https://doi.org/10.14400/JDC.2017.15.7.55

디지털시대에 플립드 러닝을 활용한 학습자 맞춤형 소프트웨어 교육 방안 연구

김경미*, 김현숙**
한동대학교 글로벌리더십학부*, 대구대학교 기계공학부**

A Study on Customized Software Education method using Flipped Learning in the Digital Age

Kyungmi Kim*, Hyunsook Kim**
Global Leadership School, Handong University*
Division of Mechanical Engineering, Daegu University**

요 약 본 연구의 목적은 프로그래밍을 처음 접하는 학습자들의 어려움을 파악하여 비전공 대학생들을 위한 대학 교양기초 소프트웨어교육 운영 방안을 모색하는 데 있다. 이를 위해 다양한 전공자들로 구성된 H 대학의 파이썬 프로그래밍 수업에서 수업시간 전 온라인으로 제출한 수강생들의 질문과 수업 후 설문조사를 통하여 체감난이도와 체감이해도를 분석하였다. 비전공자들을 위한 효율적인 수업을 위해 플립드 수업으로 진행하였으며, 오프라인 수업에서는 사전질문을 활용한 학습자 맞춤형 피드백 방식 강의로 진행하였다. 분석결과 프로그래밍 수업을 처음 접하는 학습자들을 위해서는 컴퓨터 언어의 기본개념을 배우기 전에 교육과정 초반에 문제 파악을 통한 논리적인 추상화 과정을 배정하고, 코딩 실습 전에 단원마다 그에 대한 이해를 돕는 상향식(bottom-up) 문제풀이를 통한 충분한 연습이 필요하다. 또한, 학습자의 전공계열 및 수업내용과 학습자의 진행 단계를 반영한 정밀한 교육과정 설계가 선행되어야 한다.

주제어 : 프로그래밍 수업, 초보학습자의 어려움, 플립드 수업, 소프트웨어 교육과정, 프로그래밍 수업 피드백

Abstract The purpose of this study is to identify the difficulties of learners who started programming after entering college and to search an effective software education method as university liber arts for non-science major students. In order to do this, we analyzed the difficulties of learners in Python programming classes composed of students from various majors at H University through questioning and taught them using flipped class model with pre-questions. The questions that students submit are collected online before class every time, the data on the degree of the difficulty of feeling and the understanding of feeling were obtained through the questionnaire. As a result, for learners who are new to programming, the learners should allocate the process of making the problem into a logical abstraction at the beginning of the curriculum before learning the basic concept of computer language, each lesson should be practiced through the bottom-up problems enough to provide a logical understanding before actual coding. In addition, detailed curriculum should be developed according to characteristics of learner's major, contents and conducting level.

Key Words : Difficulties of Novice Programmer, Programming Education, Software Curriculum, Feedback of Programming subject

Received 19 April 2017, Revised 19 June 2017
Accepted 20 July 2017, Published 28 July 2017
Corresponding Author: Hyunsook Kim(Daegu University)
Email: Kim.HS@daegu.ac.kr

1. 서론

컴퓨터 언어를 활용하는 프로그래밍 교육은 문제를 파악하여 창의적으로 해결하는 과정을 반복하기 때문에 사고력, 문제 해결능력, 창의력을 신장시킨다[1]. 특히 컴퓨팅적 사고능력 개발은 IT와 관련 없는 계열 전공자들에게는 자신의 전공 분야에서 IT를 통한 혁신아이템을 찾을 수 있는 능력, 기술에 대한 두려움을 극복하고 활용 대상으로 인식할 수 있는 능력, 혁신적 아이디어를 구현할 수 있는 능력, IT 전문가와 소통하는 능력 등을 갖추도록 도울 수 있다. 이러한 컴퓨팅적 사고능력은 컴퓨터 프로그래밍 언어 교육을 통해서 개발하고 훈련할 수 있다[2]. SW 분야 인재양성을 위해서는 기술적 요인으로 SW 활용도가 증대됨에 따라 SW 융합 인재양성에 중점을 두는 것이 가장 중요하다고 인식하고 있음을 보여준다[3].

이러한 프로그래밍 교육의 중요성을 인지한 현 정부에서는 2014년 7월에 “소프트웨어 중심사회 실현전략” 정책을 발표하였다. 발표된 정책에는 2017년부터 초등학교에 소프트웨어 기초 소양 수업시간을 배정하며, 중학교의 정보과목을 필수 단독과목으로 신설하고, 고등학교는 2018년부터 심화선택인 정보과목을 일반선택으로 지정할 예정이다. 대학에서는 실질적 소프트웨어 전공교육을 강화하여 모든 전공 분야 대학생들에게 실전적 소프트웨어 교육기회를 제공하려고 노력하고 있다. 하지만 이러한 정규교육과정을 운영할 인력과 교수학습 방안에 대한 연구는 주로 초・중・고등학생들을 대상으로 진행됐으며, 대학생을 대상으로 한 코딩교육에 대한 연구는 부족한 실정이다[4, 5, 6].

또한, 제4차 산업혁명은 제조업뿐만 아니라 교육 방법에서도 큰 변화를 예고하고 있다. 따라서 디지털 장비와 서비스에 익숙한 학습자들을 위해서는 전통적인 학습 전략 패러다임을 바꾸고 더 나은 학습 환경을 형성하고 흥미를 끄는 방안을 활용하여 교육하여야 한다. 플립드 러닝은 배움의 중심을 교사에서 학생으로 옮기는 것이며, 학생들의 자기주도학습 능력과 협업 능력을 향상시키고 무엇보다 활기차고 적극적인 수업 분위기를 제공한다[7, 8, 9, 10, 11, 12]. 전통적인 강의실 수업은 정보를 전달하기에는 매우 효율적이지만, 변화하는 교육환경에서 강의자, 학습자 간의 교육방법에 한계에 도달하였다. 이를 해결하기 위한 방법 중 하나가 IT기술을 기반으로 한 콘텐츠 제공과 교육방법의 변화 모색이다[13].

플립드 러닝은 강의자가 동영상 콘텐츠를 사전 제작하여, 수업시간 전에 학생들이 예습할 수 있도록 제공하고, 수업시간에는 사전 학습한 내용을 기반으로 이론을 적용하거나 토론하여 학생들이 예습한 바를 자신의 지식으로 만들 수 있도록 돕는 방식이다. 플립드 러닝은 컴퓨터공학 분야 교육에서 전공자들의 문제 해결 능력과 모델링 능력을 증진하게 한다[14]. 기존의 동영상을 활용하는 이러닝과의 차이점은 강의실 수업 시간은 그대로 유지하면서, 수업 방식을 바꾸어 진행한다는 점이다.

소프트웨어 교육을 처음 받는 초보 프로그래밍 학습자들은 프로그램의 개념을 이해하고 다양한 문제 해결을 해 나가는 것을 특히 어려워하기 때문에 이들에게 학습 효율을 높일 방안이 필요하다[15, 16]. 이에 대한 연구로는 인문계열 학생들을 대상으로 한 소프트웨어 교육 분석이 있으며, 연구 결과 수업이 진행되어 갈수록 프로그래밍에 대한 흥미도가 떨어지기 때문에 이에 대한 보완책이 필요 있음을 지적하고 있다[17].

본 연구는 사전 학습을 통해 문제해결력을 향상할 수 있는 플립드 러닝을 프로그래밍 수업에 적용하는 과정동안 초보 프로그램 학습자들의 어려움을 파악하고, 파악된 어려움에 대처하는 방식의 강의 운영 방안을 모색해 보고자 한다. 소프트웨어 기초교육을 전 교육과정으로 확산하는 시점에서, 본 연구결과는 프로그래밍 교육을 통해 학생들의 컴퓨팅 사고력(Computational Thinking)을 증진 시킬 수 있는 교육과정을 만드는데 일조할 것이다.

연구를 위하여 H 대학에서 진행한 파이썬 프로그래밍 수업 진행 결과를 분석하였다. 플립드 러닝 콘텐츠는 MOOCs 의 Coursera에서 제공하는 미시건 대학 강의 파이썬 프로그래밍 강좌를 활용하였다. MOOCs는 2012년에 시작된 온라인교육의 진화된 형태로, 하버드 대학을 중심으로 제공하는 edX, 스탠퍼드 대학을 중심으로 제공하는 Udacity, Coursera 등이 대표적이며, 글로벌 경쟁력을 갖춘 여러 대학에서 수백 개의 다양한 전공분야의 강좌를 제공하여 급격히 성장하고 있다[18].

학생들에게 프로그래밍 온라인수업을 수강하면서 생긴 질문을 LMS(Learning Management System)에 사전 등록하도록 하고, 강의자는 등록된 질문을 통하여 수강생들이 이해하기 어려워하는 원인을 파악하여 학습자 맞춤형으로 질문에 피드백한 후 오프라인 강의를 진행하였

다. 16주차 동안 4회에 걸친 시험의 학업 성취도를 파악하고, 설문조사를 통해 프로그래밍 수업에서 제출한 질문 유형, 체감 난이도와 체감 이해도를 분석하였다.

논문의 1장은 서론, 2장에서는 이론적 배경을, 3장에서는 연구 방법을, 4장에서는 연구 결과와 마지막 5장은 결론 및 제언으로 구성하였다.

2. 이론적 배경

2.1 플립러닝 개념

플립드 러닝은 교실 수업에 앞서 학습자 스스로 선행학습을 수행한 후 수업에 참여하는 방식으로, 최근의 자기 주도적 학습을 강조하는 교육 정책의 목표와 일치한다. 플립드 러닝은 기존의 블랜디드 러닝 수업 형태 중 하나로 수강생들이 교실 수업 전에 해당 수업 내용을 동영상 자료를 통해 선행 학습하고, 수업 시간에는 스스로 학습한 내용의 완성도를 높여주는 개인화된 보충학습이나 심화학습으로 진행하는 것이다.

교사가 교실에서 학생들에게 일방적 강의를 진행하는 방식과는 달리, 학생들은 제공된 동영상을 자신이 편한 장소에서 자신의 학습 능력에 맞춰 개념을 이해하고 과제를 수행한다. 또한, 교사는 도움을 요청하는 학생들을 대상으로 개별지도에 많은 시간을 할애할 수 있으며, 학생들은 각자 선행학습 한 내용을 바탕으로 강의실 수업에서 부족한 부분에 대해서는 보충학습을, 그리고 더욱 관심이 가고 흥미를 느끼는 부분에 대해서는 심화학습을 할 수 있다는 장점을 가진다[8].

2.2 프로그래밍(코딩) 교육

프로그래밍 교육은 융합형 교육의 효과적인 도구로써 문제 해결 방법을 기획, 설계하는 과정을 거치면서 문제해결능력, 논리적 사고능력, 창의적 사고력 등을 키울 수 있다. 그동안 초·중등 학생들을 대상으로 프로그래밍 교육 연구는 활발히 진행됐지만, 대학생을 대상으로 한 연구는 부족한 실정이다. 현재 대학생들은 중등교육과정에서도 프로그래밍 교육을 거의 받지 못했기 때문에 대학에서 비전공자를 위한 프로그래밍 교육 연구는 다양하게 진행 될 필요가 있다. 대학에서는 단순한 프로그래밍 교육이 아니라 일상생활 속에서 나타나는 문제나 학생의 전공 분야에서 해결해야 하는 문제들을 프로그래밍 교육을 통해 실제적으로 해결해 나가도록 돕는 교육과정과 교육방법이 필요한 것이다[6].

2.3 선행연구

플립드 수업 설계방안[8]과 플립드 수업을 통한 수강생들의 이해도와 학업성취도 향상에 관한 많은 연구가 진행되었다[19, 20, 21]. 플립드 러닝은 프로그래밍 수업을 처음 접하는 과정에 효과적이며, 플립드 러닝을 도입한 과목의 수강생들이 자기효능감과 학업 성취도가 높게 나타났다고 보고한다[19]. 플립드 러닝의 효과적 운영을 위한 방안으로는 사전 동영상 강의 뿐 아니라, 노트작성과 퀴즈, 수업 전 학생들의 질문을 활용하는 방안을 제안하고 있다[20, 21]. 이 결과를 통해 플립드 수업을 진행할 때 사전 동영상 강의의 이해도를 확인하는 다양한 방안이 필요하다는 것을 알 수 있다.

프로그래밍 초보학습자들의 어려움과 오개념의 특성을 파악한 연구에서는[14] 프로그램 수업 교수 학습 방안으로는 프로그램 개념의 정확한 이해와 내면화하기 위한 전략을 수립하여야 하고, 문제 해결 절차를 가시화할 수 있는 정교화 된 알고리즘을 작성하는 연습이 충분히 이루어져야 하며, 문제해결 과정에서 자유롭게 사고할 수 있는 전략수립과 학습해야 하는 프로그래밍 개념을 어떠한 순서로 구성하여 가르칠 것인가에 대한 심도 깊은 설계가 필요하며, 다양한 문제 해결 경험을 제공할 필요가 있음을 보여준다.

비전공자 초보 프로그래밍 학습자를 대상으로 소프트웨어 교육을 한 결과, 프로그래밍의 흥미도는 수업시간을 거듭할수록 조금씩 떨어지는 것으로 나타났으며, 개별 프로젝트를 끝낸 후에 흥미가 급상승하는 것으로 나타났다[15]. 또한, 프로그래밍 초보 학습자들은 자신의 아이디어를 표현하는 데 필요한 디자인적인 요소에 많은 시간을 사용하고 있어서, 프로그램의 논리적인 구성을 하는 데 소홀하다는 결과를 알 수 있다[22].

3. 연구 내용

3.1 연구 목적

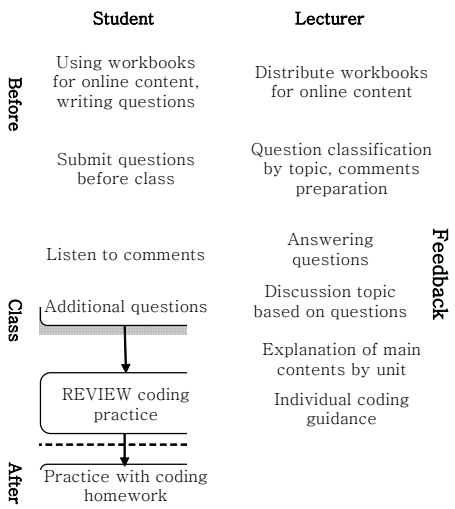
본 연구의 목적은 초보 프로그램 학습자들의 어려움

을 파악하고, 파악된 어려움에 대처하는 강의 운영 방안을 제안하는 것이다. 수강생들의 어려움을 파악하기 위하여 강의실 수업 전에 플립드 수업을 하면서 자신이 이해되지 않거나 어려운 개념에 대해 오프라인 수업 전에 온라인으로 LMS에 질문을 제출하게 한다. 제출된 질문을 통하여 수강생들의 어려움을 파악하였으며, 파악된 어려움에 대처하는 방안을 준비하여 강의시간에 적용하였다. 수업을 종강하면서 수강생을 대상으로 설문조사를 하여 체감난이도와 체감이해도를 분석하고, 사전 질문을 유형별로 확인하였다.

체감 난이도와 체감 이해도는 프로그래밍 수업이 얼마나 어렵게 느껴졌는지, 해당 단원을 얼마나 이해하고 있는지를 항목별 5점 척도로 설문지에 직접 기재한 수강생들의 응답 점수를 환산한 값을 데이터로 사용하였다.

수업 단원은 파이썬 언어의 '언어특성', '연산자', '변수', '조건문', '반복문', '메소드', '문자열', '함수', '리스트(List)', '파일입출력'으로 분류하여 파악하였다.

3.2 연구 방법



[Fig. 1] A flipped class model using preliminary questions

사전질문을 이용한 플립드 수업 모델은 [Fig. 1]에 나타난 것처럼 진행된다. 이 수업모델은 [21]의 논문에서 제안된 모델을 활용하였으며, 수업 전에 온라인 콘텐츠 강의를 수강하면서 생긴 질문들을 해당 주제 강의 시작 전에 LMS에 등록하도록 하였다. 강의자는 제출된 수강생들의 질문을 미리 주제별로 분류하고 해결책을 준비한 후에 강의실 수업을 진행하였다.

3.3 연구대상과 절차

연구를 위하여 파이썬 수강생들의 학업 성취도와 설문조사 자료를 결과 분석에 사용하였다. 연구대상 학교의 학생들은 1학년 때 무전공으로 입학하여 2학년 때 이공/비이공 계열의 전공을 각자 선택한다. 그중 교양교과목으로 파이썬프로그래밍 수업을 수강한 학생들을 대상으로 진행하였으며, 연구 대상의 구체적인 특성 분포는 <Table 1>과 같다. 융합계열은 비이공계열과 이공계열을 병행하는 경우를 의미한다.

<Table 1> Subject characteristics for Research

Factor \ Major		SE (Science&Engineering)	Non-SE (Non-Science &Engineering)	Conv. Convergence	Total	χ^2(p)
Gender	M	24(22%)	22(21%)	10(9%)	56(52%)	2.89
	F	25(23%)	19(18%)	7(7%)	51(48%)	(0.76)
Experience of Programming	Y	5(5%)	17(16%)	5(5%)	27(25%)	0.00
	N	44(41%)	24(22%)	12(11%)	80(75%)	(0.00)

연구대상의 성별 인원은 큰 차이가 없으며, 프로그래밍을 처음 접하는 학생들은 전체의 75%로 구성되어있다. 연구 대상 교과목은 파이썬 프로그래밍이며 2015년 H 대학에 전 학년을 대상으로 이공계열과 비이공계열 관계없이 쉽게 프로그래밍을 경험하게 하려는 목적으로 개설하였다. 본 연구의 목적은 대학 기초 교양교육으로써 소프트웨어 교육의 교육 방안을 모색하는 데 중점을 두고 있으므로 전공을 선택하지 않은 학생들을 대상으로 연구를 진행하였다. 따라서 프로그래밍 언어 선정과 작은 연구대상 표본수로 인해 진행한 연구 결과를 일반화하기에는 부족한 면이 있음을 인정하고 향후 과제로 남긴다.

수업의 효과를 향상하기 위해 플립드 수업 방식으로 운영하였다. 플립드 수업 콘텐츠로는 Coursera의 미시간 대학(University of Michigan) 파이썬 프로그래밍 강좌를 활용하였다. Coursera는 온라인공개강의(MOOC) 서비스 업체 중 가장 많은 가입자를 보유하고 있으며, 그 중 미시간 대학의 '모두를 위한 파이썬(Python for Everybody)

[23]' 은 Coursera를 대표하는 인기 강좌이다.

수강생들에게 오프라인 수업 전 콘텐츠 학습을 독려하고, 이해 정도를 확인하기 위하여 온라인 콘텐츠를 수강할 때 사용하는 워크북을 자체 제작하여 배포하였다. 워크북에는 강의일정, 강의 요약, 간단한 퀴즈, 강의를 수강하면서 생긴 질문들을 쓰도록 구성하였다. H 대학 파이썬프로그래밍 교과 전체 강의구성과 미시간대학 강의구성은 <Table 2>와 같다.

<Table 2> Python Programming Curriculum for Study

Week	H University	Uni. Michigan
1	Installation Python I/O Program	Introduction, -Why we Program
2	Variables, Operator	Expressions
3	If statements	Conditional Code
4	For, While	Iteration
5	Turtle	
6	Stings	Strings
7	Lists	Lists
8	Mid term	
9	Functions	Functions
10	Functions & Modules	Functions
11	Tuple	Tuples
12	Dictionary	Dictionaries
13	Class & Objects	
14	File I/O	Files
15	Graphic	
16	Final Term	

<Table 3> Total the number of questions

subject	questions(%)
Language characteristics	166(13%)
Operators	51(4%)
Variables	89(7%)
Conditional statements	26(2%)
Loop statements	26(2%)
Methods	89(7%)
String	128(10%)
Functions	281(22%)
List	306(24%)
File I/O	115(9%)
Total	1,275(100%)

교수자는 학생들이 수업 3일 전까지 LMS에 제출한 각 질문에 대한 피드백 자료와 이에 대한 대처방안을 미리 준비하였다.

수강생들이 제출한 사전 질문들의 수는 단원별로 <Table 3>과 같다. 질문의 빈도는 '리스트', '함수', '프로그래밍 언어의 특성', '문자열' 순으로 높게 나타났다. <Table 4>에 학생들의 질문 중 일부를 나타내었다.

<Table 4> Type of Pre-Questions

	Pre-Questions
Language characteristics	• Use of Python language • Differences between Python 2.x and 3.x • Differences between Python, C, and Java
Operators	• How to use operator
Method	• Whether to adjust the sort order in the sort method • In the .sort () method, alignment standard
String	• You can also enter a number in Str, which differs from what is stored as an int • Differences between built-in functions and methods in strings
Function	• Difference when function end sentence is written by print, return • I wonder where the built-in functions are stored and why I can use them without having to import modules • Whether the function's def declaration can be used in the middle of the code
List	• Differences between Dictionary, tuple and list • List components (real, integer) (string, real) different data types allowed • Whether the value corresponding to key is only integer type • Why is the order different every time a dictionaries is added?
File I/O	• How to read photo files in addition to document files

4. 연구 결과

4.1 학습자의 어려움 파악

단원별로 파악된 수강생들의 사전질문을 통해 살펴본 프로그래밍 초보 학습자들의 어려움은 다음과 같다. 변수활용 단원에서는 파이썬에서 변수 선언과 메모리 할당에 관한 이해도가 낮았다. 또한, 데이터 타입이 결정되지 않은 변수에 처음에는 숫자를 저장했다가, 이후에 문자열을 저장해도 되는지를 궁금해 한 학생들이 여러 명이었다. 문자열 단원에서는 문자열 관련 메소드 활용에 대해 어려워하였고, 정렬 기준이 무엇인지에 대한 질문의 수가 많았다.

조건절에서는 다중조건 표현에 어려움을 많이 표현하였으며, 특히 다중조건은 'if~elif'를 사용하는 경우와 다중 if문으로 표현하는 경우의 차이점을 이해하기 어려워했다. 이는 논리적인 구조를 이해하지 못하는 것이 원인이라고 볼 수 있다. 반복절에서는 while() 문에서 조건을

사용할 때, 그 조건이 반복조건인지 종료조건인지를 판단하는 데 대한 체감 난이도가 높게 나타났다.

함수 단원에서는 함수의 필요성과 개념을 이해하지 못하는 경우가 가장 많았으며, 매개변수를 사용하는 목적, 리턴값을 활용하는 방법과 목적 등을 이해하지 못한 경우가 아주 많았다. 함수를 구성하는 원리, 함수를 호출하는 원리, 함수가 수행되는 절차 등을 이해하고 활용할 수 있어야 실제적인 프로그래밍을 할 수 있는데, 반복적으로 함수를 정의해서 사용하는 등의 시행착오를 거치는 수강생들이 많음을 확인할 수 있었다.

리스트 단원에서는 인덱스 값과 리스트 구성요소를 구별하는 것을 어려워하였으며, 인덱스를 사용하여 각 요소에 접근 시 반복절 사용의 이해도가 낮았다. 또한, 리스트의 구성요소로 서로 다른 데이터 타입의 데이터를 저장할 수 있는지에 대한 질문도 높게 나타났다. 특히 문자열로 구성된 리스트를 다룰 때 중첩 반복절을 활용하여 문자열 내의 각 글자를 읽어나가는 방식을 어려워하였다. 유사한 데이터 구조를 가지는 리스트, 튜플(Tuple), 딕셔너리(Dictionary)를 모두 다 배운 후에는 각 데이터 구조의 차이점과 활용방안에 대한 질문이 가장 많았다.

파일입출력 단원에서는, 텍스트 파일이 아닌 그림이나 비디오 파일을 읽어 활용 가능한지에 대한 질문이 많았다.

4.2 해결 및 대처방안

수강생들의 어려움을 줄이는 방안으로 다음과 같은 해결 방안을 설계하고 수업에 적용하였다.

프로그래밍에 익숙하지 않을 때는 예제를 보면서 예약어(reserved word)에 관하여 설명하고, 함수 단원을 배우는 7주차에 변수와 메모리의 관계, 변수 활용방안, 전역변수, 깊은복사(deepcopy) 등에 대하여 설명하고 예제를 풀면서 점차 이해하도록 도왔다.

반복절 while() 문에서 조건을 사용할 때 나타난 어려움은 반복문이 정확하게 어떻게 동작하는지 이해하지 못한 것이 원인이기 때문에, 반복문을 설명하는 시간을 늘리고 강의시간에 풀어보는 간단한 예제를 추가하였다.

함수의 필요성과 개념 이해를 돕기 위하여 함수를 사용하는 예제와 그렇지 않은 예제를 동시에 보여주면서, 함수사용의 필요성과 효율성을 설명하였다.

리스트에서 나타난 어려움은 인덱스와 반복절 이해의 부족이라 판단하였다. 그래서 리스트 단원을 시작하면서 인덱스의 의미를 설명하는 여러 예제를 사용하여 먼저 리스트의 구성원리를 이해하도록 강의자료를 수정하고, 리스트에서 각 구성요소와 인덱스의 관계를 반복절을 사용하지 않고 출력하는 연습을 먼저 한 후, 반복절을 사용하여 리스트에 접근하도록 강의를 진행하였다.

파일입출력 단원에서는 파일 입출력이 필요한 이유를 실생활에서 벌어지는 여러 예를(성적처리, 은행관리, 학적관리, 재고관리 등) 들어 설명하였다. 그림 파일을 읽어서 출력하는 경우에는 추가 패키지를 설치하여 그래픽 처리를 하는 예제를 설명하였고, 몇 가지 그래픽 보정하는 기능을 활용하였더니 수강생들의 관심이 높아졌다.

논리구조에 대한 이해가 부족하여 파이썬 기초 문법을 활용하기 어려워했기 때문에, 논리구조를 쉽게 이해하도록 일상생활에서의 문제 해결 과정을 플로차트나 의사코드로 표현하는 교육시간을 수업 초반에 충분히 가지면 학생들이 조금 더 자신감을 가지고 기초 문법을 활용하게 될 것이다.

반복절 사용이 익숙하지 않은 시점에는 리스트 활용이 어렵다. 따라서 리스트 단원을 진행하기 전에 반복절 연습을 충분히 할 수 있도록 준비할 필요가 있다. 리스트 단원에서 튜플과 리스트의 사용 목적과 차이점을 간단히 소개하고 몇 주 후에 다시 자세히 배우도록 구성할 필요가 있다.

파일입출력은 실제 프로그래밍을 할 때 활용 빈도가 높은 편임에도 불구하고, 이 단원을 수업 마무리 부분에 배치하여 반복 연습할 기회가 없어서 이해도가 낮게 나타났다. 파일입출력 단원을 함수 다음으로 이동하는 것이 바람직하다.

4.3 해결방안의 효과성 분석

프로그래밍 수업에 대한 어려움을 분석하기 위해 세부 수업 내용에 대해 학생들이 느끼는 전공 계열별 체감 난이도와 체감이해도를 분석하였다.

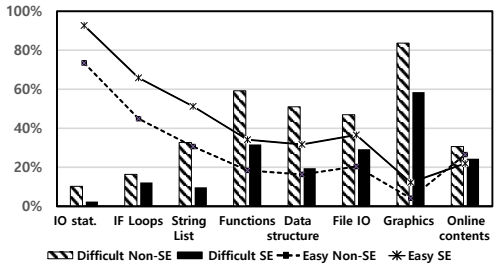
4.3.1 계열별 체감난이도 분석

<Table 5>는 계열별 체감난이도 분석결과로, 비이공계 학생들이 이공계 전공 학생들보다 전체적으로 어려워하는 것으로 나타났다. 특히 자료구조(리스트, 튜플, 딕셔너리)와 함수활용, 파일 입출력 부분에서 어려워하는 비율이 높게 나타났다.

<Table 5> Difficulties of feeling by major

Contents \ Degree of Difficulty / Major		Difficult	Normal	Easy	χ^2(p)
IO stat.	SE	5(10%)	8(16%)	36(73%)	4.55 (0.66)
	Non-SE	1(2%)	7(17%)	33(80%)	
	Conv.	3(18%)	2(12%)	12(71%)	
IF Loops	SE	8(16%)	19(39%)	22(45%)	1.55 (0.18)
	Non-SE	5(12%)	9(22%)	27(66%)	
	Conv.	4(24%)	6(35%)	7(41%)	
String, List	SE	16(33%)	18(37%)	15(31%)	0.95 (0.08)
	Non-SE	4(10%)	16(39%)	21(51%)	
	Conv.	4(24%)	8(47%)	5(29%)	
Functions	SE	29(59%)	11(22%)	9(18%)	0.74 (0.05)
	Non-SE	13(32%)	14(34%)	14(34%)	
	Conv.	7(41%)	4(24%)	6(35%)	
Data structure	SE	25(51%)	16(33%)	8(16%)	0.74 (0.05)
	Non-SE	8(20%)	20(49%)	13(32%)	
	Conv.	9(53%)	5(29%)	3(18%)	
File IO	SE	23(47%)	16(33%)	10(20%)	1.61 (0.19)
	Non-SE	12(29%)	14(34%)	15(37%)	
	Conv.	6(35%)	4(24%)	7(41%)	
Graphics	SE	41(84%)	6(12%)	2(4%)	0.83 (0.07)
	Non-SE	24(59%)	12(29%)	5(12%)	
	Conv.	12(71%)	4(24%)	1(6%)	
Online contents	SE	15(31%)	21(43%)	13(27%)	'5.68 (0.78)
	Non-SE	10(24%)	22(54%)	9(22%)	
	Conv.	4(24%)	7(41%)	6(35%)	

[Fig. 2]는 비이공계열과 이공계열 학생들의 체감난이도에 대한 결과이다. 비이공계열과 이공계열 학생들에 대한 특성을 알아보기 위해 전체 연구대상의 16%를 차지하는 융합계열 전공자는 분석에서 제외하였다. [Fig. 2]의 결과에서 보는 바와 같이 비이공계열 학생들이 이공계열 학생들보다 스트링과 리스트, 자료구조 부분에서 느끼는 체감난이도가 높게 나타나는 것을 확인할 수 있다. 이는 자료저장과 같은 컴퓨터 시스템의 전체적인 동작 원리에 대한 기본적인 이해가 없었기 때문이라고 추측할 수 있다.



[Fig. 2] Difficulties of feeling by major

4.3.2 계열별 체감이해도 분석

계열별 체감이해도의 결과는 <Table 6>과 같다.

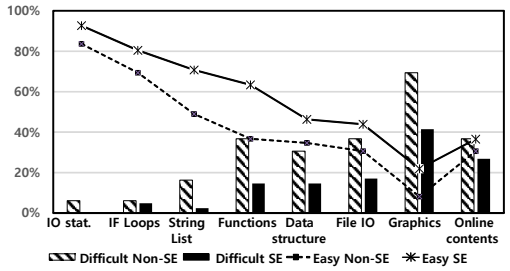
<Table 6> Understanding of feeling by major

Contents \ Degree of Difficulty / Major		Difficult	Normal	Easy	χ^2(p)
IO stat.	SE	3(6%)	5(10%)	41(84%)	0.43 (0.192)
	Non-SE	0(0%)	3(7%)	38(93%)	
	Conv.	1(6%)	4(24%)	12(71%)	
IF Loops	SE	3(6%)	12(24%)	34(69%)	0.820 (0.34)
	Non-SE	2(5%)	6(15%)	33(80%)	
	Conv.	3(18%)	3(18%)	11(65%)	
String, List	SE	8(16%)	17(35%)	24(49%)	0.77 (0.32)
	Non-SE	1(2%)	11(27%)	29(71%)	
	Conv.	2(12%)	4(24%)	11(65%)	
Functions	SE	18(37%)	13(27%)	18(37%)	0.83 (0.33)
	Non-SE	6(15%)	9(22%)	26(63%)	
	Conv.	6(35%)	1(6%)	10(59%)	
Data structure	SE	15(31%)	16(33%)	17(35%)	1.57 (0.55)
	Non-SE	6(15%)	16(39%)	19(46%)	
	Conv.	6(35%)	2(12%)	9(53%)	
File IO	SE	18(37%)	16(33%)	15(31%)	1.19 (0.45)
	Non-SE	7(17%)	16(39%)	18(44%)	
	Conv.	4(24%)	3(18%)	10(59%)	
Graphics	SE	34(69%)	11(22%)	4(8%)	0.78 (0.33)
	Non-SE	17(41%	15(37%)	9(22%)	
	Conv.	9(53%)	7(41%)	1(6%)	
Online contents	SE	18(37%)	16(33%)	15(31%)	2.72 (0.74)
	Non-SE	11(27%)	15(37%)	15(37%)	
	Conv.	1(6%)	11(65%)	5(29%)	

파이썬 프로그래밍 수업 후에는 함수활용, 자료구조, 그래픽 기능을 제외한 부분에서는 비이공계열 학생과 이공계열 학생들의 체감이해도 결과는 비슷하게 나타났다. 비이공계 학생들의 함수 활용에 대한 체감난이도가 60%, 이공계 학생들은 31%였으며, 수업 후 체감이해도는 38%와 63%로 각각 나타났다. 비이공계 학생들의 수업 후 이해도가 이공계 학생들에 비해 다소 낮게 나타나고 있다.

자료구조에 대한 비이공계 학생들의 체감난이도가 51%였으나, 수업 후 이해도는 35%인 반면, 이공계 학생들의 체감난이도는 20%에서 체감이해도가 46%로 높아졌다.

[Fig. 3]의 결과에서 보는 바와 같은 이해도의 편차를 줄이려면, 비이공계 학생들의 프로그래밍에 대한 어려움 해결을 돕는 학습자 맞춤형 설계가 필요하다고 판단된다.



[Fig. 3] Understanding of feeling by major

5. 결론 및 제언

본 연구의 목적은 초보 프로그램 학습자들의 어려움을 파악하고, 파악된 어려움에 대처하는 강의 운영 방안을 제안하는 것이다. 소프트웨어 기초교육을 전 교육과정으로 확산하는 시점에서, 본 연구결과는 정규교육과정의 모든 학습자에게 적절한 프로그래밍 교육과정과 교육방법을 설계하는데 도움이 될 것이다.

연구결과 수강생들의 질문을 통해 파악된 어려움은 다음과 같다. 미리 선언하지 않고 사용하는 변수에 대한 이해와 문자열 관련 메소드 활용과 정렬기준, 다중조건문 표현, 반복절 while() 문에서의 조건 사용, 함수를 구성하고 호출하는 원리, 함수가 수행되는 절차의 이해와 활용을 이해하기 어려워하였으며, 리스트에서는 인덱스 값과 리스트 구성요소를 구별을 잘 하지 못했다.

수업사례를 통해 프로그램 수업을 처음 접하는 학생들의 이해력을 높이기 위한 효과적 대처방안을 다음과 같이 제안한다.

프로그래밍에서 가장 중요한 변수에 대한 이해를 높이려면, 초보 학습자들에게는 변수 사용 예제를 먼저 보고 따라하게 하고, 프로그래밍에 조금 익숙해진 후에 변수와 메모리의 관계 등을 점증적으로 설명하여 이해도를 높일 필요가 있다. 조건절에서는 논리구조를 쉽게 이해하는 것을 돕기 위해 플로차트를 그려서 문제해결 절차를 표현하는 수업을 2~3주 동안 진행하는 것이 효과적이다. 또한, 함수에서는 매개변수와 리턴값을 사용하는 내장함수 활용 예제를 실행해 보고, 이 내장함수가 어떻게 기술되었는지 코드를 점검하면서 이해도를 높여야 한다. 마지막으로 리스트에서는 리스트 단원을 진행하기 전에 반복절 연습을 충분히 하여야 하며, 리스트, 튜플, 딕셔너리의 차이점을 명확하게 설명하여야 한다.

기본 문법 이해 외에도 주어진 문제를 프로그래밍하려면 배운 파이썬 문법과 어떻게 연결하는지 잘 모르겠다는 질문 빈도수가 높았다. 이러한 어려움을 극복하기 위해서는 컴퓨터 언어의 기본 개념을 배우기에 앞서 문제를 먼저 파악하여 논리적인 구조로 만드는 연습을 하는 준비과정을 교육과정 초반에 배치하는 것이 필요하다. 즉, 각 단원의 문법을 배운 후 실제 코딩하기 전에 논리적인 연습을 할 수 있는 시간을 갖게 할 필요가 있다.

설문조사를 통하여 체감난이도를 분석한 결과, 비이공계열 학생들이 이공계열 학생들보다 스트링과 리스트, 자료구조 부분에서 어려워하는 것으로 나타났다. 이는 자료저장과 같은 컴퓨터 시스템의 전체적인 동작원리에 대한 기본적인 이해가 어렵기 때문이라고 추측한다. 비이공계열 학생과 이공계열 학생들의 함수활용, 자료구조, 그래픽 기능을 제외한 부분의 체감이해도 결과는 비슷하게 나타났다. 계열별로 체감난이도와 체감이해도 차이가 나타나기 때문에 학습자의 특성에 따른 맞춤형 교육과정과 교육방법을 개발하여야 하며, 유사한 특성을 가진 학습자들 간 수준별 수업 설계가 필요하다.

제안한 본 연구의 결과는 비이공계 학생들의 융합적 사고함양을 위한 대학 기초교양 소프트웨어 교육 방법을 설계하는 데 유용하게 사용할 수 있을 것이다.

또한, 프로그래밍 수업을 운영할 때 제언할 점은 첫째, 초보 프로그래밍 학습자들에게 두려움을 극복하고, 학습의욕을 높여주기 위해서는 수강생들이 느끼는 수업 내용에 대한 어려움을 파악하고, 학습자의 진행 단계를 반영한 정밀한 교육과정 설계가 선행되어야 한다. 같은 컴퓨터 언어를 교육하더라도 학습자의 연령, 성향 및 학습 태도를 반영한 맞춤형 교육과정이 필요한 것이다. 둘째, 질문을 활용하는 플립드 수업이 프로그래밍 교육에 효과적이므로, 각 정규교육과정에서 사용할 수 있는 다양한 온라인 교육 콘텐츠를 학습 단계별로 준비하여야 한다는 것이다.

본 연구는 교육연구의 일환으로 초보 프로그래밍 수강생들의 어려움을 파악하여 효과적인 프로그래밍 교육을 시행하려는 방안을 모색하는데 그 의의가 있다. 향후 연구과제로는 풍부한 표본집단을 대상으로 초보 프로그래밍 수강자의 특성을 파악하는 연구와 다양한 프로그램 언어를 다루는 교육과정에 대한 것이다. 또한, 본 연구의

결과를 활용하여 개선된 프로그래밍 수업을 진행하여 실제적인 수업효과를 분석하고, 프로그래밍 초보 학습자나 비전공자를 대상으로 적절한 교육 방법과 수업 설계 방안을 연구하는 것이다.

REFERENCES

[1] S. H. Jin, S. B. Shin, "Case Study and Needs Analysis on Convergence Education in Engineering Colleges", Journal of Engineering Education Research, Vol. 16, No. 6, pp. 29-37, 2013.

[2] K. M. Kim, H. S. Kim, "A Case Study on Necessity of Computer Programming for Interdisciplinary Education", Journal of Digital Convergence, Vol. 12, No. 11, pp. 339-348, 2014.

[3] J. M. Lee, M. H. Rim, "Derivation of Creative SW HRD Policy Using Analytic Hierarchy Process", Journal of Digital Policy & Management, Vol. 11, No. 10, pp. 95-102, 2013.

[4] D. J. Kim, E. Y. Ha., "The Future Direction of Information Education in University according to Computerization", Journal of Digital Convergence, Vol. 14, No. 10, pp. 33-40, 2015.

[5] S. H. Park, "Study of SW Education in University to enhance Computational Thinking", Journal of Digital Convergence, Vol. 14, No. 4, pp. 1-10, 2016.

[6] S. Y. Pi, "A Study on Coding Education of Non-Computer Majors for IT Convergence Education", Journal of Digital Convergence, Vol. 14, No. 10, pp. 1-8, 2016.

[7] J. Y. Lee, S. H. Park, "An Exploratory Study on Educational Significance and Environment of Flipped Learning", Journal of Digital Convergence, Vol. 12, No. 9, pp. 313-323, 2014.

[8] D. Y. Lee, "Research on Developing Instructional Design Models for Flipped Learning", Journal of Digital Convergence, Vol. 11, No. 12, pp. 83-92, 2013.

[9] J. Lee, H. K. Park, "A Study on Cases for Application of Flipped Learning in K-12 Education", Journal of Digital Convergence, Vol. 14, No. 8, pp. 19-36, 2016.

[10] D. Y. Lee, J. H. Park, "Exploring new directions of flipped Learning with a focus on teachers' perceptions", Journal of Digital Convergence, Vol. 14, No. 8, pp. 1-9, 2016.

[11] S. J. Heo, "Learning Effect Analysis for Flipped Learning based Computer Use Instruction", Journal of the Korea Convergence Society, Vol. 8. No. 1, pp. 155-162, 2017.

[12] Y. Park, "A Theoretical Exploration of Pedagogical Meaning of Flipped Learning from the Perspective of Dialogism", Journal of the Korea Convergence Society, Vol. 8. No. 1, pp. 173-179, 2017.

[13] S. Y. Pi, "Educational Utilization of Smart Devices in the Convergence Education Era", Journal of Digital Convergence, Vol. 13, No. 6, pp. 29-37, 2015.

[14] Redekopp, M. W., Ragusa, G., "Evaluating Flipped Classroom Strategies and Tools for Computer Engineering", Proceedings of the 120th American Society of Engineering Education Annual Conference & Exposition, 2013.

[15] J. W. Choi, Y. J. Lee, "The analysis of learners' difficulties in programming learning", Journal of Korean Association of Computer Education, Vol. 17, No. 5, pp. 89-98, 2014.

[16] J. H. Ku, "Designing an App Inventor Curriculum for Computational Thinking based Non-majors Software Education", Journal of Convergence for Information Technology, Vol. 7, No. 1, pp. 61-66, 2017.

[17] J. S. Sung, S. H. Kim, H. C. Kim, "Analysis of Art and Humanity Major Learners' Features in Programming Class", Journal of Korean Association of Computer Education, Vol. 18, No. 3, pp. 25-35, 2015.

[18] M. N. Choi, H. L. Roh, "A study about a convergence development plan of MOOCs based e-learning in university", Journal of Digital Convergence, Vol. 13, No. 7, pp. 9-21, 2015.

[19] Souza, M. J. D., Rodrigues, P., "Investigating the effectiveness of the Flipped Classroom in an introductory Programming Course", The New Educational Review, Vol. 40, No. 2, 2015.

[20] Maher, M. L., Latulipe, C., Lipford, H., Rorrer, A., “Flipped classroom strategies for CS education.”, Proceedings of the 46th ACM Technical Symposium on Computer Science Education, pp. 218-223, 2015.
[21] K. M. Kim, K. Yi, “Flipped Class Model with Pre Questions and Its Application to Programming class for Novice Learners”, Journal of Korean Association of Computer Education,, Vol. 20, No. 2, pp. 11-14, 2016.
[22] S. H. Kim, S. K. Han, H. C. Kim, “Analysis of Programming Processes Through Novices”, Journal of Korean Association of Computer Education, Vol. 14, No. 1, pp. 13-21, 2011.
[23] Michigan University Python MOOC, https://www.coursera.org/specializations/python

김 경 미(Kim, Kyung Mi)

- 1987년 2월 : 고려대학교 수학교육학과 졸업
- 1992년 2월 : 한국외국어대학교 무역정보학과 석사
- 2007년 2월 : 경북대학교 컴퓨터공학과 박사
- 1986년 12월 ~ 1996년 9월 : 한국과학기술원 전자계산소 기술원
- 1997년 3월 ~ 현재 : 한동대학교 GLS학부 교수
- 관심분야 : 교육 콘텐츠, 프로그래밍 교육, 센서네트워크, 라우팅프로토콜, IT융합교육
- E-Mail : kmkim@handong.edu

김 현 숙(Kim, Hyun Sook)

- 2007년 2월 : 경북대학교 컴퓨터공학과 박사
- 2015년 3월 ~ 현재 : 대구대학교 기계공학부 교수
- 관심분야 : 교육 콘텐츠, 소프트웨어 교육, IT 융합, 모바일프로그래밍, 모바일 콘텐츠
- E-Mail : Kim.HS@daegu.ac.kr