Python CS1 as Preparation for C++ CS2

Richard J. Enbody, William F. Punch, Mark McCullen
Department of Computer Science and Engineering
Mighigan State University
East Lansing, Michigan
[enbody,punch,mccullen]@cse.msu.edu

ABSTRACT

How suitable is a Python-based CS1 course as preparation for a C++-based CS2 course? After fifteen years of using C++ for both CS1 and CS2, the Computer Science Department at Michigan State University changed the CS1 course to Python. This paper examines the impact of that change on the second course in the sequence, CS2, which kept C++ as its primary language. We report results on a CS2 class which had a mixture of students who had used either C++ or Python from our CS1 course. The CS2 class covered the same topics as previously, though with some changes, and even gave the same final exam as a previous offering. Independent samples t-tests were used to compare students from the Python group with students from the non-Python group on three outcomes: final exam grade, programming projects scores, and final grade for the course. The main result was that there were no significant differences between the groups for all three outcomes. In addition, multiple regression analysis showed that students' past performance (overall GPA) in the University predicted final grades, final exam scores, and programming project scores for the course, but there was no effect of the programming language feature: Python or non-Python. We feel this shows that the Python-based CS1 course prepared students for the C++-based CS2 course as well as the C++-based CS1 course did—while exposing them to a different, powerful and useful language.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education—*computer science education, curriculum*

General Terms

Measurement, Experimentation

Keywords

CS1, CS2, Python, curriculum, intro. to programming

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGCSE'09, March 3–7, 2009, Chattanooga, Tennessee, USA.
Copyright 2009 ACM 978-1-60558-183-5/09/03 ...$5.00.

1. INTRODUCTION

The impact of changes in a Computer Science I (CS1) course on the subsequent Computer Science II (CS2) course is an important curricular question. The goals of a CS1 introduction to programming course are often broader than simply starting computer science majors. As a result, the CS1 goals in some institutions are at a variance with the goals in the subsequent CS2 course which is usually highly targeted toward CS majors.

We had a unique opportunity to study a CS2 class after changing the programming language in the CS1 class to Python. After the first offering of this revised CS1 class, the following semester's CS2 class had a number of students from both the Python CS1 and C++ CS1 class. Two groups of students in the same class provided an opportunity for comparison.

An important aspect of this situation is that the outcomes and topics for the CS2 course were unchanged. In fact, at the end of the semester the CS2 instructor decided to use the same exam he used one year earlier, before the CS1 format shift. Thus were we able to both compare performance of groups within a class, and compare the overall class performance with a different class.

This paper is organized as follows. We begin with a look into why we changed the CS1 course—in particular, the language change. Next, we look at the CS2 course. Then we present the data we collected, analyze the data, and draw conclusions.

2. CS1 COURSE CHANGES: PYTHON

Why change the CS1 course? A CS1 course is a first course in computer science, and usually emphasises an introduction to programming. It is also a course on problem solving and applying a programming language to solve a problem. As a result, the choice of programming language can have a significant impact on the implementation of the course (see [7] for an excellent survey). A recent survey of the top thirty Ph.D. CS degree granting programs showed a distinct preference for Java [2]. For fifteen years C++ has been the language for our CS1-CS2 sequence—a long time in the computer science world.

As in some other institutions, non-CS majors have found our CS1 course to be useful. We find that now the majority of students in the course are non-CS majors who are not required to take the course. STEM students (Science, Technology, Engineering, Mathematics) are naturally drawn to the course, but we have found students from all majors in our CS1 course. As the impact of computing has grown across

all fields there has been an increasing need for students in all majors to develop some programming skills. In particular, a computing course which develops students into useful programmers after one semester is needed. We found that C++ did not adequately satisfy that need within one semester, and we were not convinced that its sibling languages, Java and C#, satisfied that need signifcantly better.

Languages such as Alice [8] and Scratch [6] have proven to be attractive introductions to computing, especially for non-majors. Media computation [3] has also been effective. Non-language approaches such as the Principles of Computation [1] have also proven to be effective. However, many such approaches are for "CS0" courses. Such courses are valuable, but we are working with a course which must prepare students for CS2, and it has not yet been demonstrated that those approaches satisfy that goal.

Python features a mixture of readability and practicality—nice features for an introductory language. It is also an interpreted language that encourages experimentation—a great learning aid. It has a number of immediately available data structures (strings, lists, dictionaries a.k.a. associative arrays, and sets) with associated functions and methods to easily manipulate those structures. It is object-oriented which helps in preparation for both solving complex problems and other languages. Python interacts well with other languages which allows one to apply high-level constructs to them. It is a free language that runs under most environments including, but not limited to, Microsoft Windows, Mac OS-X, and Linux. It includes many modern programming language features together with a seemingly limitless set of modules that extend it. Those modules come from the large and supportive Python community that has been rapidly growing.

In short, Python can be described as a "best-practices" language, providing practical tools to do a job with a minimum of effort.

Taken together these features allow a novice to focus more on problem solving and less on language issues. In addition, the built-in language features make data manipulation particularly easy allowing students to more easily work on real data. As a result, not only do students solve more challenging problems, but they also have a tool that can be used in subsequent courses, research or even personal use.

Ruby is another high-level language whose use has been increasing and could be considered for this role. One of the reasons we chose Python was the large community that has developed around it, particularly outside of computer science—that is, where Python is applied. Also, we believed that Python would be particularly appropriate in preparing students for CS2.

We are not the only institution to make the change to Python for CS1. The survey of top-30 CS Ph.D. granting programs found that MIT has started teaching their C1, a new course designed to teach freshmen introductory electrical engineering and computer science, with Python [2]. This movement is gaining momentum. A recent IEEE Computer article "In Praise of Scripting: Real Programming Pragmatism" recommends "scripting, not Java, be taught first, asserting that students should learn to love their own possibilities before they learn to loathe other people's restrictions" and concludes with the observation that "an emerging consensus in the scripting community holds that Python is the right solution for freshman programming" [5].

Our CS1 course is a fairly standard course covering problem solving, data structures, and basic programming concepts. Since there can be some variance in offerings Table 1 shows the list of topics we cover. However, that topics list does not accurately reflect the problem-solving aspect of the course. The course has a weekly closed lab where students, with the aid of a teaching assistant, work exercises. Of greater importance is the weekly programming assignment—there are eleven such projects throughout the semester. A list of programming assignments from Spring 2008 is provided in Table 2. We have observed that students were struggling more with the problem than with the implementation in Python. That is, once they understood the problem, the solution came relatively easily—exactly what one wants.

```
A. Overview:  Digital Computer Systems
B. Introduction to Problem Solving
  Engage, Visualize, Experiment, Simplify, Analyze
C. Python beginnings
  Naming, Keywords, Operators, Punctuators
  Namespaces, Types, Assignment
D. Strings and Booleans
E. Control:  Selection
  if, if-else, if-elif-else
F. Control:  Repetition
  for, while, continue, break, else
G. Data Structures I: Lists and Tuples
  Lists (mutable), methods, functions, slicing
  Tuples (immutable)
H. Functions
I. Files
J. Data Structures II: Sets and Dictionaries
  methods, functions, iterators
K. Scope
  Namespaces:  qualified & unqualfied
L. Classes:  user-defined data type
  Methods, Attributes, 'dot'-reference, self
  Constructor, Overloading
M. Exceptions
N. Sorting, Searching, BigO
  Sorting:  selection
  Search:  linear, binary
O. Testing
  Doctest, Unittest
```

Table 1: CS1 Python course topics (one semester).

Preliminary indications from three offerings is that Python is an improvement for our CS1 students—we are working on data to better establish those indications. However, the kinds of problems the students can address has changed. Students have been able to download real data from the Web and analyse it (fitting the ideas of [4]). Examples include building a simple classifier for breast-cancer data, finding cycles in sunspot data, and simulating DNA transcription. The idea is to grab some real data, parse it into useful form, and analyze it—ideal problem solving skills for scientists. Text analysis such as building a concordance or tag-cloud is quite simple in Python, and provides a tool that humanities students can use in their courses.

We expected Python to work well with CS1, but what would the impact be on our CS2 course, especially since that course uses C++ on a Linux platform?

Project	Title
1	"Hello World"
2	"Einstein Game" number puzzle
3	"Egyptian Multiplication" algorithm
4	"Mastermind" game
5	Given word: create anagram
6	Using online data: determine greatest basketball player
7	Using online actor data: find common actors given movie titles
8	Using online breast cancer data: write a classifier
9	"Aces Up" solitaire game
10	DNA transcription
11	create "Carpet Fishing" game from Dilbert cartoon

Table 2: CS1 Python weekly programming assignments (SS08).

3. CS2 COURSE

Since C++ remains the foundation of our curriculum, changing the outcomes of the CS2 course was not an option. We needed students to exit the CS2 course as good problem solvers able to program comfortably in C++. Of course, data structures and object-oriented programming are important topics in any CS2 course.

The CS2 course always devoted some time to reviewing C++ because some students would not have taken the C++-based CS1 course immediately before taking the CS2 course. In addition, some students may have transfered credits from a CS1 course based on a different language—almost always Java. With the change to Python in CS1, the CS2 course made the following adjustments (+ indicates increased coverage; - indicates decreased covereage). Note that these adjustments are the degree of coverage—not additions or deletions.

+ : C++ expressions and control structures

+ : C++ functions and libraries

- : C++ streams and strings

- : Generic functions and classes

- : Algorithm efficiency

- : Recursion

Since CS2 courses can vary signifcantly it is worth listing the topics covered: see Table 3. There is heavy emphasis on classes as abstract data types (ADT), as well as looking at standard data structures as ADTs. As with the CS1 course there are weekly programming projects so students get to solve large problems using the techniques developed in class. Also, there are weekly closed labs.

```
A. Introduction to Linux and C++
B. C++ expressions and control structures
C. C++ functions and libraries
D. Data abstraction & fundamental data types
E. Arrays, records and pointers
F. Algorithm efficiency
G. Data abstraction and classes
H. ADT Stream and ADT String
I. ADT Sequence
J. ADT Stack
K. ADT Queue
L. Generic functions and classes
M. The C++ standard template library
N. Recursion
O. ADT Binary Search Tree
P. ADT Table
```

Table 3: CS2 topics

4. DATA

The spring semester 2008 offering of our CS2 course provided a unique opportunity to study the impact of switching CS1 to Python because experience indicated that a reasonable percentage of the class would have taken the older CS1 course (C++). The first Python CS1 offerring was in the preceeding semester (fall 2007), but not all students take CS2 directly after CS1. That meant that we would have both Python-based and C++-based prepared students in one class.

In an ideal world we would have offered separate CS1 courses: one using Python, the other using C++. In that case, we could have compared within the CS1 course, and had a perfect case study for the following CS2 course. However, there are complications in using students for testing in that way.

We applied for and received approval to study students in the CS2 course from our Institutional Review Board (IRB). An important aspect of such a study is to get approval from each student for participation. Of 109 students enrolled in the CS2 course we received approval from 83. No incentive or extra credit was offered to participants. Approval was sought at the end of the semester which may or may not have had an impact on who agreed to participate. Of those 83 participants there were 62 students who had taken our Python-based CS1 course in the previous semester and 21 students who had taken some other CS1 course. That is almost an exact 3:1 ratio. Those "other" students were primarily from our C++-based CS1 course, but there were a few who had transferred from other institutions with C++ or Java experience. The language for transfer students was self-reported data.

With a 3:1 ratio between our two groups and 21 students in the smaller group we decided that we had sufficient students to get significant results.

4.1 Data Collection

We had two different sets of data for our study.

- CS2 class data from SS08:
  - Background: ACT scores, GPA, CS1 language
  - SS08 CS2 data: exam scores, programming project scores, final grade
- The same CS2 final exam was given in SS07 and SS08.

Giving the same CS2 final exam from the previous year was not part of the original plan, but occurred to us at exam preparation time. Since it happened that no students from the previous year had picked up their final exam, it seemed reasonable to give the same exam. Since this data was not part of our IRB application only cumulative class statistics can be used.

On the last day of class, the authors described the study to the students and sought written permission for participation. No extra credit or other incentives were offered. All except one student present agreed to participate. In addition to allowing us to collect and use background and class data, students were asked to fill out a brief survey. Results of that survey have not been analyzed.

4.2 Data Analysis

The question to be answered is

> *How well does a Python-based CS1 course prepare students for CS2?*

We address that question in two ways:

- Is there a difference in the CS2 performance of Python-prepared students?
  - Is there a difference *within* the CS2 class?
  - Is there a difference *between* CS2 classes?
- Is Python-preparation a predictor of CS2 performance?

4.2.1 Difference between CS2 classes

The same CS2 final exam was given in SS07 and SS08. One question was changed because it was an inappropriate question for second year's class since that material was abbreviated (on random-access vs. bi-directional iterators). That question accounted for only 2% of the final exam score.

Table 4 shows the statistics for the same final exam given in two different semesters. The SS07 class was before the Python-based CS1 course existed so there were no Python-prepared students. The SS08 class had approximately three-fourths of the students from the Python-based CS1 class. However, the exam statistics are not statistically different as shown in Table 4 ($t = -.10$, $p > 0.05$).

We know that the material in the two CS2 offerrings did not differ significantly so the final exam means indicate that the performance of the students did not differ significantly. This is one indication that Python preparation had no negative impact on the subsequent CS2 course.

4.2.2 Difference within a CS2 class

The SS08 CS2 class had both Python-prepared and students with other preparation.

Yr	n	Mean	Med'n	StdD	t-val	p
S07	109	174	177	27	-0.10	$p > 0.05$
S08	83	177	179	26		

Table 4: CS2 final exam scores in SS07 and SS08

Group	n	Mean	StdD	t-value	p
Python	62	172	40	-0.51	$p = 0.61$
not-Python	21	177	24		

Table 5: CS2 final exams in SS08

Group	n	Mean	StdD	t-value	p
Python	62	324	82	-0.85	$p = 0.40$
not-Python	21	340	56		

Table 6: CS2 programming projects in SS08

> *If we group the class into Python-prepared and not-Python-prepared, is there a difference in the performance of the two groups?*

First let's examine two outcomes in the class. Here we look at the performance of the Python and not-Python groups in the final exam and the programming projects (homework). There was no significant difference between the two groups in either outcome. The t-values for each of the analyses were: final exam ($t = -.51$, $p = .61$), and project grade ($t = -.85$, $p = .40$) as shown in Table 5 and Table 6.

That is, the Python-prepared students performance in the CS2 class was not statistically different than the non-Python-prepared students.

4.2.3 Predictors of performance

Another question to ask is:

> *Is the group (Python or not-Python) a predictor of performance in the CS2 class?*

In this case we considered three outcomes: final exam, programming projects, and final grade. We used multiple regression analysis to determine if there were differences between the two groups on the three outcomes when characteristics of the students at the time of enrollment were controlled. In addition to group, the predictor variables considered in these analyses were overall student GPA and ACT math scores. *Once again classroom group was not a significant predictor of any of the three outcomes when overall GPA and ACT scores were controlled.* Only overall GPA was a significant predictor of grades for the class and for project grades. For final exam grades, both overall GPA and ACT math scores were significant predictors of the outcome and explained 42% of the variance in final exam scores. For this analysis, the standardized regression coefficients for the three predictor variables were: overall GPA (.59), ACT math (.20), and group (.04). Students' past performance

Predictors	StdCoeff	t	Sig
GPA	0.59	6.25	0.00
ACT math	0.20	2.10	0.04
Python/not-Python	0.04	0.38	0.70

Table 7: CS2 predictors in SS08

in the University and math aptitude predicted final exam scores for the course, but there was no effect of classroom group. Table 7 summarizes the supporting statistical data.

This evidence indicates that whether a student took CS1 in Python or not is not a predictor of performance in the CS2 course.

5. LIMITATIONS

What other factors may have impacted the results? Some non-Python students in the CS2 course took their CS1 course elsewhere either as a high-school Java AP course or as a transfer. Others took their CS1 course in an earlier semester at our school. The latter students had a larger gap between their CS1 and CS2 courses that may have impacted the study. Some of the former students may have also had a gap. We have not been able to control for that difference, but do observe that similar differences exist in any semester so our comparison between semesters provides a partial answer to that question—no difference was found. Another possible factor is that the two groups differ in their abilities. We did not control for that potential factor. Another factor is the difference among CS1 instructors for all students (but we note that the CS2 instructor was the same for both semesters).

6. CONCLUSIONS

The choice of language in a CS1 course can have an impact on a number of difficult-to-measure factors within a CS1 course including problem-solving abilities and satisfaction. A critical question is the impact of CS1 language choice on a subsequent CS2 course. In this work we examined that impact. When final exams were compared for a CS2 course offered before the Python transition with a course after the transition no statistical difference was found. The first CS2 course after the transition contained a mixture of Python-prepared and non-Python prepared students. There was no statistical difference in the performance between those two student groups. Finally, regression analysis examined if Python vs. non-Python preparation was a predictor of CS2 performance and it was not.

While we would have liked to find that Python-prepared students performed better in the CS2 course, we did find that they performed as well as students prepared with other languages—in particular, the same language (C++) that was used in the CS2 course. That is, those students with a whole semester of C++ CS1 did no better in a C++ CS2 course than students who had to transition from Python. In addition, the Python students had the advantage of knowing another language that is powerful and easy to use.

We conclude that Python-based CS1 is a viable alternative for CS1 even for programs whose subsequent coursework is basesd on a different language (C++ in this case).

7. ACKNOWLEDGMENTS

Our colleague Tom Luster provided invaluable help with SPSS and analysis.

8. REFERENCES

[1] T. J. Cortina. An introduction to computer science for non-majors using principles of computation. In *SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education*, pages 218–222, New York, NY, USA, 2007. ACM.

[2] J. Forbes and D. D. Garcia. "...but what do the top-rated schools do?": a survey of introductory computer science curricula. *SIGCSE Bull.*, 39(1):245–246, 2007.

[3] M. Guzdial. *Introduction to Computing and Programming in Python: A Multimedia Approach*. Pearson Prentice Hall, New Jersey, 2005.

[4] N. Jukic and P. Gray. Using real data to invigorate student learning. *SIGCSE Bull.*, 40(2):6–10, 2008.

[5] R. P. Loui. In praise of scripting: Real programming pragmatism. *IEEE Computer*, 41(7):22–26, July 2008.

[6] D. J. Malan and H. H. Leitner. Scratch for budding computer scientists. In *SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education*, pages 223–227, New York, NY, USA, 2007. ACM.

[7] A. Pears, S. Seidman, L. Malmi, L. Mannila, E. Adams, J. Bennedsen, M. Devlin, and J. Paterson. A survey of literature on the teaching of introductory programming. In *ITiCSE-WGR '07: Working group reports on ITiCSE on Innovation and technology in computer science education*, pages 204–223, New York, NY, USA, 2007. ACM.

[8] K. Powers, S. Ecott, and L. M. Hirshfield. Through the looking glass: teaching cs0 with alice. In *SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education*, pages 213–217, New York, NY, USA, 2007. ACM.