ATLANTIS PRESS

Advances in Social Science, Education and Humanities Research (ASSEHR), volume 156

2nd International Seminar on Education Innovation and Economic Management (SEIEM 2017)

Programming Language Teaching Model Based on Computational Thinking and Problem-based Learning

Guang-ming Chen

School of computer

JiaYing University

Meizhou, Guangdong, China

***Abstract*—The paper presents the purpose of learning programming language is to master the method of learning the language, rather than just master the language itself. Therefore, the cultivation of Computational Thinking is an important goal of language learning. After analyzing the teaching of Programming Languages in university, this paper put forward the Problem-Based Learning is beneficial to achieve this aim. During teaching process, Collaborative Learning and Grouping Learning can improve teaching effectiveness. A new method, which is to carry out these ideas and organize Problem-Based Learning and Grouping Learning, is given after reconstructing the teaching contents. The result of reform reveals that students' test scores have been greatly improved, the quality and quantity of the completed design work are greatly improved, the number of programming competition awards also increased significantly. In order to make computing thinking deeply rooted, this method may be extended to other important computer basic courses.**

Keywords—Programming Language; Teaching mode; Computational Thinking; Problem-based Learning; Collaborative Learning; Group Learning

I. INTRODUCTION

A. Programming Languages

How to learn Computer language learning is a big topic in universities and colleges. Although there are hundreds of Programming Languages, but only a limited number of them will be taught in colleges and universities. C is the most used language and the basis for the more modern C++, C#, and Objective-C. Java is used in countless types of programs, from games to web applications to ATM software. In many school Java is the first choice for learning coding. Python has a design philosophy that emphasizes code readability and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale. HTML, an essential starting place for any web developer. Having a handle on HTML is vital before moving on to any other sort of web development. [1]

Broadly speaking, even though we will not use them to code directly, *Formal Languages* can also be viewed as *Programming Languages*, for these languages reveal the essence of programming and we will have a better understanding of the nature of the computer by studying them.

B. The methods of Learning Programming Languages

In theory, we can learn *Formal Language* first. Different from ordinary *Programming Language*, *Formal Language* has completely precise grammar rules and semantic rules. We believe that students who understand *Formal language* will learn any *Programming Language* very easy. But it is unrealistic, because learning *Formal Languages* is far less interesting than the specific *Programming Language*, let alone the background knowledge students must learn. In addition, not all students can understand these theories in depth.

In contrast, another choice tends to pragmatism. Some people think that since the *Formal Languages* cannot be used in coding directly, why should we introduce them when we teach *Programming Language*? It is better to make student have a certain basis before learning *Programming Language*.

Obviously, the latter prevailed now, but such a teaching method is really suitable for students of computer majors?

C. The goals of Learning Programming Languages

Obviously, what kind of goal determines what kind of action. So the first thing we have to know is the aim of learning *Programming Languages*. Learners do not just think of computer language as a tool, but rather to understand how the machine works and how to design the language. As a professional, they may become the inventor of a language and not just the user. At least, they should also have the ability to learn new languages easily.

We should tell students some of basic theories when they contact with these languages first time, rather than until studying the course of *Formal Language*. The students of computer major could have asked In-depth questions and if teachers cannot give them help, their enthusiasm for learning will be hit. At the same time, it does not help to build a solid professional foundation for students if they cannot understand their language at hand at a higher level.

Obviously, we cannot study all these languages in four years and it seems we needn't to do so, because we can let

This research was financially supported by JiaYing University Teaching Reform Project JYJG20170213 and by Higher education innovation strong school project 315B0128.

students to know how to learn these languages after explaining some ideas about profound computer theory. That is all.

D. Development of Teaching Programming languages

In this era of rapid development of *Artificial Intelligence* technology, we must also think about how to improve the learning ability of computer language. Since *Computational Thinking* (CT) has been raised, it is seen as the most important ability of the student. With the convenience of information acquisition, *Problem-Based Learning* (PBL) has a significant effect on improving learning efficiency. At the same time, *Collective Learning* (CL) and *Grouping learning* (GL) are also crucial to cultivating of students' ability to cooperate each other. Based on these ideas, we will discuss how to build a new learning mechanism to meet the requirements of the Intellectual Age.

II. THE CORES OF PROGRAMMING LANGUAGE

What is the most important thing in the *Programming Language*? The solution of this question will help us to recognize the goals and help us build a learning process that fits this goal. We believe three things we must discuss if we pay some attention to the nature of the language.

One is the rules of language, including grammar and semantics. The second is that how does a computer run a program. The third is the algorithm to solve specific problem.

It should be said that these questions cannot be easily answered in just one course. Each of them involves a learning direction that leads to a series of related courses. But from this view, we can more clearly understand the method of language teaching.

A. The rules of language

a) Syntax: First, in C, the *Syntax Rules* for *Constants*, *Variable*s, and *Function Calls* are very clearly defined. We can define *Expression* and *Statement* based on them and at last we can define all parts of *Program*.

b) Semantics: Semantics Rules are based on the three basic structures of *Order*, *Branch* and *Iteration*. The process of *Function Calls* is the core of *Operational Semantics* which can be expressed by very accurate rules.

B. Program running

Obviously, different computers can run the same *Statement* in different ways. During language learning, we must talk about this issue in a context where a lot of details have been shielded. Even in the language which contains *Pointer Operations*, such C, we can simplify a number of concrete implementation and explain the *Execution* of a *Statement* on a "*Virtual Machine*".

From this point, we can study the details of this *"Virtual Machine"*, and then gradually understand the specific *Machine Instructions*. So that we can enter a *Hardware* world naturally and we can understand the basic *Composition of Computer*. While we do not study these issues further, we can still realize the fundamentals of *Hardware* implementation and the *Hierarchical Protocols* in the computer architecture.

C. Algorithm

Seriously, the study of the *Algorithm* is not a task that must be done during language learning. But we cannot understand language without any algorithm. Leaving algorithm, the *Programming Language* is meaningless. However, when we realize that we can put aside the actual language knowledge and focus only on giving an abstract answer to a practical question, we will find what we have to do is to learn that how to translate an abstract algorithm which be expressed by mathematical models into an implementable algorithm which be expressed by the *Procedure*s of the language.

The three aspects above are the basic elements of language learning. At first glance, these are commonplace, but it is surprising that they did not attract enough attention.

During language learning, students seem to be more willing to take an intuitive way to understand the grammatical phenomenon. One of the problems of the way is that they cannot have a real understanding of the delicate language structure, which becomes an obstacle to studying languages. When reading complex codes, students will have a difficult time, for they cannot understand the true meaning of the *Statement*, let alone write their own codes. Another problem is that when they learn a new language, it is not possible to build a framework based on these concepts.

Of course it is a very difficult task to make students master these points, because students lack of basic knowledge and teachers lack of correct method. But we still try to take some measures to solve this question. In the following, we will discuss a new method which can improve students' efficiency when they learn *Programming Language*. The cultivation of Computational Thinking is the focus of this approach, and PBL provides an effective learning method. To a large extends, *Collaborative Learning* which based *Grouping Learning* will make the learning funning and interesting.

III. COMPUTATIONAL THINKING IN PROGRAMMING LANGUAGE

A. What is Computational Thinking

As Wing. Jeannette put forward, *Computational Thinking* is the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer—human or machine—can effectively carry out. *Computational Thinking* requires the following four basic techniques: [2]

Decomposition is to divide a large problem into smaller parts that are easier to handle, and then deal with these smaller parts.

Pattern recognition is the search for similarities between decomposed problems, or their common patterns. [3]

Abstraction is used to define patterns that are promoted and parameterized from specific instances. It captures the common properties of a set of objects while ignoring the irrelevant differences among them. [4]

Algorithm is effective method that can be expressed within a finite amount of space and time and in a well-defined *Formal Language* for calculating a *Function*.[5]

As we know, *Programming Languages* and their development platform are updated quickly. Even if a student learns a specific development method on a platform, it is impossible that he can use it for a long time. During the *Development Life Cycle*, the core requirements for developers are still several of the abilities mentioned earlier. If necessary, they can learn these specific operations in a short time, and the development platform will not become an obstacle.

B. Computational Thinking--A core of programming

The software development process is the transformation of the expression among different levels of an algorithm. Designer first thinks about is how to find a practical calculation model. They have to translate a language from one to another. It will be easily when people own *Computational Thinking*. So, in the face of ever-changing technical means, *Computational Thinking* is the basis for students to engage in professional work and lifelong learning. Therefore, *Computational Thinking* is the primary goal of teaching.

From the view of *Computational Thinking*, programming language need to be re-examined. The contents of *Programming Languages* can be reconstructed in accordance with the following table.

TABLE I. THE PROGRAMMING LANGUAGE IN THE VIEW OF CT

	Knowledge	**Ability**	**Contents**
The rules of language	1.Recursively 2.Formal Definition 3.Compiling 4.Memory Allocation	1. Decomposition 2.Pattern Recognition 4. Abstraction	1.Language design principles 2.Expressions 3.Function Programming 4. Abstract Data Type
Program running	1.Architecture 2.Machine Instructions 3.Procedures 4. Environments 5.Memory map	1. Pattern Recognition 2. Abstraction 3.Translation	1.Expressions 2.Statements 3.Basic Semantics 4.Formal Semantics
Algorithm	1. Enumeration 2. Search 4. Traversal 5. Match 6. Sorting 7. Retrieve 8. Randomized	1. Decomposition 2.Pattern recognition 3.Algorithm design 4.Translation	Distributed in some chapters

Obviously, such a reconstruction requires two bases. One is new textbook which can fit this point of view. Another is teachers who always emphasize these ideas. They have to pay more attention to the teaching process so that students can accept the idea of Computational Thinking in subtle way. Due to the complexity of the problem, students need to learn some new knowledge that they are not familiar with. How to monitor this process is the focus of teaching. There is also a need for specific means, such as Problem-Based Learning, Collaborative Learning and Group Learning. We can achieve our goal by these methods.

IV. NEW METHODS IN TEACHING

A. Problem-Based Learning

Unlike traditional learning methods, *Problem-Based Learning* (PBL) is student-centered. By addressing some open issues, students understand and master the basic content of learning materials. The goal of learning is not to focus on the solution of specific problems, but the skills and attributes students acquire by solving these problems.

PBL enable students to solve problems by observing and understanding the real world experience through an active learning process. The benefit of solving this problem is to make the learners acquire the skills and attributes they need. The skills and attributes include knowledge acquisition, teamwork and communication. It is important to note that the goal of PBL is not knowledge itself, but rather a universal model of learning. [6]

According to our experience, the problem should meet the following conditions.

a) Generalization: students can obtain a common approach by completing task.

b) Moderate difficulty: too simple question cannot promote students' learning ability. To the opposite, too complicated question is beyond the ability of students to learn.

c) Decomposability: students can reduce the difficulty of the problem, and this decomposition is convenient for students to work together to solve the problem.

*d) Independence:*The problem should not be too dependent on certain backgrounds, which let students cannot pay attention to the key part.

B. Collaborative Learning

PBL requires students to cooperate with each other in the process of learning, which requires the introduction of *Collaborative Learning*, thus forming a complete learning environment to achieve the desired results.

Collaborative learning is a way in which two or more people learn together. It allows collaborators to share resources and skills. [7] The members can create certain knowledge by the interpretation of the problem. People are dependent on each other and help each other under the condition of an asymmetrical role. [8]

Collaborative Learning is very suitable for PBL. Compared with the individual learning, it is beneficial to improve students' interest, promote the concentration and excitement in the collision of thought and enhance the learning efficiency. *Collaborative Learning* can also improve students' self-learning ability, change passive learning habits, and become active participants. During the designing of system, students should have good cooperation ability. *Collaborative Learning* is particularly important for achieving this goal.

C. Group Learning

Group Learning provides the conditions for PBL. [9] Yew, and Schmidt, and Hung elaborate on the cognitive constructivist process of PBL: [10]

a) By the discussion in the group, the learners understand the problems they face and activate their prior knowledge.

b) In their team, they work together to determine what problems will be study and develop possible theories or hypotheses to explain these problems. Tutors play important role because they provide some possible frameworks where students can construct knowledge relating to the problem.

c) Students work independently to research the issues after the initial team cooperation.

d) At last, the students re-group and discuss such issues again based on their new acquisitions.

In practice, *Grouping Method* has an important impact on teaching. We must consider several important factors as follow.

a) The number of each group

- In general, we may make some small groups when course is easy. Otherwise a part of students can accomplish all task and others in group will have nothing to do.
- If the team members are not familiar enough, we should divide small groups to make communicating easier.
- If the task is small, the group should be smaller than usual. Only in this case, members in group can participate effectively.
- The size of the group should make the students have the best enthusiasm. According to our experience, too large and too small groups will also reduce student participation.
- The stronger the members, the smaller the group should be. Otherwise student will finish all tasks himself.

b) The role of group members: The team leader is responsible for assigning learning task, drawing up a rule of communication. All students need to report their own learning progress and make their presentations.

*c) The principle of grouping:*Each group member has his or her own characteristics. In such group, the division of task may become an easy job. By the way, the level of each group should be about the same.

d) The Tracking of learning process: we ought to have an effective mechanism by which teachers can track the situation of students and intervene in the learning process.

e) The presentation of groups: All groups can make their presentations so as to achieve the purpose of communication between groups.

V. CONCLUSION

Today, computing technology is growing rapidly. How to design effective teaching mode becomes more important. We realize that *Computational Thinking* is the most important ability. In order to highlight it, this paper designs the mechanism of PBL and *Group Learning*. The reform re-examines the essence of computer language, extracts the elements that need to be grasped in the teaching process, and get good results.

In 2014, we began to reform the teaching of *Programming Language*. Students' learning ability has been improved. The enthusiasm of students to learn has been improved. Some progresses we can find in the Table 2.

TABLE II. SOME RESULTS OF TEACHING REFORM

		Student number	**Test results**	**Innovative software design**	**Competition Award**
Before Reform	2011	201	55.6	4	7
	2012	198	57.3	4	7
	2013	177	56.9	7	5
After Reform	2014	210	64.3	10	10
	2015	223	69.5	15	17
	2016	234	68.4	19	15

We will improve further the relevant teaching materials and experiments, summed up the experience of teaching process. We will try to extend the method to other courses and we believe that *Computational Thinking* will become the core of all curriculums in computer major.

REFERENCES

[1] https://en.wikipedia.org/wiki/Python_(programming_language)

[2] Tedre, Matti; Denning, Peter J. (2016). "The Long Quest for Computational Thinking". Proceedings of the 16th Koli Calling Conference on Computing Education Research

[3] https://en.wikipedia.org/wiki/Computational_thinking

[4] Aho and Ullman in their 1992 Foundations of Computer Science textbook define Computer Science to be “The Mechanization of Abstraction.”

[5] Rogers, Jr, Hartley (1987). Theory of Recursive Functions and Effective Computability. The MIT Press.

[6] https://en.wikipedia.org/wiki/Problem-based_learning

[7] Chiu, M. M. Flowing toward correct contributions during groups' mathematics problem solving: A statistical discourse analysis. Journal of the Learning Sciences, 17 (3), 415 - 463.

[8] Mitnik, R., Recabarren, M., Nussbaum, M., & Soto, A. (2009). Collaborative Robotic Instruction: A Graph Teaching Experience. Computers & Education, 53(2), 330-342.

[9] Chiu, M. M., & Khoo, L. A new method for analyzing sequential processes: Dynamic multi-level analysis. Small Group Research, 36, 600-631.

[10] Dillenbourgh, P., Baker, M., Blaye, A. & O'Malley, C. The evolution of research on Collaborative Learning. Retrieved February 19, 2008, from CSCL-A brief overview