Journal of The Korean Association
of Information Education Vol. 24, No. 1,
February 2020, pp. 117-129

http://dx.doi.org/10.14352/jkaie.2020.24.1.117

파이선(Python) 학습을 위한 평가 프로세스 설계

고은지 · 이정민
이화여자대학교 교육공학과

요약

본 논문은 기존 컴퓨팅 사고력 평가 연구를 분석하고 보완하여 형성적 관점에서 컴퓨팅 사고력을 평가하는 방안을 탐색하고, 텍스트 기반 프로그래밍 언어인 파이선을 활용한 프로그래밍 언어 학습 평가를 위한 평가 프로세스를 설계하기 위해 수행되었다. 이와 같은 목적으로 컴퓨팅 사고력 영역을 탐색하고 평가 설계에 관련된 연구를 분석하였다. 또한, 초보자가 학습하는 파이선 프로그래밍의 학습 영역을 확인하고, 파이선 학습을 통해 획득할 수 있는 컴퓨팅 사고력 영역을 규명하였다. 이들을 종합하여 컴퓨팅 사고력에 해당하는 구문을 분석하여 피드백을 제공하는 평가 방법을 설계하였다. 아울러, 순서도와 의사코드를 활용하여 아이디어를 나타내게 함으로써 반성적 사고를 통한 자기평가가 가능하게 하고, 커뮤니티를 활용한 코드공유 및 의사소통을 통해 동료피드백이 가능한 평가 프로세스를 설계하였다는 데에 본 연구의 시사점이 있다.

키워드 : 파이선, 컴퓨팅 사고력, 텍스트 기반 프로그래밍 언어, SW교육, 프로그래밍 평가

Assessment Process Design for Python Programming Learning

Eunji Ko · Jeongmin Lee
Dept. of Educational Technology, Ewha Womans University

Abstract

The purpose of this paper is to explore ways to assess computational thinking from a formative perspective and to design a process for assessing programming learning using Python. Therefore, this study explored the computational thinking domain and analyzed research related to assessment design. Also, this study identified the areas of Python programming learning that beginners learn and the areas of computational thinking ability that can be obtained through Python learning. Through this, we designed an assessment method that provides feedback by analyzing syntax corresponding to computational thinking ability. Besides, self-assessment is possible through reflective thinking by using the flow-chart and pseudo-code to express ideas, and peer feedback is designed through code sharing and communication using community.

Key words : Python, Computational thinking, Text-based programming language, SW education, Programming assessment

---

이 논문은 2019년 대한민국 과학기술정보통신부와 한국연구재단의 지원을 받아 수행된 연구임(NRF-2019R1F1A1040874)
교신저자: 이정민(이화여자대학교 교육공학과)
논문투고 : 2020-02-05
논문심사 : 2020-02-11
심사완료 : 2020-02-20

1. 서론

세계 산업의 흐름이 지식·정보 중심의 사회로 변화하고 있으며, 소프트웨어(SW)는 국가발전에 있어서 주요한 원동력으로 자리 잡았다. 이에 따라 교육계에서도 학생들의 기초 코딩 능력을 배양하는 것을 목표로 하여, 우리나라에서도 2018년부터 SW 교육이 교과과정에 점진적으로 포함되기 시작하였다.

교과과정에서는 특정 프로그래밍 언어의 기능을 습득하는 것뿐만 아니라, 문제해결을 위한 프로그래밍 설계 및 개발 프로세스를 통해 컴퓨팅 사고력(Computational Thinking)(이하 CT)을 향상하는 것에 중점을 두고 있다. 또한, SW 교육의 평가 방법에 대해 '다양한 평가도구를 사용하여 SW 교육을 통한 컴퓨터 사고력의 향상을 측정'이라고 명시하였고, 프로그래밍을 통해 프로젝트에 포함된 문제 선택, 아이디어 설계, 프로그래밍 구현 및 협업에 대한 포괄적인 평가를 제안하였다[33].

반면, SW 테스트는 '오류를 발견하기 위해 프로그램이나 시스템을 실행하는 프로세스', '프로그램에서 버그를 찾는 것', '프로그램의 올바른 동작을 보여주는 것', '프로그램이나 시스템이 해야 할 일을 수행한다는 확신을 심어주는 과정' 등으로 정의된다[16][36]. 이러한 테스트 과정의 부재는 프로젝트의 실패를 불러오며, 즉, SW 교육에서도 프로그래밍한 코드에 대한 평가 및 테스트가 이루어지지 않으면 학습의 실패나 지연을 일으킬 수 있음을 알 수 있다.

현 시점에서 SW 교육의 평가는 CT 평가에 초점이 맞추어져 있으며, 지금까지 개발된 대부분의 자동화된 CT 평가도구는 Brennan과 Resnick(2012)이 제안한 CT 평가 프레임워크에 기초하여 컴퓨팅 개념을 측정하는 방식으로 평가하고 있다[35][39][49]. 그러나 이들은 Brennan과 Resnick(2012)이 제안한 CT 영역 중 컴퓨팅 개념만을 측정하기 때문에 컴퓨팅 수행과 컴퓨팅 관점의 영역을 간과하고 있고 결과에 대한 총괄적인 평가만을 수행한다는 한계점이 존재한다. 아울러 이들 평가 방법 모두 Scratch, AppInventor 등의 블록 기반의 프로그래밍 평가이기 때문에 텍스트 기반의 프로그래밍 평가에 대한 연구가 필요한 시점이다.

따라서 본 연구에서는 기존 연구의 한계점을 보완하여 형성적인 관점에서 CT을 측정할 수 있는 평가 방법에 대해 탐색하고, 텍스트 기반의 프로그래밍 평가 방법을 탐색해보고자 한다.

2. 관련 선행연구

2.1 컴퓨팅 사고력(CT)의 중요성과 평가 영역

CT는 전통적으로 강조되었던 아동의 분석적 능력인 읽기, 쓰기, 산술에 추가되어야 할 능력으로 거론될 만큼 필수적인 능력으로 강조되고 있다[51]. CT는 "문제와 해결책을 공식화하여 정보처리를 효율적으로 처리할 수 있는 형태로 해결책을 표현하는 사고 프로세스"로 정의되며[52], 해결책은 컴퓨팅 단계와 알고리즘으로 표현할 수 있다[1]. 한국과학창의재단(2014)에서는 CT를 "컴퓨팅 시스템의 역량을 활용하면서 해결하고자 하는 문제를 효과적이고 효율적으로 해결할 수 있는 절차적 사고 능력"으로 정의하였고[27], 최숙영(2016)은 CT의 요소를 자료수집, 자료분석, 자료표현, 문제분해, 추상화, 알고리즘 및 절차, 자동화(코딩), 디버깅, 시뮬레이션, 평가, 일반화로 구분하였는데 이는 일반적인 문제해결과정에서 고려되어야 할 사항임을 주장함으로써 CT가 곧 실세계의 문제해결과정과 관련성이 높음을 직관적으로 보여준다[9]. 또한. 프로그래밍 학습을 통해 CT가 향상될 수 있다고 연구 결과가 발표되고 있다[24][35][43][44].

CT 평가를 위해 프로젝트 기반 포트폴리오 분석, 문서기반 인터뷰, 디자인 시나리오 개발[7] 등의 평가 방법이 제시되었으며, CS(Computer Science) 실습에서 필요한 개념에 대한 지식을 평가하는 것보다 문제를 해결하면서 활용된 기술 및 패턴에 주목할 것을 당부하였다[5].

이에 따라 다수의 학자들에 의해 CT 평가영역에 대해 논의되기 시작하였으며, 추상화, 일반화, 알고리즘, 모듈화, 분해, 데이터(수집, 분석, 표현), 평형화, 시뮬레이션 등이 CS를 통해 학습 가능한 CT 요소로 일반화되었다[3][4][7].

아울러, Brennan과 Resnick(2012)은 교육용 프로그래밍 언어(EFL)인 스크래치 맥락에서 CT를 세 가지 영역, 개념, 수행, 관점 세 가지로 크게 나누고, 컴퓨팅 개념을 다시 시퀀스, 반복, 평형화, 이벤트 조건, 연산자, 데이터로, 컴퓨팅 수행을 점진적이고 반복적인 테스트와

디버깅, 재사용과 리믹스, 추상화 및 모듈화로, 컴퓨팅 관점을 표현하기, 연결하기, 질문하기로 분류하였다[7]. 이러한 Brennan과 Resnick(2012)의 CT 프레임워크를 기반으로 스크래치 기반 자동평가 시스템을 개발한 Moreno-León 등(2015)과 Dasgupta 등(2016)은 컴퓨팅 개념 요소와 평가규준을 마련하였다[13][35].

2.2 프로그래밍 평가 설계 및 예시

먼저 설계 시나리오를 활용한 CT 평가 설계를 알아보고자 한다. Zhong 등(2016)에 의하면 설계 시나리오를 활용했을 때, Brennan과 Resnick(2012)이 제안한 CT 프레임워크 중 컴퓨팅 개념과 컴퓨팅 수행을 측정 가능하며, 설계 시나리오는 하나의 정답 코드를 완성하는 형태가 아니라 목표에 부합하는 다양한 코드를 완성 시키도록 하는 평가 방식을 취하고 있는데, 이는 문제기반학습 환경에 적합하다는 장점이 있다[55]. 구체적으로, 버그를 고정하거나 학습할 과제를 추가시킴으로써 리믹스가 가능하게 하는 등의 복잡성을 향상시킨 프로젝트를 수행하도록 하는 방법이 있다. 이를 위해 덜 완성된 프로젝트를 제시하거나 올바르지 않은 코드를 과제로 제시할 수 있다[55].

과제 설계 방식에 따라 CT의 어떤 영역을 측정하는지 결정될 수 있다. 닫힌 질문을 활용한 과제는 CT 개념만을 측정하는데 적합하며, 정의된 결과와 정의되지 않은 과정으로 설계된 semi-open task는 CT 개념과 수행을 측정하는데 적합하다. 또한, 열린 결과와 열린 과정으로 설계된 열린 질문(open task)은 semi-finished project에 학생들이 자유롭게 기능 및 시나리오를 추가하게 함으로써 CT의 전 영역을 측정하는데 적합하다. 이 외에도 학습 과정에 대한 정보를 수집하기 위해 인터뷰를 시행할 수 있으며, open task를 활용했을 때에는 성찰 보고서를 활용하여 창의적 설계를 확인할 수 있다[55].

<Table 1>은 텍스트 기반 프로그래밍 학습을 평가하기 위해 설계된 연구를 정리한 것이다.

이들 연구에서 프로그래밍 평가에 활용된 방법은 크게 세 가지로 나눠볼 수 있다. 첫 번째로, 기능적인 테스트를 수행하는 방법은 '테스트 i가 수행되었다(또는, 되지 않았다).'의 일반적인 패턴의 컴파일 결과와 CPU 시간의 효율성 등이 피드백되는데[14][19][42], 어느 부분에서 오류가 발생하였으며 어떻게 수정할 수 있는지에 대한 정보를 포함하지 않기 때문에 초보자들에게 유용하지 않다는 단점이 있다. 두 번째로 자동화된 소프트웨어 확인(Automated software verification)은 주로 버그를 찾는 방식으로[50] 때때로 실제 에러를 발견하지 못하거나 잘못된 에러를 보고하는 문제를 발생시키며, 심지어 같은 변수나 템플릿을 써야 하므로 CT 함양을 저해할 수 있다는 문제점이 보고되고 있다[32]. 세 번째는 참조 솔루션과 비교하는 방법이다[20]. 이는 초창기 알고리즘 자동평가 시스템인 'Online Judge System'의 평가 방식으로 문제와 해당 솔루션을 함께 요구한다. 그러나 학습자가 작성한 코드는 예측 불가능하므로 이와 같은 평가를 실행하기 위해서는 한 가지 문제에 대한 가능한 한 많은 솔루션을 요구한다는 제한점이 있다[54]. 이와 같은 문제를 해결하고 개인화된 피드백을 제공하기 위해 Truong, Roe와 Bancroft(2004)은 변형 규칙(transformation rule)

<Table 1> Assessment Design for Text-based Programming Learning

	Programming Language	Assessment Method	Analysis Method
Isaacson & Scott(1989)	C, C++, Perl, Pascal, Java	Program running, Program style, Practicality	Dynamic analysis
Reek(1989)	UNIX	Program running, Program style, Practicality	Dynamic analysis
Jackson & Usher(1989)	Ada programs	Compare with predefined data sets, Efficiency of CPU time	Dynamic analysis
Kurnia et al(2001)	C++, Java	Online judge system(algorithm evaluation)	Static analysis
Nygård(2016)	Python	Analysis of the number of task performed in each learning area (ex: loop)	Static analysis
Jamil(2017)	Java, C++, Python	The program dependence graph(PDG), Concept depence graph(CDGs)	Dynamic/Static analysis
Marin(2017)	Java	The program dependence graph(PDG), Concept depence graph(CDGs)	Dynamic/Static analysis

을 적용하는 방법을 선택하였고[48], Piech 등(2015)은 변수를 고정하는 방법을 고안하여 활용하였다[41]. 그러나 이들은 제한적인 규칙과 변수를 사용하여야 한다는 점에서 문제해결을 저해한다는 비판을 받았다[32].

위와 같은 방법들의 문제를 해결하기 위해 Singh 등(2013)은 학생들이 범할 수 있는 오류를 정의하고 프로그램 스케치와 제출물을 비교하는 방법을 고안하였고[46], Nygård(2016)은 특정 과제를 제시하는 방법을 선택하였다[38]. 그러나 이와 같은 평가 방법은 과제의존도가 높기 때문에 과제에 따라 평가 규칙이 수정되어야 한다는 제한점을 가지고 있다. 또한, Zuleger 등(2016)은 서로 다른 입력으로 계산된 변수 추적을 활용하여 올바르게 코딩된 과제를 군집화하였고, 각 군집을 평가하는 참조 솔루션으로 각각 채택하는 방식을 선택하였다[56]. 그러나 이 또한 가능한 모든 오류를 발견하기 위한 솔루션이 필요하고, 무한루프와 같은 기능에서는 활용 불가능하다는 단점을 가지고 있다. 따라서 프로그래밍 언어의 평가는 참조 솔루션과 같은 코드를 사용하였는지 코드의 일치도를 평가하는 방법보다는 기능적이고(functional) 의미론적인(semantic) 일치를 평가하는 것이 바람직하다[17][21][32]. 예를 들어, counter loop 구문에 대한 학습을 평가할 때, 이 구문이 들어간 코드의 일치도만을 평가하는 것이 아니라, 동등한 역할을 수행하는 코드, 즉, counter loop의 기능을 수행하는 for 구문과 while 구문 등을 활용한 동등한 코드까지 확인하는 방안을 제시하고 기능의 동등성에 중점을 두는 것이다[21].

양적 측정에 있어서 가장 용이한 방법인 다중 선택 문제(Multiple choice questions)는 프로그래밍 평가에서도 다수 사용되고 있다[31][40][45]. Shuhaida 등(2010)은 형성적 평가의 관점에서 다중 선택 문제의 실효성을 확인하기 위한 연구를 수행하기 위해 문제를 선택하고 이들 문제를 통해 평가 가능한 평가영역을 설정하였다[45]. BRACElet Problem Set[31], Parson's puzzle question[40]의 문제와 연구자가 새롭게 만든 문제에 내포된 의미론적 지식, 문제해결능력, 문제의 난이도, 구문지식을 측정하였다. 이들을 활용하여 학생의 학습과정을 평가한다는 의미에서 형성평가의 성격을 띠고 있다.

프로그래밍 학습을 형성적으로 평가하기 위한 노력은 Marin 등(2017)의 연구에서 잘 드러나고 있다. Marin 등(2017)에 의하면 학생들 간의 동료 피드백만으로는 불완전한 정보를 제공할 위험이 존재한다[32]. 따라서 이들은 개별화된 피드백을 제공함으로써 프로그래밍 초보자인 학생들에게 코드의 옳고 그름을 넘어 개선사항에 대한 충분한 안내를 제공하고자 하였고, 이들은 초보자들에게 보통 제시되지 않는 기능 테스트, 코드에서 버그를 찾기 위한 소프트웨어 검증, 다수의 참조 솔루션에 활용한 비교 등을 피드백으로 구성하여 시스템을 설계하였다. 또한, 이들은 다른 평가 시스템과는 달리 학생들이 과제를 다루는 본래의 의도와 과제의 의미를 이해하는 것을 기반으로 하고 있으므로 한 과제에서의 학생들의 패턴을 다른 과제의 평가에서 활용할 수 있다고 강조하고 있다.

2.3 프로그래밍 평가를 위한 코드 분석 방법

프로그래밍 평가를 위한 코드분석 방법은 크게 동적분석(dynamic code analysis)과 정적분석(static code analysis)으로 나뉜다. 동적분석 방법은 코드를 실행하고, 생성된 결과물을 이미 저장된 대조군에 비교하는 방법이기 때문에 프로그램의 정확성을 평가하여 학생을 역량을 평가하는 방법으로 활용된다. 반면, 정적분석은 프로그램 실행 절차 없이 소스 코드만을 검사하는 방법으로, 코딩스타일, 소프트웨어 메트릭, 프로그래밍 오류, 디자인, 특수기능을 분석하는데 활용된다. 비정형화된 과제는 일반적으로 정적분석 방법으로 프로그래밍 구조를 파악한다[2]. 본 연구에서는 정적분석을 활용한 평가툴을 분석하고자 하였다.

먼저, Marmoset 시스템[47]과 Web-CAT[14] 시스템은 학생들이 작성한 코드의 유닛 단위를 테스트하여 코드 평가를 수행한다. 이는 정적 분석 도구로부터 코드범위 분석 및 피드백을 통해 학생들이 받는 평가를 보완했다는 의미가 있다.

Truong 등(2004)은 JAVA 프로그래밍을 평가하기 위해 정적분석 도구를 개발하였다. 학생들의 부족한 프로그래밍 수행이나 일반적인 논리적 에러로부터 순환복잡도(cyclomatic complexity), 불필요한 논리적 표현(redundant logic expression), 구조적 유사성 등의 평가요소를 수립하였고, 학생들이 제출한 코드를 분석하여 이들 평가요소에 대한 피드백을 제공하였다. 또한, 대조군과 비교하여 모델에는 각 코드가 몇 번 사용되었고 본인의 코드에는 몇 번 사용되었는지 한눈에 비교하기

쉽게 설계하였다[48].

Curator system는 버지니아 폴리테크닉 주립대학교에서 개발한 코드 평가 시스템으로, 셸 스크립트(예: .bat, .exe, .sh)를 기반으로 컴파일 및 실행을 처리하는 방식으로 평가된다[14]. 과제는 교수자가 부여하는 방식으로 활용될 수 있는데, 컴파일 및 실행의 평가가 자동으로 이루어지기 때문에 설계자는 설계, 스타일, 기록과 같은 영역을 평가하는데 많은 시간과 노력을 기울일 수 있다. 이 평가 방식에서는 피드백을 부여하고 높은 점수를 얻을 수 있도록 수정기회를 제공하기 때문에 학생들이 과제에 대한 정보를 얻기 위해 조기에 과제를 제출하는 등의 효과를 확인할 수 있었다.

최근 교육용 프로그래밍 언어인 Scratch, AppInventor 등의 블록 기반 프로그래밍 언어가 개발되면서 이를 활용하여 작성한 코드를 테스트하기 위해 정적분석 방법을 활용한 평가 시스템이 개발되기도 하였다. Scrape[53]는 스크래치 파일에서 패턴을 이해하는데 도움이 되는 시각화 도구로 'loop를 얼마나 사용하는지?', '각 프로그램에 loop가 얼마만큼 나타나는지?', '프로그램에서 사용하는 중첩 수준은 어떠한지?'를 살펴보는데 유용한 도구이다. 나아가 Hairball[6]은 스크래치 프로그램에 포함된 개념이 의미상 부정확한지, 불완전한지에 대해 단계적인 피드백을 제공한다. Hairball은 Scrape와 마찬가지로 과제의 필수 구성요소를 사용했는지 확인하는데 도움을 줄 수 있다[6].

위와 같이 초기 평가 시스템이 프로그램 테스팅에 중점을 두었다면, 최근에는 CT을 평가하기 위한 평가 툴의 중요성이 대두되면서 다음과 같은 평가 툴이 개발되었다. 예를 들어, Dr. Scratch는 Hairball의 플러그인을 사용하고 있지만, Hairball에서 제공하지 않는 CT 점수 및 CT를 신장시키는데 필요한 피드백을 제공하고 있다. 뿐만 아니라 프로그래밍에 사용되었지만 중요하지 않은 스프라이트의 이름, 코드 반복, 실행되지 않은 코드 등을 보고하고 기술의 향상을 꾀하고 있다[35].

반면, CodeMaster는 K-12 교육과정에서 CT를 학습하는데 중점을 두고 있는 문제 기반 상황에서의 프로그래밍 활동을 평가하기 위한 도구이다[49]. App Inventor와 SNAP!을 활용한 프로젝트를 기반으로 CT를 평가하기 위해 Dr.Scratch와 유사한 영역을 자동으로 평가하고 등급을 부여하는 무료 웹 기반 시스템으로, 정답이 없는 비구조화된 복잡한 프로그래밍 활동에 중점을 두고 개발되었다. 이것은 학생들이 학습 과정을 통해 수행에 대한 즉각적인 피드백을 얻거나 교사들이 프로젝트의 등급을 매기거나 평가하기 위해 사용된다[49].

선행연구를 통해 얻게 된 평가 설계에 대한 시사점은 다음과 같다.

첫째, 기존에 개발된 프로그래밍 평가 툴이 블록 기반 프로그래밍 언어에 한정되어 있다는 점을 보완할 필요가 있다. 블록 기반 프로그래밍 언어의 학습에서 텍스트 기반 프로그래밍 언어의 학습으로의 전이를 고려하고 텍스트 기반 프로그래밍 언어의 학습을 통해 고차원적 컴퓨팅 사고로의 향상이 가능한 점을 고려하여야 한다[34]. 따라서 본 연구에서는 초보자들이 비교적 접근하기 쉬운 텍스트 기반 프로그래밍 언어인 파이선을 평가하기 위한 프로세스를 설계할 것이다.

둘째, 블록 기반 프로그래밍 언어 평가 툴이 Brennan과 Resnick(2012)이 제시한 CT 영역 중 CT 개념만을 평가한다는 한계점이 지적됨에 따라 CT 수행, CT 관점이 포함하고 있는 재사용 및 리믹스, 표현, 알고리즘, 추상화, 협업 등을 평가할 수 있는 방법에 대한 논의가 필요하다.

셋째, 기존 텍스트 기반 프로그래밍 언어의 평가가 코드의 일치도를 평가하고 있으나, 이는 예측 가능한 모든 솔루션을 평가 시스템 내에 포함하고 있어야 한다는 단점이 있다. 따라서 코드의 일치도를 평가하는 방법을 지양하고 기능적 동일성을 평가하는 정적분석 방법을 평가의 방법으로 고려할 것이다.

넷째, 닫힌 질문(Closed task)을 활용한 과제 제시는 학생의 CT 개념만을 측정하는데 그칠 수 있으므로, 열린 질문(Open task)를 활용한 과제 제시를 제안한다. 단, 학생들이 특수한 문제 상황에 대한 구체적인 사고를 유도하기 위해 실제적인 문제 상황을 제시하고자 한다.

3. 연구목적 및 연구방법

초 · 중 · 고등학교뿐만 아니라 대학교육에서도 SW 중점 대학을 선정하여 비전공자들에게 SW 교육을 제공하는 등 SW 교육이 영역을 확장하고 있으며 SW 교육을 위한 가이드라인에도 평가의 중요성이 강조되고 있다. 그럼에도 불구하고 여전히 평가를 위한 프레임워크 및 평가에 관련된 연구는 블록 기반 프로그래밍 언어에 한

정되고 있다. 텍스트 프로그래밍 언어는 블록 프로그래밍 언어에 비교하여 다른 텍스트 프로그래밍 언어로의 전이 가능성이 높고, 컴퓨팅 사고력 향상에 더욱 도움을 준다[34]. 따라서 이후의 연구에서는 많은 상용 응용 프로그램에서 스크립트 언어로 활용되는 프로그래밍 언어 학습 지원에 주목하여 본 연구에서는 비교적 초보자들이 다루기 쉬운 문법으로 프로그래밍 가능한 파이선 학습에 초점을 두고 평가 방안에 대해 논의하고자 한다.

이를 위해 본 연구에서는 사례분석과 선행연구 분석을 통한 질적연구 방법으로 다음과 같은 과정을 수행하였다. 첫째, 초보자들이 주로 학습하는 파이선 프로그래밍 학습 영역을 탐색하였다. 파이선 프로그래밍에 관련된 선행연구 중 초보자인 학생들을 위한 커리큘럼을 살펴보고 이를 종합하여 본 연구에서 평가영역의 바탕이 되는 학습 영역을 규명하고자 한다. 둘째, 파이선 프로그래밍과 관련된 선행연구를 분석하여 파이선을 통해 함양할 수 있는 CT 영역을 확인하고자 하였다. 마지막으로 초보자들의 파이선 학습요소와 CT 영역을 종합하여 평가요소를 규정하고 문헌분석을 통해 얻게 된 시사점을 적용하여 평가 방법과 그 과정을 설계하고자 하였다.

이후 컴퓨터공학 및 컴퓨터교육 전문가 2인에게 2차례 검토를 받았다. 1차 검토는 평가 프로세스 수립을 위해 필요한 파이선 학습영역과 컴퓨팅 사고력 영역에 관한 것으로, 프로세스 구성요소에 관련된 의견을 수렴하기 위해 수행되었다. 1차 검토를 통해 얻게 된 의견을 프로세스에 설계에 적용한 후, 2차 검토를 통해 향후 개선에 대한 제안사항을 얻고자 하였다.

4. 파이선 프로그래밍 평가를 위한 프로세스 설계

4.1 초보자를 위한 파이선 프로그래밍 학습영역 분석

파이선 프로그래밍에서 초보자들에게 적합한 학습 영역을 도출하기 위해 선행연구에서 활용된 커리큘럼을 종합하고 활용된 명령어 및 학습 영역을 확인해 보고자 한다.

<Table 2>에 나타난 바와 같이 이들 선행연구에서는 변수, 연산자, 데이터(순차), 조건문, 반복, 함수와 관련된 명령어에 대한 학습을 포함하고 있음을 확인하였다. 변수에서는 변수의 정의와 활용, 연산자에서는 사칙연산과 논리적 연산자, 데이터 타입에서는 정수형태, 실수형태, 문자열, 그리고 라이브러리를 학습하게 함을 알 수 있다. 또한, 리스트에서는 리스트의 이름과 길이(len), 리스트 요소(sort)를 기본 내용으로 익히고, if, if-else, if-elif-else, True/False와 같은 형태의 조건문과 for, range, while 등과 같은 형태의 반복문, input, print와 같은 기본 함수와 내장함수, 외장함수, turtle 라이브러리, 나아가 함수의 정의 등 함수의 기본 영역도 익히도록 구성되고 있다.

<Table 2> Python Learning Area for Novice

	Han(2018)	Kim & Han(2018)	Kang(2019)
Variable	variable	variable	variable
Operator	arithmetic logical		operator
Data type (Sequence)	int float string	int float library random dictionary string	
List & Sort	list & sort	list	list
Conditional	if if-else if-elif-else	if elif else T/F	conditional statement
Loop	range while for	for range while	loop
Function	function	built-in function external function turtle library def	turtle library function input print

이를 기반으로 본 연구에서는 이와 같은 영역이 초보자인 파이선 학습자가 반드시 학습해야 할 학습영역으로 규정하고, CT 평가영역과의 관련성을 파악하고자 하였다.

4.2 파이선 프로그래밍에서 CT 평가요소 분석

본 연구는 파이선이라는 텍스트 기반 프로그래밍 언어를 통해 CT 신장이 가능할 것이라고 가정하고 파이선 프로그래밍 학습의 평가를 위해 파이선 프로그래밍에서의 CT 요소를 연구한 선행연구를 분석하였다(<Table 3> 참조).

한국교육학술정보원에서 'SW교육 교수학습 모형 개발 연구'에서 제시한 개발중심모형(DDD)을 중심으로 파이

선 프로그래밍 학습에서 CT 요소를 분류한 이영석(2018)은 탐구(Discovery) 단계에서 패턴 인식을, 설계(Design) 단계에서 추상화와 알고리즘을, 개발(Development) 단계에서 자동화와 추론을 경험할 것이라고 하였다[30].

권정인(2019)은 컴퓨팅 사고를 문제해결과 관련된 것이라고 보고, 분석(Analysis), 설계(Design), 개발(Development)에 해당하는 CT 요소를 분류하였다[29]. 분류된 CT 요소 중 데이터 수집, 분석, 표현은 분석 단계에, 문제분해, 추상화, 알고리즘 및 절차는 설계 단계에, 자동화, 시뮬레이션, 평형화는 개발 단계에 포함되어 있다. 반면, 지식, 수행, 태도의 관점에서 CT 영역을 분류한 최숙영(2019)[10]의 분류는 지식영역에 해당하는 CT 요소가 Brennan과 Resnick(2012) CT 개념에 해당하는 순차, 조건, 반복, 연산자, 함수가 지식 영역에 해당하며, 추상화, 분해, 알고리즘이 수행 영역에 해당한다.

세 연구 모두 Wing(2008)이 강조한 추상화와 자동화를 포함하고 있으며, 그 외에도 알고리즘적 사고를 모두 포함하고 있다.

특히, 최숙영(2019)의 연구에서는 자신감, 인내심, 의사소통, 협업, 창의성이 태도 영역에 포함되어 있어 정의적 태도를 중시함을 알 수 있고, 즉, Brennan과 Resnick(2012)의 분류 중 CT 관점을 중시하는 경향을 보였다.

4.3 1차 전문가 검토

4.1과 4.2에서 수행한 내용에 대해 1차 전문가 검토를 실시하였다. 1차 검토 결과 주요 제안사항 및 적용사항은 다음과 같다.

첫째, CT의 양적 측정을 위해 도출한 CT 요소(Brennan과 Resnick(2012)의 CT 개념에 해당하는 요소와 구문)를 어떤 표현을 사용해는지에 따라 기초/심화로 절대적인 점수를 부여하는 것은 바람직 하지 않다. 이와 같은 의견에 따라 본 연구는 세련되지 않은 코드라도 적절한 단계로 논리를 충분히 표현하고 있는지에 초점을 맞추고자 한다.

둘째, 위에서 제시된 CT 영역에 대한 평가 및 피드백 뿐만 아니라 오류메시지에 대한 피드백이 필수적이다. 프로그래밍 학습에 있어서 오류를 찾아내고 수정하는 능력인 디버깅은 프로그래밍 능력을 향상시키는 필수적인 요소인데, 초보 학습자들의 경우 컴파일 후 오류메시지를 접했을 때 포기하려는 경향이 크므로, 이를 평가 과정에서 고려하는 것이 바람직하다. 본 연구는 이와 같은 의견을 수렴하여 피드백 설계에 있어서 CT 향상에 필요한 피드백 뿐만 아니라 에러메시지에 대한 피드백을 포함하고자 하였다.

셋째, 알고리즘 능력 향상을 위해 순서도와 의사코드를 활용할 필요가 있다. 파이선에서 사용하는 코드 규칙을 이해하는 것에 앞서, 내가 사용하는 언어와 간략한 도형을 통해 아이디어를 논리적으로 표현하는 활동은 학생들에게 성찰의 기회를 제공할 것이다.

넷째, 커뮤니티나 포럼과 같은 형태의 상호작용 도구를 지원함에 따라 CT 영역 중 의사소통이나 협업 향상의 수단으로 활용될 수 있다. 본 연구는 상호작용 도구를 제공하고, 형성적 평가 방법으로써 동료평가를 위한 수단으로 활용하고자 하였다.

<Table 3> Computational Thinking through Python

Lee(2018)		Kwon(2019)		Choi(2019)	
Discovery	pattern	Analysis	data collection	Knowledge	Sequence
			analysis		Conditional
			representation		Loop
					Operator
					Function
Design	abstraction	Design	problem decomposition	Practice	Abstraction
			abstraction		Decomposition
	algorithm		algorithms & procedures		Algorithms
Development	automation	Development	automation	Attitude	Confidence
			simulation		Patience
	inference		parallelisms		Communication
					Collaboration
					Creativity

4.4 파이선 프로그래밍 평가 프로세스 설계

선행연구 분석을 통해 초보 학습자들이 파이선 프로그래밍에서 변수, 연산자, 데이터(순차), 조건문, 반복, 함수와 관련된 구문을 학습하는 것을 확인하였다[18][22][25]. 초보자들은 주로 파이선 프로그래밍에서 구문을 학습하고, 학습한 내용을 적용하여 코딩을 수행한다. 프로그래밍에서 구문은 문제를 해결하기 위한 절차를 추상적으로 표현하는데 필수적으로, Brennan과 Resnick(2012)은 이 영역을 CT 영역 중 CT 개념으로 분류하였다. 즉, CT 평가에 있어서 구문에 대한 평가가 요구되는데, 활용된 구문을 분석하고 이를 기반으로 학습자에게 피드백으로 제공함으로써 학생 자신의 강 · 약점과 코딩 스타일을 파악하게 할 수 있게 한다. 또한, 파이선 초보 학습자들이 오류에 대한 피드백을 얻게 하고 코딩의 효율성 측면에서 개선에 필요한 정보를 획득하도록 하는 것이 평가의 목적이 될 수 있다.

따라서 본 연구에서는 학생들의 코드에서 활용된 구문 중 변수, 연산자, 데이터, 조건문, 반복, 함수 각 영역에 해당하는 파이선 구문의 사용을 평가하고, 이와 함께 오류를 분석하여 개별적 피드백을 제공하는 방식의 평가 프로세스를 설계하였다.

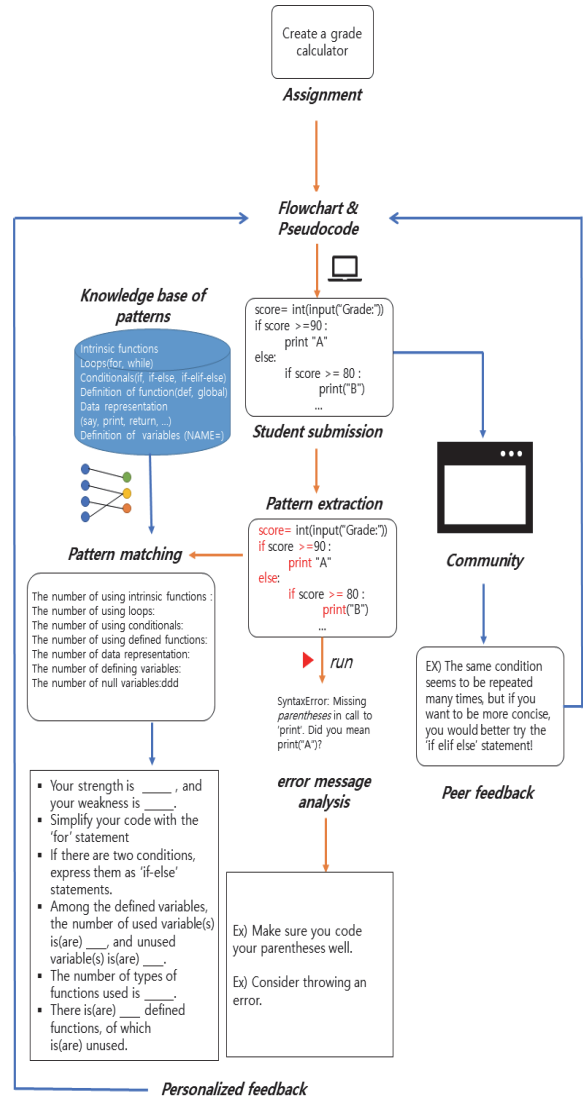
구체적인 설계안은 다음과 같다((Fig. 1) 참조).

교수자가 제시한 개념 또는 주제에 대해 학생들이 제시한 답안에서 CT 요소를 포함하는 코드를 추출하고, 이를 시스템에 저장되어있는 패턴과 매치시키는 과정을 거쳐 각 횟수를 측정함으로써 학생의 코드에서 가장 많이 활용된 CT 영역과 약세를 보이는 CT 영역을 피드백으로 제공한다. 이와 같은 방법은 기존의 코드 평가가 솔루션 코드와 학생이 제출한 코드를 비교하여 문법적 오류만을 판단하는 방식[8][15][20]을 탈피한 것으로, 코드 작성의 다양성을 인정한다는 점에서 기존 동적 측정 방법과는 차별된다. 또한, 다른 방법으로 코딩하거나 코드를 간략하게 나타내는 방법을 제시함으로써 다양한 코딩방법을 익히도록 한다.

코드를 실행했을 때 나타나는 오류메시지를 그대로 학생들에게 제공하는 것은 초보자들의 인지적 부담을 가중시킬 수 있으므로[14], 파이선으로 실행시켰을 때 출력되는 에러메시지를 에러 유형에 따라 분류된 기준을 기반으로 분석하여 초보자들에게 에러 개선에 대한 유용한 정보를 제공하는 평가를 설계하고자 하였다.

이와 같이 개별화된 피드백을 제공하는 것은 프로그래밍 초보자인 학생이 학습 개선에 대한 통찰을 얻도록 하므로[32], 자신의 학습결과와 피드백 내용을 프로그래밍 학습에 적용하게 하고, 형성적인 평가가 학습 과정에 나타날 수 있다.

종합하면, 본 연구에서는 코드의 문법과 기능적 오류를 함께 분석하여 개별화된 피드백을 제공하여 학습 개선을 이끄는 평가의 설계를 제안한다.



(Fig. 1) Assessment process for python programming

평가를 통해 제공하고자 하는 피드백은 동등한 기능을 수행하는 코드작성에 관한 힌트(예: (한 구문에 조건을 여러 번 제시할 때 if만 사용한 경우) if-elif-else 구문을 사용하여 같은 기능을 수행하는 코드를 작성해보세요.), 각 CT 영역의 점수와 그에 대한 피드백(예: CT 영역 중 강점은 데이터표현 약점은 반복입니다.), 죽은 코드 등에 대한 피드백(예: 정의된 변수 중 활용된 변수는 1개 활용하지 않은 변수는 0개입니다), 오류메시지 분석(예: 괄호가 잘 코딩되었는지 확인하세요. 에러를 예외처리 하는 방법을 고려해보세요. 등)이다. 이와 같은 설계는 현재의 CT 수준을 측정할 뿐만 아니라 자신이 문제해결을 위해 어떤 구성요소를 주로 사용하고 있는지 이해하고 이를 스스로 분석하게 함으로써 CAS(2015)[12]가 강조한 CT 구성요소인 '분해' 능력과 김재경(2017)[23]이 제시한 컴퓨팅 사고의 추상적 개념 중 '문제 및 산출물 분석'을 함양하고자 한 것이며, 죽은 코드를 확인하게 함으로써 불필요한 사항을 줄이고 문제를 더 이해하기 쉽게 만드는 '추상화' 능력을 함양하기 위함이다. 또한, 오류메시지에 대한 분석은 Brennan과 Resnick(2012)의 CT 분류 중 컴퓨팅 수행(computational practices) 중 오류를 찾아 수정하는 디버깅 과정에서 초보자들을 지원하기 위한 방안이다.

또한, 추상화와 알고리즘적 사고능력의 함양을 위한 방법으로 '순서도(flow-chart)'나 '의사코드(pseudocode)'를 활용하고자 한다((Fig. 1) 참조). Kitchin(2017)은 통합적인 문제해결을 위해 알고리즘적 사고능력의 중요성을 인식하고 의사코드와 순서도의 활용을 주장하였다[26]. '순서도'는 본격적인 프로그래밍 전 자신의 아이디어를 도식화하는 방법으로, 코드를 작성하는 순서와 매우 밀접한 관련이 있고, 수학 등의 교과에서도 학생들의 알고리즘적 사고를 기르기 위해 활용되는 방법이기도 하다. 또한, '의사코드'는 코드를 작성하는 사람이 이해하기 쉽게 자신의 언어로 코드를 작성하는 것으로, 학생들이 이러한 과정을 통해 논리적 사고력을 배양할 수 있다. 뿐만 아니라 본 연구에서는 코드 작성 후 자신의 아이디어를 가시적으로 나타낸 순서도 또는 의사코드 내용의 반영 여부에 대해 자기평가가 가능하도록 프로세스를 설계하였다.

마지막으로 아이디어의 공유와 코드의 확장, 리믹스 및 재사용을 위한 커뮤니티 공간을 확보하고 이 과정을 학습 과정이자 평가요소로 평가 프로세스에 포함하였다((Fig. 1) 참조). 커뮤니티에서 코드에 대한 리뷰, 프로그램 수정, 비판적이고 건설적인 피드백을 주고 받는 것은 형성적 평가 방법 중 동료평가의 주된 방법이다[11]. 아울러 이는 컴퓨팅 사고력의 요소 중 의사소통 및 협업을 위한 강력한 도구가 될 수 있다[10].

4.5 2차 전문가 검토

설계된 프로세스를 기반으로 2차 전문가 검토를 의뢰하였고, 검토 결과 얻게 된 제안사항은 다음과 같다.

첫째, 평가 결과에 따른 피드백을 유형화하고, 구체적인 예시를 제공할 필요가 있다.

둘째, 본 연구에서 설계한 프로세스를 적용하여 평가 시스템을 개발할 필요가 있다. 또한, 학습자가 학습역량 변화 추이를 확인할 수 있도록 개발된 평가 시스템을 기대한다.

셋째, 설계된 평가 프로세스를 통해서 실제로 학생들에게 유용한 정보를 제공할 수 있는지 테스트가 필요하다.

5. 결론

본 연구는 프로그래밍 학습을 통한 CT 함양의 중요성을 확인하고, 기존 평가 방식을 분석하여 텍스트 기반 프로그래밍 언어인 파이선을 학습하는 학생들의 인지부하를 줄여줌과 동시에 CT 계발에 도움이 되는 평가 프로세스를 설계하고자 수행되었다.

선행연구를 통해 CT 평가요소와 파이선을 활용한 초보자의 프로그래밍 학습 요소에 대해 탐색한 결과, 코드 분석을 위한 CT 평가요소로 변수, 연산자, 데이터(순차), 조건문, 반복, 함수 등을 선정하였다. 아울러, 순서도 · 의사코드를 활용한 아이디어 표현을 자신이 작성한 코딩에 대한 반성적 사고를 가능케 하는 수단으로 활용하고, 커뮤니티를 활용한 의사소통을 통해 동료평가 및 동료 피드백이 가능하도록 설계하였다.

더불어 파이선을 통한 학습을 평가하기 위한 평가 프로세스를 설계하기 위해서는 다음과 같은 내용이 고려되었다.

첫째, 파이선을 통해 한 가지 기능을 수행하는 코드를 다양한 방법으로 프로그래밍할 수 있고, 어떤 방법이

더 우월한 방법인지 판단하는 것은 적절하지 않다. 따라서 어떤 코드를 사용했는지에 따라 점수를 부여하는 방식이 아닌, 사용한 코드가 적절하게 구현되는지와 함께 동등한 기능을 수행하는 코드가 존재함을 인지시켜주는 피드백을 제공해야 한다는 것이다.

둘째, 파이선에서 제공하는 에러메시지는 프로그래밍 초보자인 학생들에게 인지 부하를 일으킬 가능성이 크다. 따라서 각 오류코드와 해당 오류가 나타나는 오류 유형을 정교화하여 오류 수정에 대한 지원을 제공해야 한다는 것이다.

셋째, 프로그래밍을 수행하는 것은 논리적 사고력과 알고리즘적 사고를 요구하기 때문에 이를 지원하기 위한 학습 지원이 필요하다. 따라서 본 연구에서는 코딩을 수행하기 전 아이디어를 논리적으로 표현해 볼 수 있도록 순서도와 의사코드를 활용하는 기회를 제공하고 이를 학습 및 평가 과정에 포함하는 것을 제안하였다.

넷째, 동료와의 아이디어 공유, 코드에 대한 질의응답, 코드의 재사용과 리믹싱은 다수의 연구에서 중요성을 강조하고 있는 협업의 방법으로써, 커뮤니티를 조성하여 의사소통을 가능하게 하고, 동료 피드백을 제공하는 것을 평가의 프로세스로 포함하는 것을 제안하였다.

단, 본 연구는 설계를 위한 연구로, 개발단계까지는 수행하지 않았다는 제한점이 있다. 추후 연구에서 본 연구를 기반으로 한 개발 절차를 수행하고 실제 프로그래밍을 배우는 학생들에게 적용하여 프로그래밍 학습에서 이를 활용함에 따라 CT 향상이 실제로 이루어지는지 확인해 볼 필요가 있다.

참고문헌

[1] Aho, A. V. (2012). Computation and computational thinking. *The Computer Journal, 55*(7), 832-835.

[2] Ala-Mutka, K. M. (2005). A survey of automated assessment approaches for programming assignments. *Computer Science Education, 15*(2), 83-102.

[3] Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems, 75*, 661-670.

[4] Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community?. *Acm Inroads, 2*(1), 48-54.

[5] Bienkowski, M., Snow, E., Rutstein, D., & Grover, S. (2015). *Assessment design patterns for computational thinking practices in secondary computer science: A first look*. SRI International.

[6] Boe, B., Hill, C., Len, M., Dreschler, G., Conrad, P., & Franklin, D. (2013, March). Hairball: Lint-inspired static analysis of scratch projects. *In Proceeding of the 44th ACM technical symposium on Computer science education* (pp. 215-220).

[7] Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. *In Proceedings of the 2012 annual meeting of the American educational research association*. (Vol. 1, p. 1-25). Vancouver, Canada.

[8] Cheang, B., Kurnia, A., Lim, A., & Oon, W. C. (2003). On automated grading of programming assignments in an academic institution. *Computers & Education, 41*(2), 121-131.

[9] Choi, S. (2016). A Study on Teaching-learning for Enhancing Computational Thinking Skill in terms of Problem Solving. *The Journal of Korean association of computer education, 19*(2), 53-62.

[10] Choi, S. (2019). Review of Domestic Literature Based on System Mapping for Computational Thinking Assessment. *The Journal of Korean association of computer education, 22*(6), 19-33.

[11] Computing At School [CAS] (2013). *Computing in the National Curriculum : A Guide for Primary Teachers*. Retrieved from http://www.computingatschool.org.uk/data/uploads/CASPrimaryComputing.pdf

[12] Computing At School [CAS] (2015). *Computational Thinking : A Guide for teachers*. Retrieved from http://community.computingatschool.org.uk/resources/2324

[13] Dasgupta, S., Hale, W., Monroy-Hernández, A., & Hill, B. M. (2016, February). Remixing as a pathway to computational thinking. *In Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing* (pp. 1438-1449).

[14] Edwards, S. H., & Perez-Quinones, M. A. (2008, June). Web-CAT: automatically grading programming assignments. *In Proceedings of the 13th annual conference on Innovation and technology in computer science education* (pp. 328-328).

[15] Foxley, E., Tsintsifas, A., Higgins, C. A., & Symeonidis, P. (1999). Ceilidh, a system for the automatic evaluation of students programming work. *Proceedings of CBLISS*, 99.

[16] Gelperin, D., & Hetzel, B. (1988). The growth of software testing. *Communications of the ACM, 31*(6), 687-695.

[17] Gouda, K., & Hassaan, M. (2016, May). CSI_GED: An efficient approach for graph edit similarity computation. *In 2016 IEEE 32nd International Conference on Data Engineering* (ICDE) (pp. 265-276).

[18] Han, Y. (2018). Analysis of Effectiveness of Programming Learning for Non-science Major Preliminary Teachers' Development of Computational Thinking. *Journal of the Korean Association of Information Education, 22*(1), 41-52.

[19] Isaacson, P. C., & Scott, T. A. (1989). Automating the execution of student programs. *ACM SIGCSE Bulletin, 21*(2), 15-22.

[20] Jackson, D., & Usher, M. (1997, March). Grading student programs using ASSYST. *In Proceedings of the twenty-eighth SIGCSE technical symposium on Computer science education* (pp. 335-339).

[21] Jamil, H. M. (2017). *Smart assessment of and tutoring for computational thinking MOOC assignments using MindReader*. arXiv preprint arXiv:1705.00959.

[22] Kang, E. (2019). Structural Software Education Model for Non-majors - Focused on Python. *Journal of Digital Contents Society, 20*(12), 2423-2432.

[23] Kim, J. (2017). Development of Rubric for Assessing Computational Thinking Concepts and Programming Ability. *The Journal of Korean Association of Computer Education, 20*(6), 27-36.

[24] Kim, S., Ham, S., & Song, K. (2015). Analytic Study on the Effectiveness of Computational Thinking based STEAM Program. *The Journal of Korean association of computer education, 18*(3), 105-114.

[25] Kim, T. & Han, S. (2018). Development of Python Education Program for Block Coding Learners. *Journnal of the Korean Association of Information Education, 22*(1), 53-60.

[26] Kitchin, R. (2017). Thinking critically about and researching algorithms. *Information, Communication & Society, 20*(1), 14-29.

[27] Korea Foundation for the Advancement of Science & Creativity (2014). *Research for Introducing Computational Thinking into Primary and Secondary Education*.

[28] Kurnia, A., Lim, A., & Cheang, B. (2001). Online judge. *Computers & Education, 36*(4), 299-315.

[29] Kwon, J. (2019). Research of Computational Thinking based on Analyzed in Each Major Learner. *The Journal of Society for e-Business Studies, 24*(4), 17-30.

[30] Lee, Y. (2018). Python-based Software Education Model for Non-Computer Majors. *Journal of the Korea Convergence Society, 9*(3) 73-78.

[31] Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., ... & Simon, B. (2004). A multi-national study of reading and tracing skills in novice programmers. *ACM SIGCSE Bulletin, 36*(4), 119-150

[32] Marin, V. J., Pereira, T., Sridharan, S., & Rivero, C. R. (2017, April). Automated personalized feedback in introductory Java programming MOOCs. *In 2017 IEEE 33rd International Conference on Data Engineering* (ICDE) (pp. 1259-1270).

[33] Ministry of Education, Korea (2015). *Software Education Instructional Guidance*.

[34] Moon, M., & Kim, K. (2008). Python programming

education for elementary school students. *Journnal of the Korean Association of Information Education, 9*(1), 33-41.

[35] Moreno-León, J., Robles, G., & Román-González, M. (2015). Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking. RED. *Revista de Educación a Distancia, 1*(46), 1-23.

[36] Myers, G. J., Sandler, C., & Badgett, T. (2011). *The art of software testing*. John Wiley & Sons.

[37] Nam, C., & Kim, J. (2019). Development of computational thinking based Coding_Projects using the ARCS model. *Journal of the Korean Association of Information Education, 23*(4), 355-362.

[38] Nygård, S. H. (2016). *Automatic self-evaluation system for novice Python developers* (Master's thesis, NTNU).

[39] Ota, G., Morimoto, Y., & Kato, H. (2016, September). Ninja code village for scratch: Function samples/function analyser and automatic assessment of computational thinking concepts. *In 2016 IEEE Symposium on Visual Languages and Human-Centric Computing* (VL/HCC) (pp. 238-239).

[40] Parsons, D., & Haden, P. (2006, January). Parson's programming puzzles: a fun and effective learning tool for first programming courses. *In Proceedings of the 8th Australasian Conference on Computing Education, 52*(pp. 157-163).

[41] Piech, C., Huang, J., Nguyen, A., Phulsuksombati, M., Sahami, M., & Guibas, L. (2015). *Learning program embeddings to propagate feedback on student code*. arXiv preprint arXiv:1505.05969.

[42] Reek, K. A. (1989). The TRY system-or-how to avoid testing student programs. *In ACM SIGCSE Bulletin, 21*(1), 112-116.

[43] Sáez-López, J. M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five schools. *Computers & Education, 97*, 129-141.

[44] Seo, Y., Yeom, M., & Kim, J. (2016). Analysis of Effect that Pair Programming Develope of Computational Thinking and Creativity in Elementary Software Education. *Journal of The Korean Assocaition of Information Ecucation. 20*(3), 219-234.

[45] Shuhidan, S., Hamilton, M., & D'Souza, D. (2010). Instructor perspectives of multiple-choice questions in summative assessment for novice programmers. *Computer Science Education, 20*(3), 229-259.

[46] Singh, R., Gulwani, S., & Solar-Lezama, A. (2013). Automated feedback generation for introductory programming assignments. *In Proceedings of the 34th ACM SIGPLAN conference on Programming language design and implementation* (pp. 15-26).

[47] Spacco, J., Hovemeyer, D., Pugh, W., Emad, F., Hollingsworth, J. K., & Padua-Perez, N. (2006, June). Experiences with marmoset: designing and using an advanced submission and testing system for programming courses. *In ACM Sigcse Bulletin, 38*(3), 13-17.

[48] Truong, N., Roe, P., & Bancroft, P. (2004, January). Static analysis of students' Java programs. *In Proceedings of the Sixth Australasian Conference on Computing Education, 30*(pp. 317-325). Australian Computer Society, Inc..

[49] Von Wangenheim, C. G., Hauck, J. C., Demetrio, M. F., Pelle, R., da Cruz Alves, N., Barbosa, H., & Azevedo, L. F. (2018). CodeMaster--Automatic Assessment and Grading of App Inventor and Snap! Programs. *Informatics in Education, 17*(1), 117-150.

[50] Vujošević-Janičić, M., Nikolić, M., Tošić, D., & Kuncak, V. (2013). Software verification and graph similarity for automated evaluation of students' assignments. *Information and Software Technology, 55*(6), 1004-1016.

[51] Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35.

[52] Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of*

the Royal Society A: Mathematical, Physical and Engineering Sciences, 366(1881), 3717-3725.

[53] Wolz, U., Hallberg, C., & Taylor, B. (2011, March). Scrape: A tool for visualizing the code of Scratch programs. *In Poster presented at the 42nd ACM Technical Symposium on Computer Science Education, Dallas, TX.*

[54] Xu, S., & San Chee, Y. (2003). Transformation-based diagnosis of student programs for programming tutoring systems. *IEEE Transactions on Software Engineering, 29*(4), 360-384.

[55] Zhong, B., Wang, Q., Chen, J., & Li, Y. (2016). An exploration of three-dimensional integrated assessment for computational thinking. *Journal of Educational Computing Research, 53*(4), 562-590.

[56] Zuleger, F., Radicek, I., & Gulwani, S. (2016). Feedback generation for performance problems in introductory programming assignments. *In Software Engineering* (pp. 49-50).

저자소개

고 은 지

2009 전북대학교 수학과(학사)

2013~2015 이화여자대학교 교육대학원 교육공학 · HRD(석사)

2015~현재 이화여자대학교 대학원 교육공학과 박사과정

관심분야 : SW 교육, 교육용 프로그래밍 언어, 학습공동체

e-mail : kej1987@nate.com

이 정 민

2001 이화여자대학교 교육공학과 (학사)

2003 이화여자대학교 교육공학과 (석사)

2009 플로리다주립대 교육심리 및 교육공학 (박사)

2009 퍼듀대학교 연구원

현재 이화여자대학교 교육공학과 부교수

관심분야: 창의적 문제해결, SW교육, 학업정서

e-mail: jeongmin@ewha.ac.kr