Exploration on Cultivating Students' Abilities Based on Python Teaching Practice

Xinxiu Wen
School of Information Science and Engineering
East China University of Science and Technology
Shanghai, China
wenxinxiu@ecust.edu.cn

Zeping Yang, Zhanquan Wang, Min Zhao
School of Information Science and Engineering
East China University of Science and Technology
Shanghai, China
{yangzeping, zhqwang, zhaomin}@ecust.edu.cn

Abstract—**Computer programming courses (C, Java, C++, etc) intended for non-majors are facing a lot of problems in cultivating undergraduates' abilities of computational thinking and problem solving. Students are frustrated with complex syntax of these languages and are confused with their practical applications. In the paper, our exploration on students' abilities cultivations based on Python teaching practice is introduced. Lesson plan, teaching method and teaching effect are presented in detail. We wish our research and practice could give some hint for the reform of introductory programming courses.**

Keywords-problem solving; Python; task-driven; teaching method; computational thinking

I. INTRODUCTION

Due to the critical conflict between limited class hours and complex languages syntax, computer programming courses (C, Java, C++, etc) intended for non-majors are facing a lot of problems in most universities all over the world [1, 2]. In East China University of Science and Technology (ECUST), most of science and engineering students take C programming as their first programming course, which the total class hours are 48 class hours (32 class hours for theoretical study and 16 class hours for experimental exercise). Having worked on teaching C programming for ten years, our research group found the following reasons hinder the improvement of computer abilities of non-computer major students:

- C language has rich syntax details. Students have to spend much time on syntax learning and error correcting, which leads to the absence of confidence.
- Class hour is relatively limited. Teachers are busy finishing the appointed teaching task, which lead to the ignoring of application ability training.
- Thinking mode's cultivation is scarce. Students couldn't understand the meaning and function of this course.

In 2006, Jeannette M. Wing [3] gave the definition of computational thinking (CT). She said that computational thinking was a fundamental skill for everyone, not just for computer scientists. Just like reading, writing, and arithmetic, computational thinking should be added to every child's analytical ability. Computational thinking has attracted much interest in education field since it was proposed. Many researchers think that computational thinking should be combined into curriculum to form a solid foundation for further skill development [4, 5].

In 1989, Guido von Rossum [6] created Python language. It has been proved an easy language for beginners and used in the industry such as Google, Philips, etc [7]. In short, Python has the following characteristics:

- It is simple and flexible;
- It is free and powerful;
- It is interpreted and object-oriented.

Because it could help students concentrate on the problem itself rather than syntactic details, we think Python as a teaching language is more suitable than C to cultivate students' abilities of computational thinking and problem solving.

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 introduces our lesson plan. Section 4 describes in detail our teaching method and teaching effect. Conclusions are presented in Section 5.

II. RELATED WORK

Joseph D. Oldham [8] introduced the advantages of Python and the disadvantages of Python in teaching basic programming concepts and constructs. Their students report showed that Python is easy for a beginning student to practice on interesting projects. Arnd Backer [9] represented their experience on course construction of computational physics by using Python as the programming language. The course's main goal was to enable students to solve problems in physics with the help of numerical computations. The result illustrated that their preliminary effort which adopted Python as programming language to teach computational physics was very successful. Linda Grandell et al [10] presented a case analysis of using Python to teach programming in high school students, which emphasized programming rather than Python. The initial research indicated that Python could be a suitable language for novices. Nikolaos Avouris et al [11] reported their experience of introducing collaborative and project based methods in the first year course using Python as a programming language, and discussed the effects of these methods to students' attitude and performance.

III. LESSON PLAN

Python programming is an elective computer course for all undergraduate students in ECUST. The total class hours are 40, which include 24 theoretical class hours and 16 experimental class hours.

A. The objectives of teaching

The basic tasks of course are to make students master the basic theory knowledge of Python language and the basic thought of program design, and the ultimate objectives of course are to teach and improve the students' abilities of analyzing problems, solving problems and computational thinking.

B. The distribution of total weeks

The course contents and teaching plan are as follows (T represents theoretical class hours and E represents experimental class hours). Computational thinking characteristics are permeated into the case analysis of each chapter:

1) Introduction to Python (T(2) +E (1))
   1.1 Computational thinking and its characteristics
   1.2 Evolution of Python
   1.3 Characteristics of Python
   1.4 Case analysis 1
2) Basic data types (T(2) +E (1))
   2.1 Value and type
   2.2 Variable assignment
   2.3 Type conversion
   2.4 Case analysis 2
3) String, list, tuple and dictionary (T(4) +E (2))
   3.1 String
   3.2 List
   3.3Tuple
   3.4 Dictionary
   3.5 Case analysis 3
4) Condition and loop(T(4) +E (2))
   4.1 If statement
   4.2 While statement
   4.3 For statement
   4.4 Break and continue statement
   4.5 Case analysis 4
5) File reading and writing(T(2) +E (2))
   5.1 File reading
   5.2 File writing
   5.3 Batch reading and writing
   5.4 Case analysis 5
6) Function and parameter(T(2) +E (2))
   6.1 Function definition
   6.2 Argument passing
   6.3 Recursive function
   6.4 Case analysis 6
7) Object-oriented design(T(2) +E (2))
   7.1 Class definition
   7.2 Class inheritance
   7.3 Case analysis 7
8) WxPython GUI design(T(2) +E (2))
   8.1 Introduction to WxPython
   8.2 Form design
   8.3 Control design
   8.4 Event response
   8.5 Case analysis 8
9) Network programming(T(2) +E (1))
   9.1 Introduction to network communication
   9.2 Client design
   9.3 Server design
   9.4 Case analysis 9
10) Database Programming (T(2) +E (1))
   10.1 Introduction to database
   10.2 Database creation
   10.3 Database connection
   10.4 Case analysis 10

C. Examination grading

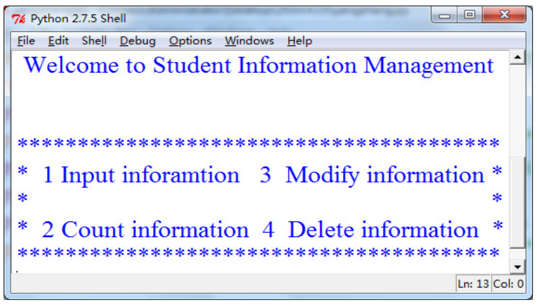
A final examination would be arranged in the end of semester. The examination is in computer test form and realizes separation of teaching and examination. The examination would be scored by centesimal system. Final grade is composed of examination (60%) + experiment (10%) + attendance (10%) + homework (20%).

IV. TEACHING METHOD AND EFFECT

Python programming is a practical course which needs students spend much time on exercise. Computational thinking has such characteristics as abstraction, automation, transformation, type-checking, error-correction, prevention, etc [3]. Having studied CT for years, we think that ability cultivation of computational thinking should be integrated into curriculum teaching closely, rather than be taught boringly.

A. The task-driven teaching method

In our school, students who select Python programming course are very interested with program design and are eager for problem solving. Therefore, we choose the computer lab as our classroom, which facilitates students' learning and programming. Having compared different ways of teaching, task-driven teaching method is adopted to improve students' interest and confidence. At the beginning of semester, we illustrate several interesting or functional teaching examples to students and list the tasks that they have to finish after each class. For example, the score management system is related to students and is easy for them to understand system demands. The whole system is showed in the first class and the system function is divided into different modules, which illustrates the abstraction and decomposition characteristics of computational thinking. Having finished the chapter 1, students are asked to print the graph as figure1 to improve their abilities of simulation. Having studied chapter 2 and chapter 3, they should have the abilities to store students information (code, name, math score, computer score, etc) into lists and calculate total (average, max, etc) score of each course. During the experiment of chapter 2 and chapter 3, they learn the characteristics of CT such as type checking and transformation. Chapter 4 and chapter 5 ask them to handle lots of student information, save these information into file (notepad, excel, etc) and take information from these files, which help them master the concept of error correcting and deadlock. Chapter 6 and chapter 7 require students to use functions or object-oriented design method to improve system's modularity and reusability. Kinds of search algorithms and sort algorithms could help students understand recursion and compromise characteristics of CT. Chapter 8 demands students using WxPython to build user interface and handle event response (Figure 2). Chapter 9 and Chapter 10 claim students to show data on web page and store data in the Access database, which involve characteristics of prevention, protection and recovery. Students learn computational thinking from the example of score management system during the semester, which is practical, acceptable and interesting.



```
 Welcome to Student Information Management


*****************************************
*  1 Input inforamtion   3  Modify information *
*                                       *
*  2 Count information  4  Delete information  *
*****************************************
```

Figure 1. The print interface example

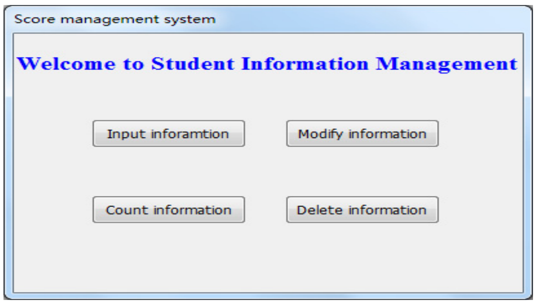
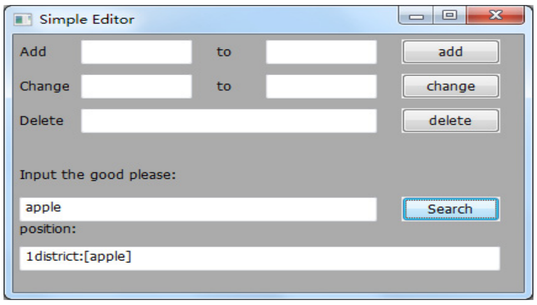


Figure 2. The WxPython example

B. *The evaluation of teaching effect*

Improving students' abilities of problem analyzing and solving are very important when learning a programming language. In addition to emphasizing the importance of computational thinking in the first class, we ask students to design and realize a small project with windows combined with their interest or majors in every semester. When students first receive the task at the beginning phase of semester, most of students think that it is impossible for them finish the work. However, we insist on saving five minutes in each class and talking about what questions could be solved depended on studied knowledge. By this way, students' programming abilities of solving small questions are improved rapidly. At the same time, their self-confidence is increasing gradually.

Take Python programming course in the 2013-2014 academic year in ECUST as an example, all students finished their tasks in the end of semester, although there were some difference among systems' complexity and robustness. Every student was asked to give a short presentation on his (her) project and share his (her) ideas with classmates, which improved students' expressive abilities and thinking abilities. The projects that students developed include tetris, supermarket management systems (Figure3), twenty-four point (Figure 4), personal count management system, etc.



apple

1district:[apple]

Figure 3. The supermarket management system

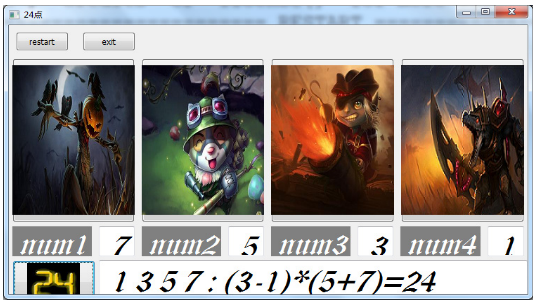


Figure 4. The twenty-four point

V. CONCLUSIONS

Fostering interdisciplinary talents with abilities of computational thinking and professional problem solving are essential requirements for computer fundamental education in most universities. Having taught C programming for 10 years in computer center of ECUST and instructed Python programming for two semesters, we conclude that selecting a suitable programming language could save much time and pay more attention on core problems, which is very important for thinking mode cultivation and application ability training. The experimental results illustrate that Python is a suitable tool to achieve our teaching objectives. Future research aims to explore Python applications in various majors and help students solve professional problems in work and study.

ACKNOWLEDGMENT

The authors would like to express their appreciation to the anonymous referees for their valuable suggestions and comments. This work was partially supported by the Shanghai Undergraduate Teaching Reform Project at 2013 under grant No. YH0126121.

REFERENCES

[1] J. Holvitie, “Breaking the Programming Language Barrier: Using Program Visualizations to Transfer Programming Knowledge in One Programming Language to Another,” 12th IEEE International Conference on Advanced Learning Technologies, 2012, pp.116 – 120.

[2] D. Krpan, I. Bilobrk, “Introductory Programming Languages in Higher Education,” MIPRO 2011, May 23-27, Opatija, Croatia, 2011, pp. 1331-1336.

[3] J. M. Wing, “Computational Thinking,” Communication of the ACM, 2006, 49(3): 33-35.

[4] J. A. Qualls and L. B. Sherrell. “Why computational thinking should be integrated into the curriculum,” Journal of Computing Sciences in Colleges, Volume 25 Issue 5, May 2010, pp. 66-71.

[5] J. Lu, G. Fletcher, “Thinking about computational thinking,” 40th SIGCSE Technical Symposium on Computer Science Education, 2009, 260-264.

[6] G. V. Rossum, “Python Tutorial,” May 1995.

[7] H. Fangohr, “A comparison of C, MATLAB, and Python as teaching languages in engineering, ” in Computational Science ICCS 2004, ed, 2004, pp. 1210-1217.

[8] J. D. Oldham, “What happens after Python in CS1?,” Consortium for Computing Sciences in Colleges, Centre College. Danville, KY. 2005.

[9] A.Backer. “Computational Physics Education with Python,” Computing in Science & Technology. 2007, 9(3): 30-33.

[10] L. Grandell et al, “Why Complicate Things? Introducing Programming in High School Using Python,” Eighth Australasian Computing Education Conference (ACE2006). 2006.

[11] N. Avouris et al, “Teaching Introduction to Computing through a project-based collaborative learning approach,” 14th Panhellenic Conference on Informatics, 2010.