesting, and satisfying programs. Beginner programmers who do not learn much about control structures until the second part of their course may not find programming very appealing.

With a dynamic language such as Python, it is tempting to try a *statements-first* approach in CS1. This approach may consist of (1) statement-by-statement execution in interactive mode of the Python interpreter and (2) writing statements directly within a module and executing them in file input mode. The learning benefits of interactive programming are indisputable. However, learning how to program by writing modules that are plain sequences of statements, without much use of functions or classes, may be detrimental to the student’s ability to design and write well-structured programs later.

Python is a language that offers a reasonable *middle way* between the *objects-first* and *statements-first* approaches. The *middle way* consists of (1) using functions from the very beginning to encapsulate code and to structure programs and (2) introducing control structures as early as possible, but only as parts of properly designed functions. The *middle way* allows students to write interesting and well-structured programs early in the course, and prepares the ground for classes and objects in the second part of the CS1 course.

2.3 The Need for Java in CS2

While Python is an excellent choice as a first language, Java is a great choice for a second language. Java should be studied as a second language for basically the same reasons it should be avoided as a first language:

- Classes are required for everything students do in Java. With hybrid languages, such as C++, PhP, or Python, students can avoid classes and actually never learn OO design.
- Java is a mainstream commercial language.

As a commercial language, Java is favored by employers. In addition, Java is needed for upper-level courses.

To address the contradictory requirements to CS1/CS2: *simplicity* versus *commercial object-orientation,* we have adopted a "Python First, Java Second" approach to CS1/CS2. The guiding idea is to start introductory computer science education with a gentle Python-based first course and then continue with a comprehensive Java-based second course. Our Java-based CS2 course has the same extensive online component [5] as our Python-based CS1 course, to the extent that lectures are declared as optional.

3. IMPLEMENTATION

We have implemented a "Python First, Java Second" CS1/CS2 course sequence at Chapman University (est. 1861), California, USA, beginning in the 2004/05 academic year.

At Chapman University, the CS1/CS2 courses enroll a diverse student body. Both courses are required for students majoring in Computer Science, Mathematics, and Computer Information Systems. Computer Science majors specialize in either Software Design Emphasis or Integrated Circuit Design. Within the Mathematics major, the university offers a Joint Engineering Program with the University of California, Irvine.

3.1 "Python First" Course

The focus of this paper is on our "Python First" CS1 course. In this course, Python programming is used as a vehicle for a technical introduction to computing. The course is not exclusively targeted on "learning how to program in Python" but is actually focused on "programming in Python to learn about computing".

The "Python First" course is taught around a monotonically increasing sequence of subsets of the Python language. Each subset is self-contained and depends only on previously covered subsets. This approach ensures that students are exposed to new concepts and techniques gradually, without disturbing references to important concepts that are to be studied later.

Technically, the "Python First" course content is organized as a sequence of topics, as outlined below.

1. Preliminaries
2. Computing Basics
    - 2.1. Introduction
    - 2.2. Control Structures: Selection
    - 2.3. Control Structures: Iteration
    - 2.4. Functions
3. Object-Based Computing
    - 3.1. Data Collections: Lists and Dictionaries
    - 3.2. Strings, Files, and the Web
    - 3.3. Graphics and Interfaces
4. Classes
5. Review

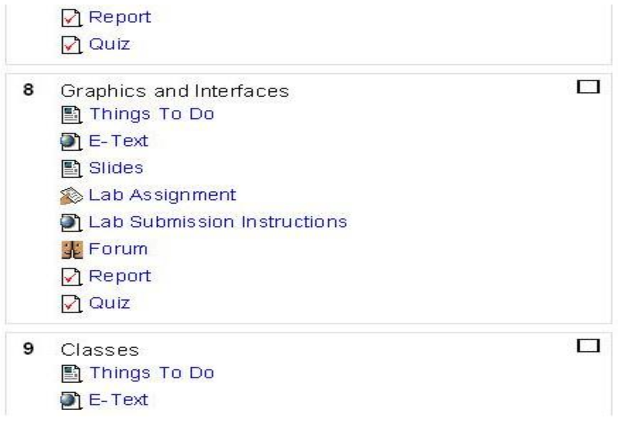
Whenever possible, concepts are introduced completely and at once. As an exception, some concepts are initially introduced in a limited fashion and then revisited in a later topic for full coverage. For example, string basics are presented in the *Introduction* then sting objects are fully covered in the topic on *Strings, Files, and the Web*. Likewise, simple "void" functions without parameters are presented in the *Introduction*, while full coverage of functions is offered later in a dedicated topic. Partial coverage of primary concepts is sometimes necessary to open early opportunities for existing lab experiments and exploration.

3.2 Online Study Pack

This "Python First" course of study is supported by a comprehensive online study pack [4]. The study pack offers a variety of custom-made learning modules, such as *e-texts, slides, lab assignments, forums,* and *quizzes*. We are the original authors of all learning modules.

The "Python First" study pack is based on Moodle, an increasingly popular free lightweight course management system. The home page of the study pack consists of a list of the principal course topics, together with links to available learning contents modules. The appearance of a single topic in the "Python First" home page is depicted in Fig. 4.

Each topic starts with an *e-text*, which is a chapter from a digital textbook. Although the *slides* follow the *e-text*, they are detailed and self-contained, and can be used by students as an alternative reading resource. One online *quiz* comes with each topic and is normally due a few days after the topic has been finished in lectures.



Report
Quiz

8 Graphics and Interfaces
Things To Do
E-Text
Slides
Lab Assignment
Lab Submission Instructions
Forum
Report
Quiz

9 Classes
Things To Do
E-Text

Figure 4. Learning modules in the "Python First" study pack

Each *lab assignment* incorporates *detailed self-guided labs* and also a collection of useful *sample programs*. Completed labs are submitted online when ready, which liberates students from the need to physically deliver lab work to the instructor at specific time and in a specific location. Furthermore, students file an online lab report which immediately gives them provisionary credit for the lab assignment. Online lab submissions are subject to an *audit* by the instructor.

Quizzes and *lab assignments* allow multiple online submissions. Students can use various forms of help and can submit work as many times as they want. Because a submission can always be corrected by yet another submission (prior to the programmed deadline), students are free to explore, experiment, and learn at their own pace. Students are well aware that *quizzes* and *lab assignments* are intended more as learning tools, rather than as principal evaluation tools, and the majority of students use them to learn and prepare for exams.

The type and amount of digital content that is included in the "Python First" digital pack [4] is outlined in Table 1.

Table 1. Digital contents in the "Python First" study pack

Learning Modules	Total Volume
E-texts (10)	66,000 words
Slides	730 slides
Lab assignments	36,000 words in 37 required self-guided labs and 25 optional labs
Sample programs	58 sample programs
Quizzes	280 questions

It is important to clarify that a *digital study pack* is not the same as an *online course*. At Chapman University in California for example, the "Python First" online pack is used to teach on-site courses, with lectures and supervised labs conducted two times a week. Most entry-level students benefit best from face-to-face onsite instruction with lectures and supervised labs. It is also possible to use the "Python First" online study pack for online courses. This pack can support remedial online courses or independent study online courses for mature and/or independent learners.

3.3 Self-Guided Labs

A typical *self-guided lab* starts with three sections: *Objectives*, *Background*, and *Lab Overview*. The *Objectives* section defines the lab goals. The *Background* section introduces concepts and techniques necessary for the particular lab but not covered in the *e-text*. The *Lab Overview* section outlines the steps that should be performed to complete the lab. The rest of the lab text contains detailed *step-by-step instructions* of how to complete the lab.

Students are free to choose one of three recommended approaches to a self-guided lab:

- Follow the detailed step-by-step lab instructions, or
- Try the lab independently, but also consult the instructions when needed, or
- Perform the lab independently, without looking at instructions at all.

Any of the above approaches can be beneficial for students, depending on their background, motivation, and physical location. For example, a student who chooses not to attend a certain supervised lab and prefers to work independently instead, can follow the detailed step-by-step lab instructions. A student who is gaining confidence may choose to try the lab independently, but also peek at the instructions when needed. A more advanced student may choose to perform the entire lab independently, without looking at instructions at all.

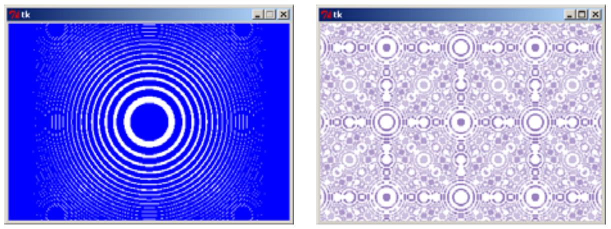


Figure 5. A sample fractal (left) and a student fractal (right)

Most labs require the exploration and then the enhancement of sample programs. For example, *Lab B* from the topic on *Graphics and Interfaces* is accompanied with a sample fractal program. The sample program draws the pattern shown in Fig. 5 (left). For this lab, students are asked to study the sample fractal program and then to modify it so that their new program produces a different and original pattern. As a final activity, students are expected to post their original pattern in a forum, together with a brief description of their enhanced program. This lab unleashes student creativity and triggers competition that results in some interesting designs, such as the one shown in Fig. 5 (right).

During the entire course of study, students are expected to complete *37 required self-guided labs*. Also, students may choose to work on *25 optional labs* for extra credit. Optional labs do not provide detailed step-by-step instructions and are usually more challenging than required labs. In a fourteen-week semester, required labs average 2.6 per week, while optional labs average 1.8 per week.

CONCLUSIONS

This paper introduces a new "Python First" lab-based digital introduction to computer science. Others have already successfully used Python in CS1 courses [7, 10]. The original contribution of our approach is the comprehensive online study pack which includes detailed self-guided lab assignments, among other useful digital resources [4]. With detailed self-guided labs, students are free to make various choices that best suit their background and goals. Our "Python First" course liberates students from the complexity of commercial languages and is expected to stimulate them to explore computing careers.

In the fall 2005 semester, we administered a voluntary anonymous survey of student perceptions of the "Python First" course. We adopted some excellent survey questions from [2].

Answers to the first survey question reveal that more than half of all students were true beginners (Table 2).

Table 2. Preliminary student programming background

What was your knowledge of programming (in any language, not necessarily Python) when you came to this course?	
Practically you were a beginner.	53.3%
You knew some programming but not much.	26.7%
Your programming knowledge was intermediate.	13.3%
You were good in creating programs.	6.7%
You were already an excellent programmer.	0%

Answers to the second question demonstrate that students prefer online resources to printed ones. Online sample programs, the online study pack in its entirety, and the online labs are identified as most beneficial (Table 3).

Table 3. Preferred learning resources

How have various resources helped you (would have helped you if available) in learning how to program? Select importance from 1 (least important) to 5 (most important).	
Sample programs (available online)	4.7
The entire online study pack	4.5
Online lab assignments	4.3
Online e-texts	3.7
Online slides	3.3
Printed (by you) lab assignments	3.1
Printed (by you) e-texts	2.5
Printed (by you) slides	2.5
Published paper lab manual (if it were available)	1.9
Published paper textbook (if it were available)	1.8

Answers to the third question reveal that students recognize the entire course of study to be more beneficial than any individual activity, and that lectures are considered least beneficial (Table 4).

Table 4. Preferred learning activities

How have various activities helped you learn in this course? Select importance from 1 (least important) to 5 (most important).	
The entire course of study	4.2
Scheduled lab sessions, working with someone	4.0
Taking quizzes online	3.8
Scheduled lab sessions, working alone	3.7
Studying with someone else	3.7
Working on programs outside of labs	3.7
Studying alone	3.5
Lectures	3.3

Answers to the fourth question show that student like and would recommend Python (Table 5).

Table 5. Python evaluation

Select answers to these questions from 1 (least important) to 5 (most important).	
Would you recommend Python as a first language to beginners in computing?	4.5
Would you recommend Python programming to others?	4.1
Would you recommend another language, not Python, as a first language to beginners?	2.4

Our "Python First" course has been so well received that it has been recently approved by Chapman University as a general education science elective, thus opening new recruitment opportunities for the computer science major. Since the introduction of Python in CS1, the attrition rate in our Java-based CS2 courses has not exceeded 4%.

4. REFERENCES

[1] Kelleher C., R. Pausch. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys*, Vol. 37, Issue 2 (June 2005), 83 - 137.

[2] Lahtinen E., Ala-Mutka K., Järvinen H-M. A study of the difficulties of novice programmers. *ITiCSE'05* (Caparica, Portugal), 14-18.

[3] Mahmoud, Q., W. Dobosiewicz, D. Swayne. Redesigning introductory computer programming with HTML, JavaScript, and Java. *SIGCSE'04* (Norfolk, Virginia), 120 - 124.

[4] Radenski, A. *Introduction to Computing with Python*. http://www.studypack.com.

[5] Radenski, A. *Object-Oriented Computing with Java*. http://www.studypack.com.

[6] Rossum G. van. *Computer programming for everybody*. http://www.python.org/doc/essays/cp4e.html

[7] Shannon C. Another breadth-first approach to CS 1 using Python. *SIGCSE'03* (Reno, Nevada), 248-251.

[8] *The TeachScheme! Project.* http://www.teach-scheme.org/.

[9] TIOBE Software. *TIOBE programming community index.* http://www.tiobe.com/tpci.htm

[10] Zelle J. Python as a first language. http://mcsp.wartburg.edu/zelle/python/python-first.html.