3

처음 배우는 프로그래밍 언어로 무엇을 어떻게 가르칠까?

대전대학교 ▌이 찬 도*

"교수님, 프로그램 잘 하는 학생 있으면 좀 보내주세요. 요즈음에는 좋은 프로그래머를 찾기가 정말 힘드네요."

"교수님, 프로그래밍 너무 어려워요. 이거 반드시 배워야 해요?"

프로그램은 정보통신 기술을 구현하는 필수적인 요소이다. 프로그램을 잘 만들기 위해서는 창의적이고 논리적인 사고를 갖추어야한다. 프로그래밍 기술은 자동차 운전처럼 며칠 만에 습득할 수 있는 단순한 운동기능이 아니다[1]. 훌륭한 건축물을 세우는 데는 몇 년이 걸린다. 날림 공사를 하지 않기 위해서는 기초를 튼튼히 해야만 한다. 아무리 겉모습이 아름답더라도 기초가 부실한 건물은 비가 샌다던지 하는 여러 문제점을 드러내고 따라서 그 기능을 제대로 다 하지 못하게 마련이다. 프로그래밍도 마찬가지이다. 기초가 튼튼해야 한다. 그러기 위해서는 처음 프로그래밍을 배우는 학생들이 재미있게 프로그램에 입문하도록 도와주어야 한다. 특정 언어의 문법을 가르치는 것 못지않게 논리적으로 문제를 풀 수 있는 능력을 길러주는 것이 중요하다. 그러나 현실은 그렇지 못한 것 같다. 프로그래밍 언어의 문법을 가르치는데 너무 치중하다 보니 학생들은 쉽게 싫증을 느끼고 프로그래밍을 기피하게 된다. 어려서부터 인터넷을 접하고 컴퓨터 능력이 뛰어난 G세대 학생들이 정작 프로그래밍은 싫어한다면 그 원인은 그들을 가르치는 우리에게 있는 것이 아닌지 돌아보아야 한다. 산업현장에서는 쓸 만한 프로그래머가 없다고 아우성이고 학생들은 재미도 없고 취업전망도 불투명하며 보수도 적은 프로그래머란 직업에 별로 매력을 느끼지 못 한다. 문제는 무엇일까? 물론 프로그래머를 적절히 대우하지 않고 3D업종으로 여기는 사회풍조도 문제이지만 우리의 교육방식에는 문제가 없었을까? 어떤 언어를 어떻게 가르칠까 깊은 고민이 필요하다고 본다.

가장 중요한 것은 자질이 뛰어난 학생들을 많이 유치하고 그들이 흥미를 느끼고 즐겁게 프로그래밍을 배울 수 있는 환경을 조성하는 것이다. 언어도 잘 선택해야 하겠고, 언어를 배우는 소프트웨어 환경도 GUI를 포함하는 사용자 중심이 되어야 하겠다. 또한 프로그래밍 예제나 실습 과제도 텍스트 기반을 벗어나 멀티미디어를 다룬다면 학생들의 배우고자 하는 욕구를 잘 끌어낼 수 있을 것이다. 그동안 우리는 가장 많이 필요로 하는 언어라서 처음부터 C(C++)나 Java를 가르쳤다[2]. 그러나 이 언어들은 바로 실용적으로 사용할 수 있는 프로그램용이라 모든 복잡한 요소들을 다 가지고 있다. 게다가 실습예제나 과제도 주로 텍스트 기반이다. 그러다 보니 처음 배우기에는 너무 어렵다. 왜 배우는지, 어떻게 쓰이는지 전체 그림을 그릴 수도 없다. 자연스럽게 흥미를 잃게 되고 수업을 포기하는 학생들이 늘게 된다. 어린아이가 외국어를 배울 때 처음부터 복잡한 문법부터 시작하지는 않는다. 아이들이 어려서부터 즐겁게 놀면서 외국어에 자연스럽게 친숙해지도록 환경을 만들어 준다면 해당 언어를 쉽게 배운다는 것은 교육계의 정설이다. 프로그래밍 언어도 하나의 언어인 이상 학생들이 놀이 하듯 즐기며 쉽게 배울 수 있도록 환경을 꾸며주는 것이 제일 중요하다. 대학에 들어 와서 처음 프로그래밍 언어를 배우는 학생들이 기초를 튼튼히 다질 수 있도록 도와주는 것이 우리의 첫 번째 목표가 되어야 한다. 처음 마당에서 잘 놀아본 학생은 다음 놀이터에서는 어떻게 놀게 될까 기대하게 되고, 이러한 성취감과 기대감은 훌륭한 프로그래머를 만드는 원동력이 된다[3].

그러면 처음 배우는 언어가 갖추어야 할 기준에는 어떠한 것이 있을까[4]?

(1) 가능하면 쉽고 단순해야 한다. 외국어를 배울 때 가장 간단한 대화부터 배워 나가 듯 프로그래밍의 개념을 습득하고 논리적인 문제해결 방법을 기를 수 있는 기본 도구를 갖추고 있다면 충

* 종신회원

분하다. 우리의 목표는 바로 쓸 수 있는 실용기술을 가르치는 것이 아니라 기본소양을 기르는 것이 되어야 한다.

(2) 흥미를 갖고 즐겁게 배울 수 있어야 한다. 외부의 요인에 의해 억지로 하는 학습은 효과가 적다. 스스로 내면에서 솟구치는 욕구가 강할 때 학습효과는 가장 높다.

(3) 프로그램의 결과를 즉시 확인할 수 있어야 한다. 프로그램을 작성하여 컴파일과 디버깅을 마친 후 실행을 통해 결과를 보는 것이 일반적인 프로그램 개발 단계이다. 이 과정은 매우 지루하여 학생들을 지치게 한다. 컴파일 에러는 외계어 같아서 이해할 수 없고, 천신만고 끝에 컴파일을 마친 후 실행을 하니 원하는 결과가 안 나와 좌절한다. 처음 배우는 프로그래밍 언어는 이러한 여러 절차를 일일이 거칠 필요 없이 프로그램 작성에서 결과 출력까지 한 눈에 확인할 수 있어야 한다.

위에 열거한 기준을 만족시킬 수 있는 언어로는 무엇이 있을까? 요즘 새롭게 주목 받고 있는 몇 가지를 소개하고[5] 그 언어들을 어떻게 통합하여 가르칠 수 있는지 살펴보고자 한다.

Python[6]은 로썸(Guido van Rossum)이 1991년 처음 발표한 언어이다. 절차적이고 객체 지향적이며 함수적인 모든 문법구성 요소를 지원한다. 인터프리터도 제공하므로 프로그램 개발이 쉽다. 무엇보다도 문법이 단순하여 쉽게 배울 수 있으며 풍부한 라이브러리를 제공하므로 멀티미디어와 인터넷 프로그램 등 다양한 어플리케이션을 어렵지 않게 만들 수 있다. Windows, Macintosh, Linux 등 거의 모든 플랫폼을 지원하며 오픈 소스라 무료로 사용할 수 있으며 C(C++)나 Java 등의 다른 언어와 쉽게 결합할 수 있다. 그림 1은 "안녕!"하는 인사말을 출력하는 예이다.

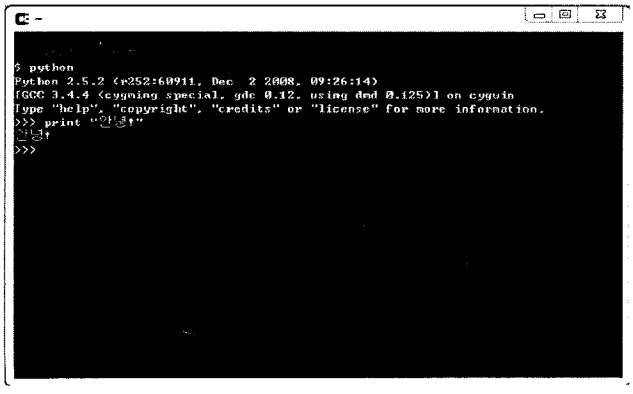

그림 1 Python 프로그램의 예

Scratch[7,8]는 MIT에서 개발하였으며 누구나 쉽게 스토리, 게임, 애니메이션, 시뮬레이션을 만들고 서로 공유할 수 있도록 디자인되었다. 마치 레고 블록을 쌓는 것처럼 조각들을 이어 붙여 프로그램을 만들고 바로 실행하여 결과를 확인할 수 있다. 다국어를 지원하므로 한글로 메뉴를 보고 한글로 출력하는 것도 문제가 없다. 개인이 만든 프로젝트는 커뮤니티를 통하여 공유가 가능하고 다른 프로젝트를 다운 받아 리믹스 하는 것도 가능하다. 그림 2는 Scratch의 실행예이다. Scratch의 캐릭터는 스프라이트(sprite; 요정)라고 불리는데 동물, 사람, 교통수단 등 다양하게 제공되는 여러 파일중 하나를 선택하여 불러올 수도 있고 직접 그려서 사용할 수도 있다. 그림에서 가장 왼쪽에 보이는 여러 프로그램 구성요소 중 원하는 것을 스크립트 아래에 끌어다 놓으면 프로그램이 만들어진다. 예에서는 "'깃발'이 클릭되었을 때" "'안녕!' 말하기"를 실행한다.

Alice[9,10]는 현재 CMU에서 개발하고 있으며 프로그램 명령문을 나타내는 그래픽 타일을 끌어다 놓아 3D프로그램을 한다. Scratch와 마찬가지로 대화형(interactive)으로 프로그램을 할 수 있어서 바로 결과를 눈으로 확인할 수 있다. 한 가지 아쉬운 점은 한글이 지원되지 않아 메뉴도 영어이고, 한글 입출력도 가능하지 않다. 그러나 대학생 정도의 영어실력이라면 메뉴를 읽고 사용하는 데 큰 불편은 없으리라 여겨진다. 그림 3은 Bob이 "Hello"라고 말하는 프로그램의 예이다. 그림에서 왼쪽 위에 보이는 것이 Alice World의 구성요소이다. camera, light, ground는 기본으로 주어지고 bob은(그림에서는 가운데에 검은 사각형만 나타나고 보이지 않지만) "add objects"를 통해 추가했다. 왼쪽 아래에 보이는 것이 구성요소들을 프로그램할 수 있는 메서드들인데 원하는 것을 오른쪽 아래의 해당하는 이벤트 메서드에 가져다 놓으면 된다. 다양한 이벤트는 오른쪽 위의 창을 통해 조정한다. 프로그램의 실행은 "Play"를 누르면 된다. 그림에서는 bob이 "Hello"를 2초간 말하는 화면의 캡처이다.

위에서 쉽고 재미있게 프로그램을 배울 수 있는 Python, Scratch, Alice를 소개하였다. 이 언어들을 다양하게 조합하여 처음 프로그램을 배우는 학생들에게 가르친다면 즐겁게 프로그램을 배우며 기초도 튼튼하게 다지고 창의적이고 논리적인 문제풀이 방법도 함양할 수 있으리라 사료된다.

(1) Python을 한 학기 가르친다. Python은 C(C++)나 Java보다 문법이 훨씬 간단하고 쉬워서 학생들

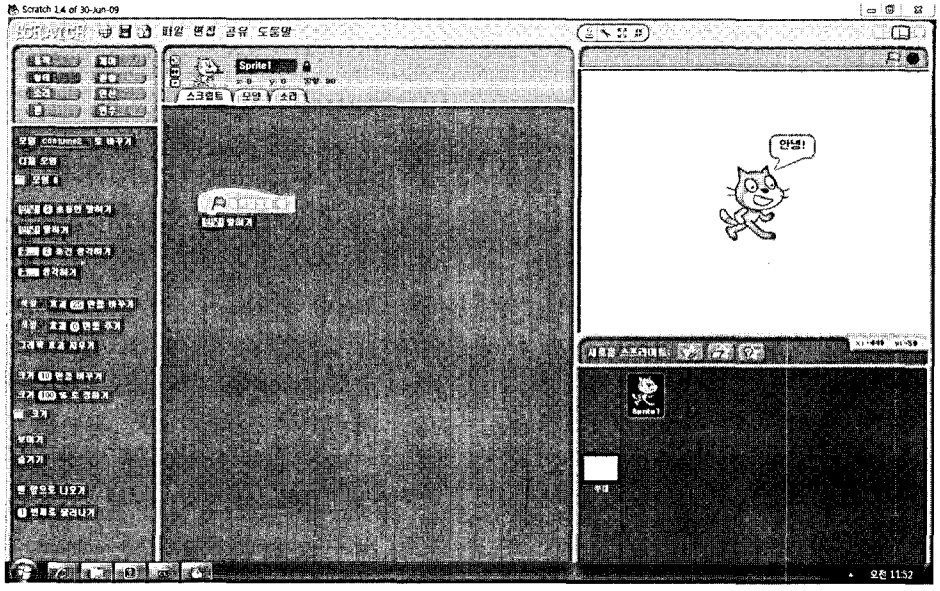

그림 2 Scratch의 실행 화면

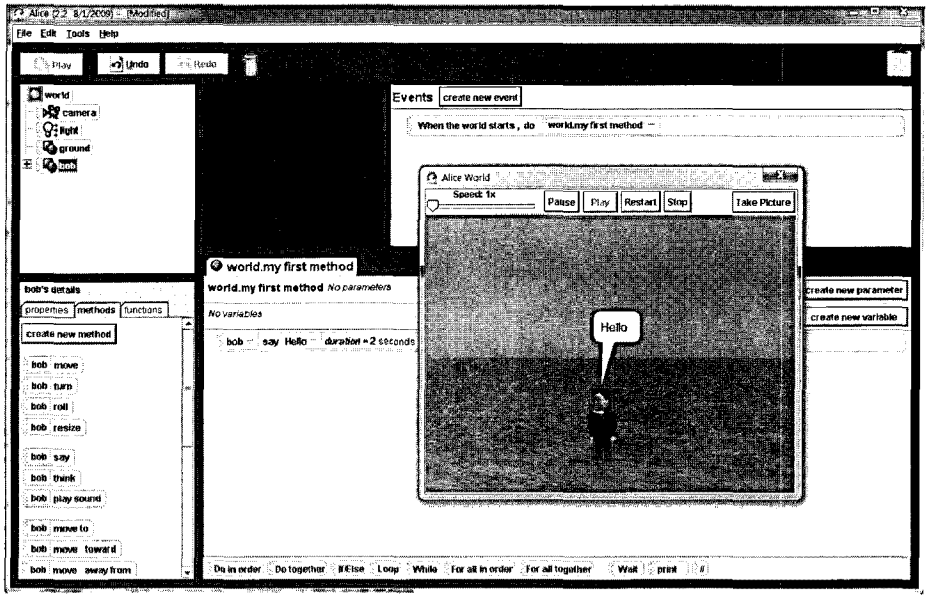

그림 3 Alice의 실행 화면

이 별 어려움 없이 프로그래밍의 거의 모든 개념을 익힐 수 있으며 어떻게 창의적으로 프로그램을 만들어야 하는지 튼튼한 기초를 쌓을 수 있다. 또한 다양한 라이브러리를 이용하여 멀티미디어 프로그램을 만든다면 학생들의 흥미도 배가시키고 학습 성취도도 높일 수 있다[11]. Python을 배운 후에 C(C++)나 Java로 옮겨가도 별 무리 없이 잘 배울 수 있다[12].

(2) Scratch나 Alice를 이용하여 몇 주간 기본 개념을 가르친 후 Python, C(C++), 또는 Java로 옮겨간다. 이 방법은 미국 여러 대학에서 다양하게 시도 하고 있는데 이전에 프로그래밍 경험이 전혀 없는 학생들은 Scratch나 Alice를 통해 기초를 익힌 후에 Java를 배운 것이 매우 유용하였다고 평가하고 있다[13,14]

Google의 학술검색을 통해 최근 몇 년 간을 검색해 보면 Python, Scratch, Alice를 대학 첫 프로그래밍 과목에서 가르친 결과를 보고하는 논문이 제법 많은 것을 확인할 수 있다. 이제 우리도 C(C++)나 Java를 텍스트 기반으로 가르치는 전통적인 방법을 지양하고 새로운 방법을 시도해 보면 좋지 않을까 제안해 본다.

마지막으로 주제와는 조금 다르지만 Go[15] 언어를 잠깐 언급하고 졸고를 마치려 한다. Go언어는 구글에

서 개발하여 2009년 11월 공개한 C를 대신할 만한 시스템 언어이다. 아직은 개발단계의 실험적 언어이긴 하지만 컴파일 속도가 빠르고 실행 또한 빠른 언어라고 발표하고 있다. C 대신 Go를 가르치는 것도 당장은 아니더라도 하나의 선택사항으로 염두에 두고 Go가 어떻게 진화해 나가는지 시장에서는 어떻게 받아들이는지 관심을 갖고 지켜보아야 하겠다.

참고문헌

[1] Norvig P., Teach Yourself Programming in Ten Years. http://www.norvig.com/21-days.html

[2] http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

[3] Koffman E., "All I Really Need to Know I Learned in CS1," SIGCSE 2009 Keynote Address. (available at http://www.temple.edu/cis/directory/tenure/documents/KoffmanSIGCSESlides.pdf)

[4] Gupta D., "What is a Good First Programming Language?," Crossroads, Vol. 10, No. 4, pp.7-7, 2004. (available at http://www.acm.org/crossroads/xrds10-4/firstlang.html)

[5] Intersimone D., "Scratch, Squeak, Alice and Go -- programming for kids, adults and everyone else," Computerworld Blogs, November 23, 2009. (available at http://blogs.computerworld.com/15138/scratch_squeak_alice_and_go_programming_for_kids_adults_and_everyone_else)

[6] http://www.python.org

[7] http://scratch.mit.edu

[8] Resnik M. et al., "Scratch: Programming for All," Communications of the ACM, Vol. 52, No. 11, pp. 60-67, 2009.

[9] http://www.alice.org

[10] Kelleher, C. and Pausch, R., "Using storytelling to motivate programming," Communications of the ACM, Vol. 50, No. 7, pp. 58-64, 2007.

[11] Guzdial, M. and Ericson, B., Introduction to Computing and Programming in Python: A Multimedia Approach, Prentice Hall, 2009.

[12] Enbody, R. J., Punch, W. F., and McCullen, M., "Python CS1 as preparation for C++ CS2," ACM SIGCSE Bulletin, Vol. 41, No. 1, pp. 116-120, 2009.

[13] Malan, D. and Leitner, H., "Scratch for budding computer scientists," ACM SIGCSE Bulletin, Vol. 39, No. 1, pp. 223-227, 2007.

[14] Lorenzen T. and Sattar A., "Objects first using Alice to introduce object constructs in CS1," ACM SIGCSE Bulletin, Vol. 40, No. 2, pp. 62-64, 2008.

[15] www.golang.org

이 찬 도

1975 서울대학교 독어교육과 학사
1984 Arizona State University 독문학과 석사
1987 Indiana University 컴퓨터과학 석사
1991 Indiana University 컴퓨터과학/인지과학 박사
현재 대전대학교 정보통신공학과 교수
관심분야 : 자연어처리, 지능형에이전트, 문서범주화, 소프트컴퓨팅, 하이퍼문학, 컴퓨터교육
E-mail : cdlee@dju.kr