International Journal of Innovation and Research in Educational Sciences
Volume 7, Issue 3, ISSN (Online) : 2349–5219

New Teaching Method of Python Programming for Liberal Arts Students

Xiangrong Shi * and Yuangao Chen
School of Information Management and Artificial Intelligence, Zhejiang University of Finance and Economics, Hangzhou, China.
*Corresponding author email id: dataworm@zufe.edu.cn

Date of publication (dd/mm/yyyy): 30/06/2020

Abstract **– The teaching of Python programming language is popular all over the world. At present, many researches focus on the language itself, but there is very little attention regarding how to teach the liberal arts students to achieve good teaching results. This article studies relevant literature and, combined with the practice of teaching reform, introduces typical measures token in our university. In summary, we have proposed a comprehensive ESA program, namely teaching Essential syntax, exhibiting Smoking code and taking Academic cases. Besides, we also consider the proper IDE, resources from the internet and effective assessment methods for liberal arts students. These details also help improve teaching effects. The data from the Teaching Management System shows that our reform program has achieved better student satisfaction, and students' attitudes toward the course have changed from fear to favor.**

Keywords **– Liberal Arts Students, Essential Syntax, Smoking Code, 80%~20% Law.**

I. INTRODUCTION

With the advent of the information society, rapidly emerging technologies are bringing new changes to all works of life. Software is redefining the way people live, and traditional industries have being affected by the wave of information and intelligence. People have to face the problem of how to integrate with information technology to provide better products and services, so as to be invincible in the competition. At the same time, the software has changed lots of aspects of people's lives, such as shopping, dining, traveling, reading, learning, paying and other behaviors. Looking at the world today, it is very different from a decade ago or even several years ago.

Today, more and more people believe that software development is not a patent for programmers, but an indispensable tool in people's work and life, just like foreign languages, office software and driving. It is no exaggeration to say that programming capability is a necessary skill in the 21st century. Many people are already considering what to learn, how to learn and such details. Indeed, there are more than 500 programming languages in the world, among which there are a number of leaders who are in the top of the TIOBE list. TIOBE (The Importance of Being Earnest) is a famous programming language leaderboard, and the Python language is currently ranked third.

The Python language owns many advantages. Guido committed to designing Python to be elegant, clear and simple. Python advocates such coding styles: writing as little code as possible, trying to write code that is easy to be understood. It is good at implementing complex logics with concise code. Besides, there are some other reasons for choosing this language, for example, open source, portability, free, object oriented feature, there are many new resources on Python available in the network, but the most important reason is that Python is a simple but powerful language to learn. This is a fair conclusion to draw from experience in teaching Python to our students in the last several years [1]. While developing programs, it is often the case that to achieve a certain object, some other programming languages may require 20~30 lines of code, while Python may require only

3~5 lines. In addition, Python has the advantages of being free, open source, and cross-platform. It supports rich data structures, object-oriented programming, and numerous built-in functions and modules. Python can cope with a large number of application scenarios, including: data analysis and visualization, automated operation and maintenance, machine learning, natural language processing, web crawlers, web server development, embedded development and so on. It is known as full stack programming Language. Many well-known companies such as Google, Yahoo, and even NASA use Python extensively.

From a learning point of view, Python is a programming language that is relatively easy to learn and get started. The learning curve is very gentle, and it is very suitable as a teaching language for computer science. Some top universities all around the world have adopted Python to teach programming courses for both computer and non-computer majors, more than 70% of the top 100 universities in the United States have set up Python programming courses [2].

However, the teaching of programming for liberal arts students is a hard work. Due to the complexity of programming and the lack of basic computer theory of students, the teaching effect is not so good, and teachers and students both feel difficult. Some typical problems include:

- Python's grammatical elements and typical application scenarios are particularly rich. Syntax concepts include: statement, keyword, identifier, data type, operation, type conversion, assignment statement, statement block, branch structure, loop structure, function, string, list, tuple, dictionary, set, object-oriented programming. In different universities, the content included in the teaching list includes: regular expressions, sorting methods, network programming, graphical user interface, in-depth OOP, etc. But, how should these rich grammatical elements and applications be selected and organized for liberal arts students?
- There is a contradiction between the limited teaching hours of the course and the massive teaching content. How to design a teaching plan to balance these factors?
- Liberal arts students are not interested in boring code. With the deepening of teaching and learning, they would be fear of difficulties. How can they overcome difficulties and feel interested in learning Python?

In response to the above problems, some educational researchers have put forward reform proposals from different angles, such as: Leo (2013) pointed out that students in liberal arts colleges have been shown to learn from peer discussion in PI (Peer Instruction) and both students and faculty have reported that they value PI in their classrooms [3], so his research implies introducing PI in the programming education of liberal arts students. Xu (2018) specifically proposed that the teaching of the Python language should fully display the characteristics of Python itself, and shouldn't bring the programming habits of other languages to Python, so that the written codes do not have the Pythonic style [2], which is also the typical problem what we found in teaching research. Based on meta cognition theory and learning ecosystem theory, Tong (2020) proposed new teaching ideas, that is to eliminate "zero foundation" before class, to practice project iteration and analyze counterexamples in class, and to strengthen exercises of silent writing, copy writing and code designing after class [4].

To address the issue of such a complicated syntax system of Python language, Kui (2018) proposed a scheme based on minimum knowledge set [5]. Tian (2019) proposed the usage of network resources to supplement the lack of knowledge and the novel curriculum assessment methods for the characteristics of the weak basic

computer knowledge of economic management majors [6]. Above inspired the research in this study.

However, these new reform programs are based on a specific angle, and solve some specific problems in the current liberal arts students' Python teaching. They have limitations and are not systematic and comprehensive.

In order to make a fundamental and thorough change to the entire teaching model of Python programming, we proposed a combination method, ESA, after extensive research and consultation. E represents the essential characteristics of Python, what teachers should focus on and lay in the syllabus. S means smoking code, teachers should explain the backbone of the code while not every detail, so that the program can exhibit its output just as the smoke rises from the firewood pile. Once we see the rising smoke, we know that the key node of the routine is correct, thus we can deliver valuable knowledge within limited teaching hours. A means teachers should give as many academic examples as possible to stir up students' interest. Examples should be carefully designed instead of taking from traditional textbooks directly. This comprehensive method helps students learn and master the core syntax of Python in programming practice. The entire set of reform schemes are as follows.

II. Redesigning Course Syllabus

While teaching liberal arts students Python programming courses, teachers should consider the characteristics of liberal arts students and the goal of talent training. We know that the main goals for liberal arts students to learn Python are as follows: One is to further understand the principles of computers. The second is to master the mode of human-computer interaction. The third is to learn the data analysis and visualization for specific subject. As for the development of web servers, the development of web crawlers and the design of GUI, it should not be listed as a priority.

Taking Zhejiang University of Finance and Economics as an example, its subject major belongs to economics and management. Liberal arts students make up 85% of the university. They study Python programming like science and engineering students, but the training plans for them are different, and the syllabus designed for them is shown in Table 1.

Table 1. Different teaching content for different students.

Audience	Common content	Different content
For science and engineering students	• Basic syntax • Branch & loop • Functions & modules	• Regular expression • Object oriented programming • Graphic user interface
For liberal arts students	• String • List & tuple • Dictionary & set	• Numpy (module for data processing) • Matplotlib (module for data visualization)

After adjusting the syllabus described in Table 1, liberal arts students can conduct more targeted learning, which is in line with the setting of talent training goals. However, the content of the common part should not be taught exactly the same. It is necessary to apply the ESA method to achieve a more ideal training effect.

III. Comprehensive Reform of Teaching Methods

A. *Teaching Essential Syntax*

In China, many students have already learned programming languages such as Basic, C or Java script in middle school. Among these, some basic syntaxes are very similar to Python. Therefore, in the stage of university education, we can set these common contents as self-study part, and focus on the essential syntax of Python. In our teaching practice, we divided Python's syntaxes into two categories according to Table 2.

Table 2. Pick out the essential syntax of Python.

Python's Syntax	What is essential and should be emphasized
Basic syntax	• About integer, you can represent an almost infinite integer • Operator in and is • Assignment statement, in Python assignment means labeling a variable • Block, in Python several statements are combined into a block by using same indentation depth
Branch & loop	• while...else... • for...else...
Functions & modules	• In python, a function will return None if return statement is omitted • Lambda function • Essential of parameters passing • Useful functions: map, eval, range
String	• Immutable object and mutable object • A string is an immutable object • Useful functions: len, max, min
List & tuple	• A list is mutable object, while a tuple is an immutable object • List comprehension • Generator expression
Dictionary & set	• Dictionaries are unordered containers • Sets have unique elements
Others	• Advanced function parameter form • Useful functions: reduce, filter, sorted, zip, enumerate, product, any/all, exec • Iterable and Iterator • From generator expression to functional generator • Useful modules: math, random, time, sys, os, itertools

B. *Exhibiting Smoking Code*

While explaining programs, it will unnecessarily take up too much class time if the entire routine is introduced in detail, which is not helpful for students to grasp the essence of the kernel knowledge. Therefore, as an experienced teacher, you should make the code smoking. So, what is smoking code? it comes from the idea of smoke test. To put it simply, smoke test is the most basic and fundamental test for the purpose of confirming overall correctness. If smoke test fails, it means that there are major problems in whole program. If the smoke

test passes, it means that the core steps of the routine are verified to be valid.

We can utilize the smoking code to teach programming, which is based on the law of 80%~20%. For example, following is a piece of code to calculate the real roots of the quadratic equation of one variable. It handles all possible input errors, and it is perfect, see Code 1.

Code 1. Calculate the real roots of a quadratic equation.

```
import math
prompt = "Please input 3 integers:\n"
while True:
inp = input(prompt)
   try:
      step = 0
      a, b, c = map(float, inp.split(","))
      step = 1
      delta = b*b - 4*a*c
      root = math.sqrt(delta)
      step = 2
      rst1 = (-b + root)/(2*a)   #line 12
      rst2 = (-b - root)/(2*a)   #line 13
      break
   except:   #line 15
      if step == 0:
         print("The inputs are bad, please check and reenter.\n")
elif step == 1:
         print("Please keep b**2 - 4*a*c >= 0.\n")
      else:
         print("a should not be zero.\n")
print(rst1, rst2)
```

It can be seen from Code 1 that although the key step of the program has only two statements, that is line 12 & 13, rst1 = (-b + root)/(2*a) and rst2 = (-b - root)/(2*a). But there are a lot of supporting steps in the front, these steps are to make the program run successfully to get here. Assuming that this program has been run thousands of times, and most of the inputs are in line with the specifications, then the except clause on line 15 rarely has a chance to run. Therefore, each part of the code in the program, their chances of being executed are not equal, there is so called 80% ~ 20% law.

In programming, the program executes 20% of the code in 80% of the time, and 80% of the code in 20% of the time. Most of the 80% of the code that is rarely executed is used to make the program more robust. If the input is verified and judged, the existence of these codes is very necessary. In reality, the ratio may be even

larger than 8:2. For teachers, we should grasp the kernel part of 20%, so that the audience can see the input and output and then take the main idea as soon as possible.

In the above program, reading the input, calculating and outputting the result are the three key steps. We can rewrite the original code to the smoking code as shown in Code 2.

Code 2. Calculate the real roots of a quadratic equation using smoking code.

```
import math
inp = input("Pls input 3 numbers.\n")
a, b, c = map(float, inp.split(","))
delta = b*b - 4*a*c
root = math.sqrt(delta)
rst1 = (-b + root)/(2*a)
rst2 = (-b - root)/(2*a)
print(rst1, rst2)
```

Through the simplest example, we can see the correct input and output if the test example is appropriate, therefore, we know that the program is working, and only the details need to be added. This process is like lighting a fire under a pile of firewood. When the smoke is seen from above through the firewood pile, we know that the flame has already ignited and the whole path has been unblocked.

C. *Taking Academic Cases*

The importance of examples in exercises in Python Programming is perhaps the highest of all courses. In order to obtain good teaching results and stimulate learning enthusiasm, it is necessary to select a batch of good examples carefully. Most textbooks do not pay attention to this point. The examples they use may not matter to students who major in computer science, but for liberal arts students, examples that are too abstract may make them lose their interests for learning, academic cases should be considered.

In our teaching practice, we focus on combining the design of examples with the students' majors. For example, Code 3 is an example of temperature conversion from Celsius to Fahrenheit using user defined function.

Code 3. Temperature conversion from C to F.

```
def C2F(c):
    f = c * 1.8 + 32
    return f
c = 30
f = C2F(c)
print(f)
```

For students majoring in finance, I usually redesign this example as a similar one, which is to conduct exchange rate conversion, as shown in Code 4.

Code 4. Conversion based on exchange rate.

```
exchange_rate = 7.07
def USD2RMB(usd):
rmb = usd * exchange_rate
    return rmb
usd = 30
rmb = USD2RMB(usd)
print(rmb)
```

For students majoring in management, I usually redesign this example as a similar one, which is to calculate labor price, as shown in Code 5.

Code 5. Calculating labor price.

```
price = 30
defcalc Payment(hours):
    payment = hours * price
    return payment
hours = 12.5
payment = calc Payment(hours)
print(payment)
```

By setting up academic cases related to their majors, we can make students understand Python's instrumentality, so as to stimulate their motivation to learn this popular language.

D. *Choosing Proper IDE*

For learning Python, choosing a good IDE (Integrated Development Environment) will make the learning process smoother. For liberal arts students, we believe that the choice of tools should be as light as possible, and do not make the process of familiarity with tools an extra burden. Thus, we recommend a light IDE called thonny [7].

Thonny is a new Python IDE mainly developed in Institute of Computer Science of University of Tartu, Estonia. It can make program visualization a natural part of the beginners' workflow. Among its prominent features are different ways of stepping through the code, intuitive visualization of the call stack, step-by-step expression evaluation and a mode for explaining the concepts of references and heap [8]. In Windows environment, the thonny's start interface is shown in Figure 1.

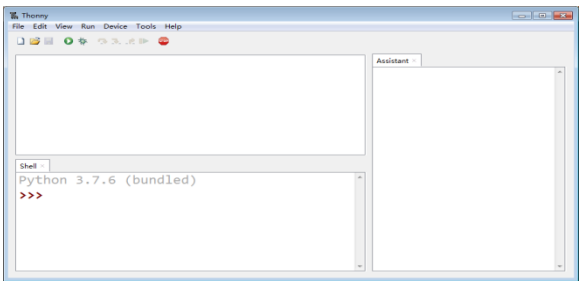

Fig. 1. An IDE for beginner: Thonny.

Besides, installation of Thonny does not require the installation of native Python, it comes with the latest version of Python, the current version is 3.7.6. Because Thonny removes some features that are not commonly used, it consumes very little memory, so it starts quickly and the user's experience is good.

E. *Introducing the Internet + Teaching Method*

Python is an emerging information technology course. At present, high-quality teacher resources are scarce, but there are a large number of high-quality teaching resources on the Internet. So, teachers can make full use of network resources, gather excellent materials scattered on the Internet, and recommend to students, including original electronic textbooks, courseware and even teaching videos. The following are some excellent materials and teaching resources:

- Python for Everyone, Charles R. Severance, https://www.py4e.com.
- Learning Python, Mark Lutz, O' REILLY.
- Fluent Python, Clear, Concise, and Effective Programming, Luciano Ramalho, O' REILLY.
- Think Python: How to Think Like a Computer Scientist, Allen B. Downey, O' REILLY.
- Effective Python 59 Specific Ways to Write Better Python, Brett Slatkin, Pearson.
- Socratica's channel and class on Python.
- Kevin Markham (founder of data school)'s channel and class on Python data analytics.

Also, we can use the new media platform to build a learning group and a public account, and push scientific and technical essays on Python programming skills on the public account to stimulate students' interest and enthusiasm for learning [9]. We can also use the Internet platform to answer questions on the course and enhance students' recognition of the course. In order to make up for the shortage of classroom teaching hours, the MOOC platform that provides flip teaching can be adopt to give kernel lecture in the videos before class and only answer questions or practice in class [10].

F. *Reforming Curriculum Assessment Method*

The assessment of the course must reflect the students' understanding of the basic syntax, mainstream usage, mastery and application ability. Therefore, it is not appropriate for teachers to design examination into the style of mechanically repetition of some knowledge to be learned [11]. In some similar courses, teachers often use "big homework" or "a complex task" to check the level of students, but this open inspection method is difficult to prevent fraud and will cause unfairness. We recommend a combination of computer based and paper filled examination style. Students answer on the test paper, but some problems can be solved only by programming on computers.

In traditional paper based test, such questions as shown in Test 1 are very common, but we do not think this should be the focus of the exam, because we should not assume that the user does not have a computer.

Test 1. Traditional paper based test.

1. Run len ("I love Python") in the shell, the output is _____.
2. Run [1, 5, 3]. sort() in the shell, the output is _____.

In the computer based test, questions in Test 1 are not valuable, because the candidate owns a computer. Therefore, problems need to be designed more intensively as shown in Test 2.

Test 2. Traditional paper based test.

1. To slice a string s = "ZUFE" using s [_____], you can obtain its reversed version "EFUZ".

2. Expression list(map(lambda r:round(_____,2), range (1, 6))) output a series of circumference with the radius 1~5: [6.28, 12.57, 18.85, 25.13, 31.42].

By carefully designing this reversed-style questions, we can examine students' levels of knowledge mastery and flexible application more comprehensively.

IV. Comparison of Students' Satisfaction

In the two academic years of 2018 and 2019, I took courses of Python Programming for different majors of Zhejiang University of Finance and Economics. The students were from the School of Accounting and the School of Finance. After the first semester, I felt that the teaching effect was not very satisfactory, and then initiated a series of reforms as introduced above. At the end of each semester, students are requested to give an evaluation of the course, including satisfaction indicators and textual comments.

Table 3 gives the students' satisfaction index before and after the teaching reform. These data are collected from the Teaching Management System of the Academic Affairs Office.

Table 3. Changes in student satisfaction before and after teaching reform.

Semester	**2018-2019-2**		**2019-2020-1**	
Index	**Number**	**Proportion**	**Number**	**Proportion**
Very satisfied	21	65.63%	19	86.36%
Basically satisfied	10	31.25%	3	13.64%
Dissatisfied	1	3.13%	0	0.00%

It can be seen that the very satisfied ratio rises from 65.63% to 86.36%, and basically satisfied and dissatisfied ratio drops from 34.38% to 13.64%.

Meanwhile, there are 16 negative comments in the 44 textual review in the first semester, which mainly reflect the difficulty of the course being beyond the students' understanding, such as:

1. Sometimes I cannot understand.
2. I hope it can be more detailed.
3. Content is jumping, sometimes I don't understand.
4. I hope the teacher can give more details in the class....The zero-based students are very painful.
5. Can you speak a little bit more slowly.
6. It's okay, I just want to say the content of the class is a bit confusing.
7. I hope it's slower and simpler.
8. Teach more carefully.

9. I hope it can be simpler.
10. We are all students who have zero foundation in computer language. I hope that teachers can consider the basics in the future.
11. It should be more detailed.

In the second semester (after reform), there were only 1 similar opinion appeared in the 32 text comments. The comment about the course changed from difficult to understandable. The 2 comments are as follows:

1. I hope the teacher will speak slowly in the experimental class. (The only review similar to before the reform.)
2. Mr. Shi is serious and responsible, and the course content is very thorough. (Regarding the difficulty of the course, students gave the opposite conclusion.)

V. CONCLUSION

With the advent of the era of big data and artificial intelligence, it is necessary for students of all majors to learn programming languages [12]. Currently, Python is a concise and efficient language suitable for beginners to learn. This study combines the conclusions from literature and the practice of our own teaching reform, and puts forward several reform suggestions for the teaching of Python for liberal arts students.

According to the analysis, we believe that liberal arts students learn Python mainly to cultivate computer thinking and engage in data processing in the future, rather than become real programmers. Our suggestions include:

1. We need to adjust the syllabus to discard some content that is valuable to the computer majors but not valuable to liberal arts students.
2. Based on the fact that students already have a certain degree of programming learning experience, we believe that the essential syntax of Python should be emphasized.
3. In order to help students grasp the key knowledge of Python in limited hours, we should not pursue exhibiting a perfect program, but give a piece of smoking code.
4. In order to maintain students' interest in learning and let them understand the application scenarios of Python in their own majors, teachers should design academic cases close to their major.

Furthermore, we suggest that we should make full use of high-quality resources on the Internet to assist teaching. It is also necessary to reform the assessment and evaluation methods so that students feel fair in the course. Practice in Zhejiang University of Finance and Economics shows that our teaching reform has achieved better student satisfaction. Students' attitude towards the course has changed from fear and worry to recognition and favor. In the future, we will focus more on the reform of course content and the design of teaching examples to achieve better results.

ACKNOWLEDGMENT

This work was partially supported by Chinese National Planning Office of Philosophy and Social Science Project (17BGL047) and Teaching Reform Research Project of Zhejiang Higher Education in the 13th Five-

Year Plan (jg20180201).

REFERENCES

[1] Rufinus, J. and Y. Kortsarts, Teaching an introductory programming course for non-majors using Python. Director, 2006: p. 07.
[2] Xu, W., Feasibility analysis of opening Python programming course for Liberal Arts Major. The Science Education Article Collects, 2018(33): p. 59-61,68.
[3] Porter, L., et al. Peer instruction in computer science at small liberal arts colleges. in Proceedings of the 18th ACM conference on Innovation and technology in computer science education. 2013.
[4] Tong, L., Teaching design of Python course for liberal arts majors based on Meta cognition. Computer Education, 2020(1): p. 148-150,154.
[5] Kui, X., et al., Teaching method reform of python language programming course based on minimum knowledge sets. Mechatronic Systems and Control, 2018. 46(4): p. 181-186.
[6] Tian, X., et al., Research on big data principles and practice course constructing of economic management major: Teaching design based on Python Language. Computer Knowledge and Technology, 2019. 15(34): p. 145-146.
[7] URL: https://thonny.org/.
[8] Annamaa, A. Thonny, a Python IDE for learning programming. In proceedings of the 2015 ACM conference on Innovation and Technology in Computer Science Education. 2015.
[9] Liang, T., et al., Reform and practice of mixed teaching of basic programming course in new media environment. Computer Education, 2019.8: p. 132-136.
[10] Zhao, Y., et al., Construction of MOOC designed with C language programming guided by computational thinking. Experimental Technology and Management, 2018, 35(4): p. 147-150.
[11] Victor T. N. and Joel C. A., Improving Non-CS Major Performance in CS1, SIGCSE' 15: Proceedings of the 46th ACM Technical Symposium on Computer Science Education, 2015, p. 558–562.
[12] Liu Q., Teaching practice of Python Programming Course in Big Data Era, International conference on Computer Science and Education Technology (CSET 2018), 2018: p. 1-4.

AUTHOR'S PROFILE

First Author
Dr. Xiangrong Shi received the Ph.D. degree from the department of Control Science and Engineering, Zhejiang University, in 2014. He currently is an Associate Professor with the School of Information Management and Artificial Intelligence, Zhejiang University of Finance and Economics, China. His research interests include educational management, data mining and artificial intelligence.

Second Author
Dr. Yuangao Chen received the Ph.D. degree from School of Management, Zhejiang University, in 2013. He currently is a Professor with the School of Information Management and Artificial Intelligence, Zhejiang University of Finance and Economics, China. His research interests include educational management, supply chain and new retail.