석 사 학 위 논 문

초등학생 대상 파이썬 활용 탐색 알고리즘 수행시간 교육 프로그램 개발 및 적용

Development and application of Search algorithm execution education program for elementary school students using Python

제주대학교 교육대학원

초등컴퓨터교육전공

공 기 표

2019년 8월

초등학생 대상 파이썬 활용 탐색 알고리즘 수행시간 교육 프로그램 개발 및 적용

Development and application of Search algorithm execution education program for elementary school students using Python

지도교수 김 종 훈

이 논문을 교육학 석사학위 논문으로 제출함

제주대학교 교육대학원

초등컴퓨터교육전공

공 기 표

2019년 5월

공 기 표의

교육학 석사학위 논문을 인준함

심사위원장 김 종 우 인

심사위원 박 남 제 인

심사위원 김 종 훈 인

제주대학교 교육대학원

2019년 6월

목 차

표 목 차

그 림 목 차

국 문 초 록

초등학생 대상 파이썬 활용 탐색 알고리즘 수행시간 교육 프로그램 개발 및 적용

공 기 표

제주대학교 교육대학원 초등컴퓨터교육전공

지도교수 김 종 훈

본 연구는 초등학생의 논리성 향상을 위해 파이썬을 활용한 탐색 알고리즘 기반 수행시간 비교 및 분석 교육 프로그램을 개발하고 적용하여 그 효과를 분석했다. 본 교육 프로그램은 ○○도내 초등학교 6학년 133명을 대상으로 실시한 사전 요구분석 결과를 활용하였고, ADDIE(Analysis-Design-Development - Implementation-Evaluation) 모형의 절차에 따라 개발하였다. 개발한 교육 프로그램의 효과를 검증하기 위해서 ○○대학교에서 실시한 교육기부 프로그램의 지원자 25명을 대상으로 6일간 42차시 수업을 진행하였고, GALT(Group Assessment of Logical Thinking)검사를 통해 교육의 사전·사후 효과를 비교·분석하였다. 분석해 본 결과, 본 연구에서 개발한 SW(Software)교육 프로그램이 초등학생의 논리성에 긍정적인 영향을 줄 수 있다는 것을 알 수 있었다.

주요어 : 파이썬, 탐색알고리즘, SW교육, 논리성

Ⅰ. 서 론

1. 연구의 필요성

최근 세계 경제·사회 환경이 소프트웨어를 중심으로 급격하게 변화하고 있다. 여기서 말하는 소프트웨어 중심사회는 소프트웨어가 혁신, 성장 그리고 가치창출의 중심이 되어서 개인과 기업 그리고 국가의 경쟁력을 좌우하는 사회라고 할 수 있다.(미래창조 과학부, 2014) 이러한 시대의 흐름을 반영하여 교육의 방향도 변화하고 있다. 기존의 컴퓨터 활용교육 중심에서 SW교육이 정보교육의 주류가 되고 있다. SW교육은 국가의 경쟁력을 향상시킬 수 있는 수단이며 문제해결능력과 논리적 사고력을 길러주고 컴퓨터의 정보처리과정을 이해할 수 있게 한다는 측면에서 교육적 가치를 지닌다. 또한 21세기의 의사소통 방식이 디지털화 되어 간다는 점을 고려 할 때, 프로그래밍 능력은 모든 학습자 더 나아가 사회 구성원 전체가 갖춰야 할 기본 소양이 된 것이다.(정구원, 2015)

교육선진국이라 할 수 있는 영국, 미국, 핀란드 등에서는 이미 소프트웨어에 관심을 가지고 필수 과목으로 지정하여 교육을 실시하고 있다.(김종훈, 2015) 우리나라 역시 2015 개정 교육과정의 역량 중 지식정보처리 역량을 신장시킬 수 있게끔, 학교에서는 2018년부터 SW교육이 의무화되고, 2019년부터 초등학교에서도 5~6학년 학생들이 SW교육을 필수적으로 이수하도록 개정되었다.(교육부. 2015)

SW교육을 위해 만든 교육용 프로그래밍 언어의 종류는 다양하다. 하지만, 학생들에게 흥미와 관심을 가지게 하면서 학습하기 쉽고 여러 가지 형태의 응용프로그램으로 확장할 수 있는 언어로 접근하는 것이 알맞다.(김경미・김현숙, 2014) 텍스트 언어를 배우면 다른 텍스트 프로그래밍 언어에 전이성이 높아서 기본적인 프로그래밍의 지식과 구조를 쌓게 되고 그 지식을 프로그램을 개발하는 데 활용, 적용할 수 있어서 용이하다. 텍스트 프로그래밍 언어가 익숙해지기까지 시간이 오래 걸리지만 복잡한 프로그래밍을 하는 데 유리하고 블

록형 프로그래밍 언어에 비해 정보과학사고능력(Computational thinking) 향상에 도움이 된다는 연구결과가 있다.(서성원, 2010) 그러나 대부분의 초등교육현장에서는 초기진입단계가 쉽고, 흥미롭게 접근할 수 있는 요소가 많은 엔트리나 스크래치와 같은 블록코딩 기반의 교육용 언어를 사용하고 있었다.

이에 따라 본 연구에서는 초등학생의 논리성을 향상시키기 위해 텍스트 프로그래밍 언어인 파이썬을 활용하였고, 사전 요구분석을 하여 파이썬을 통한 탐색 알고리즘에 수행시간 비교 및 분석에 중점을 두어서 교재 및 프로그램을 개발하였다. 초등학교 4, 5, 6학년 학생들 중에서 지원자 표본 25명의 학생을 대상으로 투입하였다. 논리성은 GALT검사지의 논리력 사고력 항목인 '보존(Preservation)', '비율(Ratio)', '변인통제(Variable control)', '확률(Probability)', '상관(Correlation)', '조합(Combination)' 6가지 요소들로 구분하여 설정하였다.

Ⅱ. 이론적 배경

1. 파이썬(Python)

Python은 1991년에 네덜란드의 귀도 반 로섬 (Guido van Rossum)이 개발한 객체지향 프로그래밍 언어로, 국내에서도 많이 알려져 있는 공개 소프트웨어 중 하나이다. 외국에서는 학습의 목적은 물론 실용적인 부분에서도 많이 사용되고 있는 프로그래밍 언어이다. 그 대표적인 예를 보면, 구글(Google)이나 인포시크(Infoseek)에서 사용되는 검색 프로그램들, 야후의 인터넷 서비스 프로그램, NASA, 유튜브(Youtube) 등이 Python으로 개발되었다. 그리고 Python은 윈도(Windows)와 리눅스(Linux) 그리고 매킨토시(Mac) 등 대부분의 운영체제를 지원하기 때문에 이식성과 확장성에 있어서 좋다.(박응용, 2001) 그리고 파이썬 프로그래밍 언어는 비교적 배우기 쉽고 그래픽 처리 기능이 단순해서 프로그래밍을 처음 접하는 초보자가 배우기에 적절하다. 또한 앱이나 웹 형태로도 개발하기 유용하기 때문에 융합형 교육을 한 프로그래밍 언어로도 활용 가능성이 높다는 장점을 가지고 있다.(Python Software Foundation, 2017)

2. SW교육

SW교육이란 현실 속에 존재하는 다양한 문제에 대해 해결방법을 '컴퓨터' 기반으로 찾는 것을 말한다. 즉, 컴퓨터를 이용하여 자료를 수집, 분석하고 효율적으로 해결하는 과정을 통해 사고력을 향상시키는 교육이라고 할 수 있다. 학습자는 소프트웨어 교육을 통해 컴퓨터가 문제를 해결하는 방식에 대해 이해하게 되며, 컴퓨터의 문제해결 방법과 비슷한 논리로 문제에 접근하게 된다. 이러한 일련의 과정을 거치면서 자연스레 프로그래밍 언어를 학습하게 되고 문제해결능력 및 사고력 향상을 기대할 수 있게 된다.(고준강, 2018)

SW교육에서는 창의융합 인재를 양성하는 것을 목표로 삼으며 '생활과 소프트웨어', '알고리즘과 프로그래밍' 그리고 '컴퓨팅과 문제해결'의 3가지 영역을 통해 정보윤리의식과 태도를 통해 실생활에서 발생하는 문제들을 '컴퓨팅 사고력'으로 해결할 수 있는 역량을 지닌 인재로 키워내는 교육이라고 할 수 있다.(김한성, 2016) 비교적 최근에 들어서야 SW교육에 관한 연구가 활발하게 진행되고 있기에 SW교육의 개념적 구분이 아직까지는 명확히 합의된 것은 아니지만 공통적으로 '컴퓨팅 사고'를 통해 학습자의 문제 해결력 향상에 좀 더 중점을 두고 있는 교육 방법이라고 말할 수 있다.(성정숙・김현철, 2015)

3. 논리적 사고

논리적 사고력의 개념에 대해서 학자들마다 의견이 다르며 합의된 의견을 도출하기에는 어려움이 있어 보인다.(소홍열, 1982) 하지만 대체로 논리적 사고는 귀납적-연역적 추리와 관련되며 보다 넓은 비판적 사고라고 볼 수 있다.(Fisher, R.(ed.) 1987) 좁은 의미의 논리적 사고는 전통적인 논리학의 근간이 되며 논증에 있어서 형식 논리에 보다 중점을 두고 있다. 반면 넓은 의미의 논리적 사고는 사고의 형식적인 측면을 기초로 하지만 사고 내용의 타당성 여부를 고려하는 비판적인 측면이 부가되는 정신능력을 일컫는다.(곽병선, 1985)

본 연구에서는 논리적 사고에 대한 개념을 '대상들을 논리적으로 분석 비교하고, 비판적으로 볼 수 있으며 주어진 대상들 간의 관계를 타당성에 근거를 두고 사고하는 추리 능력'으로 정의하고, 논리적 사고력을 측정하기 위한 도구로 미국 Georgia대학에서 개발된 GALT(Group of Logical Thinking) 검사지를 활용한다. GALT에서 측정하는 논리적 사고의 하위요소는 <표 Ⅰ-1>과 같다.(이민영, 2016)

<표 Ⅰ-1>논리적 사고력 유형

논리 유형	내용
보존 논리 (Conservation Reasoning)	물체의 양, 크기, 모양, 위치가 변화되어도 물체는 일정하게 보존된다는 개념을 이해하는 논리
변인 통제 논리 (Controlling Varuavles Reasoning)	주어진 상황에 영향을 끼칠 것이라고 예상되는 모든 변인들 중에서 특정한 변인이 나타내는 효과를 알아보기 위해 특정 변인 이외의 모든 변인들을 통제함으로써, 주어진 상황과 특정 변인과의 관계를 파악할 수 있는 논리
비례 논리 (Proprotional Reasoning)	두 비가 주어졌을 때 그 비의 값이 동일하다는 논리를 기초로 하여 어떤 문제의 정량적인 관계를 파악하고 해결하는 논리
확률 논리 (Probabilistic Reasoning)	우연히 일어난 사건 중에서 특정한 어떤 사건이 일어날 확률을 이해하고 계산 할 수 있는 논리
조합 논리 (Combinational Reasoning)	문제를 해결해 나가는 과정에서 나타날 수 있는 모든 경우의 수를 빼놓지 않고 또한 중복되지 않게 셈 할 수 있는 논리
상관 논리 (Correlational reasoning)	두 개념들 또는 두 대상 간의 관련성 여부를 규명할 수 있는 논리

4. 텍스트형 프로그래밍 언어

본래 텍스트형 프로그래밍 언어는 컴퓨터 프로그래밍에 필요한 명령어들을 문자로 입력하는 방식으로 정확하게 구문을 사용해야 한다는 것과 언어 습득을 위해 오랜 시간이 걸린다는 단점들이 있다. 하지만 프로그램의 작성 능력이 향상됨에 따라 프로그래밍 시간을 단축하고, 비교적 짧은 코드로 다양한 프로그래밍을 만들 수 있다는 장점이 있다. 그리고 디버깅을 통해서 분석적이고 논리적인 사고가 향상될 수 있다.(유진아, 2008)

텍스트형 프로그래밍 언어를 사용하는 학습 프로그램에서 먼저 고려해야 하는 것은 학습자들의 인지적 부담의 해소 문제이다. 학습이 이루어지는 동안 학습자는 프로그래밍 언어의 문법 학습에 치중하기보다는 문제를 해결하는 과정과 방법이 더욱 중요하다.(박대륜, 2018)

5. 탐색 알고리즘

탐색 알고리즘은 기억 공간에 저장된 데이터 혹은 주어진 입력 데이터 집합에서 어떠한 조건이나 성질을 만족하는 데이터를 탐색하는 알고리즘으로써, 자료 구조 및 알고리즘 과목에서 가장 기본이 되는 알고리즘이다. 탐색 알고리즘은 실제 프로그래밍에서 아주 많이 쓰이고 있으나, 기본 개념이 간단하고 직관적이라서 교육에 있어 수월한 알고리즘으로 알려져 있다.(정인기, 2002)

본 연구에서는 탐색 알고리즘 중에서 주어진 데이터 집합에서 원하는 데이터를 처음부터 순차적으로 찾는 선형 탐색(Linear Search)와 정렬된 데이터 집합에 대해 이분화하면서 탐색하는 방법인 이진 탐색(Binary Search) 을 중점적으로 비교하면서 수행시간을 분석하고자 한다.

6. 알고리즘 수행시간

알고리즘 수행시간은 알고리즘이 어떤 문제를 해결하는 데 걸리는 시간을 말한다. 동일한 문제를 해결하는 데 있어 같은 결과를 도출하더라도 얼마나 오랜 시간이 걸리는지에 따라서 컴퓨터의 효율이 결정되기 때문에 알고리즘 성능을 평가하는데 중요한 기준이 된다. 알고리즘 수행시간을 시간복잡도(Time Complexity)라고도 하며 명령문의 수행 빈도수를 측정하여 분석할 수 있다.(전산용어사전 편찬위원회, 2005) 본 연구에서는 초등학생들이 '알고리즘 수행시간'이라는 개념에 대해서 이해해보고, 단순히 문제를 해결하는 것을 넘어서 최소한의 시간으로 문제를 해결하는 것에 대한 고민과 중요성을 느껴보고 시행착오를 거쳐 실습해볼 수 있는 알고리즘 교육에 중점을 두었다

7. 선행연구분석

김진동, 양권우의 연구에서는 초등학생에게 실생활 속 사례들에서 여러 가지 알고리즘 학습을 실시하였다. 그 결과 논리적 사고력에 있어서 유의미한 차이가 있음을 보였다. 본 연구에서는 이를 발전하여, 컴퓨터를 활용한 SW교육에서 알고리즘에 대한 수행시간 분석 교육이 논리적 사고력에 미치는 영향을 분석하였다.(김진동, 양권우, 2010)

박대륜의 연구에서는 초등학교 6학년을 대상으로 로봇 활용 파이썬 학습 프로그램을 개발하고 적용하였다. 그 결과로 로봇 활용 파이썬 학습 프로그램은 학생들의 컴퓨팅 사고력에 긍정적인 영향을 주었으며 초등학교 학생들에게도 텍스트형 프로그래밍 언어를 이용한 학습 프로그램이 의미 있다는 점을 밝혔다.(박대륜, 2018)

문미예의 연구에서는 블록형 프로그래밍 학습이 가지는 한계점들을 극복하기 위해서 텍스트형 프로그래밍 언어인 파이썬으로 교육하기 위한 방안을 개발하였다. 그 결과, 학생들에게 긍정적인 동기를 부여하고 학생들의 아이디어와 생

각을 구현할 수 있는 학습도구가 될 수 있을 것이라는 결론을 내렸다. 다만, 학생들에게 실제로 적용을 해보지 못한 한계가 있음을 밝혔다.(문미예, 2018)

유인환의 연구에서는 텍스트 기반 프로그래밍 언어인 파이썬을 적용한 SW교육이 기존의 코드블록 중심의 SW교육에 다양성을 추구할 수 있으며, 앞으로의 활용 가능성에 의미를 발견할 수 있었다고 밝혔다. 다만 한계점으로 초등학생에게 영어의 의미 파악, 코딩 방식 자체에 대한 어려움이 있었고 보완 대책이 필요함을 제언하였다.(유인환, 2018)

이에 본 연구에서는 기존에 언플러그드를 활용한 알고리즘 교육으로 논리석 사고력 향상을 이끌어낸 연구를 넘어 스스로 탐색 알고리즘 수행시간 분석을 중심으로 논리적 사고력에 어떠한 영향을 미치는지 연구한다. 그리고 파이썬을 활용하여 컴퓨팅 사고력뿐만 아니라 논리적 사고력에도 긍정적인 영향력을 미칠 것이란 가설을 세우고 연구를 진행하였고, 학생들에게 실제로 적용해보아 연구의 타당도를 높이고자 한다. 또한 텍스트 기반 프로그래밍 언어에 많은 장점을 가지고 있음에도 어렵다는 문제점을 보완하기 위해 초등학생 수준에 맞게 교재를 개발하였다.

Ⅲ. 연구 방법

1. 연구 가설

귀무가설: 파이썬을 활용한 탐색알고리즘 수행시간 중심 SW교육이 학습자의 논리성에 미치는 긍정적인 영향이 없다.

대립가설: 파이썬을 활용한 탐색알고리즘 수행시간 중심 SW교육이 학습자의 논리성에 미치는 긍정적인 영향이 있다.

2. 연구대상

본 연구에서 개발한 프로그램의 효과를 알아보기 위해 OO대학교에서 교육기부 프로그램을 실시하였다. 프로그램의 지원자 표집에 의한 지원자 표본(Volunteer sample) 25명의 학생을 선정하였다. 연구대상 관련 학년 및 성별은 〈표 Ⅱ-1〉과 같다.

〈표 Ⅱ-1〉 연구대상 학년 및 성별

학년	남성	여성	합계
4 학년	5	0	5
5 학년	6	2	8
6 학년	7	5	12
합계	18	7	25

3. 교육프로그램

본 연구에서는 교육 프로그램 개발의 모형 중 하나인 Dick과 Carey의 ADDIE 모형에 따라 교육 프로그램을 〈표Ⅱ-1〉와 같이 개발하였다. 총 5단계인 분석, 설계, 개발, 실행, 평가를 순서대로 실시하였고 학습자의 요구분석결과에 따라 교육프로그램을 설계 및 개발하였고 수업을 통해 실행했으며 그 결과를 논리성 검사를 통해 평가하였다.

〈표Ⅲ-1〉 교육 프로그램 개발 모형

Analysis (분석)	학습자 분석 사전 요구분석(고학년 대상 설문)
Design (설계)	수행목표 설정 및 활용도구 선정 -파이썬을 활용한 탐색알고리즘 수행시간 교육의 효과 평가도구 설계 -논리성 검사(GALT)
Development (개발)	교수·학습 과정안(42차시) 학생 활동지(42차시)
Implementation (실행)	SW교육 실시
Evaluation (평가)	학습자 사후 논리성 검사(GALT)

가. 요구분석

본 연구는 ADDIE모형의 절차에 따라 Rossett의 요구 분석 모형을 사용하였다. Rossett 모형은 기업 교육에서 활용되는 교육 요구 분석 모형으로써 요구 분석의 실행자들이 적용하기 쉬운 안내를 제공한다. 요구분석은 초등학교 6학년 133(남72, 여61)명을 대상으로 실시하였다.

〈표 Ⅳ-1〉 SW교육 경험

	경험해본 적 있다	경험해본 적 없다
응답	82(61.7%)	51(38.3%)

SW교육을 받아 본 적 있는지의 여부는 〈표 Ⅳ-1〉과 같다. 과반수의 학생들이 SW교육 경험이 있음을 알 수 있었다.

〈표 Ⅳ-2〉 SW교육 흥미도

흥미도	응답
매우 그렇다	19(14.3%)
그렇다	35(26.3%)
보통이다	46(34.6%)
그렇지 않다	16(12%)
매우 그렇지 않다	17(12.8%)

SW교육에 대한 학생들의 관심도는 〈표 Ⅳ-2〉의 결과와 같다. 긍정 응답이 부정 응답보다 더 높은 것을 알 수 있었다.

〈표 Ⅳ-3〉 SW교육의 필요성

필요성	응답
매우 그렇다	29(21.8%)
그렇다	44(33.1%)
보통이다	43(32.3%)
그렇지 않다	12(9%)
매우 그렇지 않다	5(3.8%)

SW교육의 필요성에 대해서는 〈표 Ⅳ-3〉을 볼 때 다수의 학생들도 동의하는 것으로 드러났다.

〈표 Ⅳ-4〉 SW교육 도구

SW교육 도구	응답
교육용 프로그래밍 언어(스크래치, 엔트리 등)	52(63.4%)
피지컬 컴퓨팅 도구	19(23.2%)
언플러그드	6(7.3%)
텍스트형 프로그래밍 언어	4(4.9%)

SW교육을 받아본 적 있는 학생들(N=82) 대상으로 SW교육에서 주로 받아본 수업 도구에 대한 결과는 〈표 Ⅳ-4〉와 같았다. 주로 교육용 프로그래밍 언어(스크래치,엔트리 등)을 통한 교육이 주를 이루었고, 뒤를 이어 피지컬 컴퓨팅 도구를 활용한 것으로 나타났다. 반면, 텍스트형 프로그래밍 언어의 경우는 가장 낮은 비중을 차지하였다.

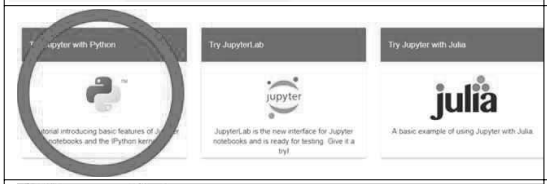
〈표 Ⅳ-5〉 알고리즘 수행시간 교육의 필요성

필요성	응답
매우 그렇다	14(10.5%)
그렇다	24(18%)
보통이다	78(58.6%)
그렇지 않다	8(6%)
매우 그렇지 않다	9(6.8%)

알고리즘 수행시간에 대한 교육의 필요성에 대해서 〈표 Ⅳ-5〉와 같았다. '모르겠다'라고 답한 학생이 과반수 이상이었는데, 학생들에게는 '알고리즘 수행시간'에 대한 개념이 다소 어렵거나 생소하였기에 중립응답이 많았을 것이라고 추측한다. 그리고 긍정응답(28.5%)이 부정응답(12.8%)에 비해 조금 더 높은 비중을 차지하였음을 알 수 있다.

나. 프로그램 설계

본 연구에서 텍스트형 프로그래밍 언어를 접해본 학생 및 파이썬을 활용해본 학생들이 거의 없었기에 따로 설치가 필요 없으며, 인터넷만 연결되어 있다면 언제든지 사용할 수 있는 Project Jupyter의 파이썬 툴을 사용한다. Jupyter 속 파이썬에서는 별도의 소프트웨어 설치 없이 코드, 텍스트, 데이터 시각화 등을 지원하며 가정으로 돌아가 과제를 수행할 시에도 간편하기에 학생들의 학습용으로 적합하다. 다음은 교재의 가장 첫 부분으로 Project Jupyter를 통해 파이썬을 시작하는 방법을 설명하기 위한 그림이다.

	Step1. 구글 크롬 브라우저를 설치합니다.
	Step2. 주소창에 try.jupyter.org를 입력합니다.
	Step3. 빨간색 동그라미로 표시된 'try jupyter with Python'을 클릭합니다,
	Step4. File탭에서 New Notebook Python을 클릭합니다.

〔그림 Ⅰ-1〕 파이썬 시작 방법 설명

파이썬을 활용한 SW교육 프로그램을 설계하기 위해서 교육 대상자의 학습수준 및 프로그래밍의 난이도를 고려하여 구성하였다. 학습내용은 〈표Ⅴ-1〉과 같다.

〈표 V-1〉 파이썬 활용 알고리즘 교육 프로그램 내용

차시	주제
1-7	• 오리엔테이션 • 사전 검사지 투입 • 파이썬 기본 기능 습득
9-14	• Print, Input 명령문 연습 • Random 명령문 연습 • 산술, 관계 명령문 연습
15-21	• 배웠던 내용 반복 연습 • List 명령문 연습
22-28	• 선형탐색에 대한 이해 • 이진탐색에 대한 이해
29-35	• 개별 프로젝트 과제 계획 • 개별 프로젝트 과제 수행
36-42	• 개별 프로젝트 과제 발표 • 사후 검사지 투입

파이썬과 같은 텍스트형 프로그래밍 언어는 처음 접하는 학생이 많기 때문에 파이썬의 기초 문법을 1일차에 집중적으로 학습할 수 있도록 구성했다. 2일차에는 다방면으로 활용이 가능한 랜덤 코드, 산술 코드, 관계 코드 등을 학습하도록 구성하고 3일차에는 그동안 배웠던 코드 등을 활용해보고 연습해보는 시간을 가지도록 구성했다. 또한 탐색 알고리즘의 기초가 되는 리스트를 학습하도록 하였다. 4일차에 탐색알고리즘 중에서 선형탐색과 이진탐색을 비교해보고 어떤 방식

으로 작동하는지에 대해서 살펴보고 학생들이 직접 코딩을 해보며, 두 탐색방법의 차이점과 수행시간 차이를 분석해보게 한다. 5일차에는 개별 프로젝트 학습법을 활용하여 학습자가 주도적으로 다양한 문제를 해결하고 과제를 수행하도록 하고 6일차에 발표 및 사후 검사를 하며 교육프로그램을 종료하게 구성하였다.

특히 알고리즘 수행시간 분석에 초점을 맞추어 1~3일차에 기본 문법, 조건문, 반복문 등을 학습할 때에도 어떤 방식으로 코딩을 해야 수행시간이 줄어들 지에 대해 고민해보고 같은 결과가 나오지만 수행시간이 다른 코드를 비교해보고 분석해보고 더 나은 방법은 없을 지 고민해보는 과정을 거치도록 지도하였다. 4~5일차에 이루어지는 두 가지 탐색프로그램(선형탐색, 이진탐색) 학습에서는 어떠한 상황에서 어떠한 방법이 더 효율적인 지에 대한 답을 스스로 찾아갈 수 있게 지도하였다. 또한 교육 참여자 대부분이 파이썬과 같은 텍스트형 프로그래밍 언어를 처음 접하였기 때문에 영어타자 연습을 틈틈이 실시하고 집으로 돌아간 뒤에도 매일 복습 및 예습 과제를 인터넷 카페에 업로드할 수 있도록 지도하여 집중적인 교육이 가능하도록 계획하였다.

다. 교재 개발

〔그림 Ⅱ-1〕 과 〔그림 Ⅱ-2〕, 〔그림 Ⅱ-3〕은 실제 사용한 교재를 발췌한 부분이다. 교육을 진행함에 있어서 기본적인 명령문을 학습할 때부터 지속적으로 수행시간에 대한 개념을 학습할 수 있도록 구성하였으며, 초등학생 수준을 고려하여 가장 기초적인 '선형탐색'과 '이진탐색'의 수행시간을 비교하는 활동을 넣어봄으로써, 학생들이 직접 최적의 경우와 최악의 경우, 그리고 탐색을 성공하는 평균을 계산해본다. 그리고 이를 파이썬을 활용하여 그래프로 그려보아 시각적으로 비교 및 분석을 한다.

〔그림 Ⅱ-1〕은 교육 프로그램 수업 중 기본적인 명령문에 있어서도 수행시간을 고려해야 한다는 점을 학습하는 부분이다.

◎ 반복문 이해하기

Q. ‘컴퓨터’를 100번 출력한다면?

print(‘컴퓨터’) print(‘컴퓨터’) print(‘컴퓨터’) . . . print(‘컴퓨터’) print(‘컴퓨터’) print(‘컴퓨터’)	<실행> ‘컴퓨터’ ‘컴퓨터’ ‘컴퓨터’ . . . ‘컴퓨터’ ‘컴퓨터’ ‘컴퓨터’

☞이 프로그래밍의 문제점은 뭘까요?

Q. while문을 이용해서 ‘컴퓨터’를 100번 출력하기 위한 알고리즘을 만들어 보시오.

i = 100
while ( ):
  print('컴퓨터')
  i=i-1
	<실행>
	‘컴퓨터’
	‘컴퓨터’
	‘컴퓨터’
	.
	.
	.
	‘컴퓨터’
	‘컴퓨터’
	‘컴퓨터’

〔그림 Ⅱ-1〕 교육프로그램 교재 예시 그림

〔그림 Ⅱ-1〕활동을 통해 같은 결과를 만들어내는 명령문이라고 하더라도 수행시간을 몸소 비교해보며 어떠한 명령문이 더 효율적인지에 대해서 스스로 생각해보고 수행시간의 중요성에 대해 반복적으로 깨달을 수 있도록 교재를 구성하였다.

〔그림 Ⅱ-2〕은 교육 프로그램 수업 중 탐색 방법에 따른 수행시간을 표로 비교해보는 활동이다. 실제 교육현장에서도 학생들이 이러한 유형의 교재내용을 해결해나가면서 정답을 찾는 방식이 아닌, 어떻게 하면 더욱 효율적으로 프로그래밍을 할 수 있을 것인지에 대해 시행착오를 겪는 현상을 발견할 수 있었다.

◎ 선형탐색과 이진탐색의 수행시간 비교

	선형탐색		
	최적의 경우	최악의 경우	평균
5개의 숫자 탐색	1번 째 탐색	5번 째 탐색	(1+2+3+4+5)/5 = 3번 째 탐색
100개의 숫자 탐색			
1000개의 숫자 탐색			
10000개의 숫자 탐색			

	이진탐색	
	최적의 경우	최악의 경우
5개의 숫자 탐색	1번 째 탐색	3번 째 탐색
100개의 숫자 탐색		
1000개의 숫자 탐색		
10000개의 숫자 탐색		

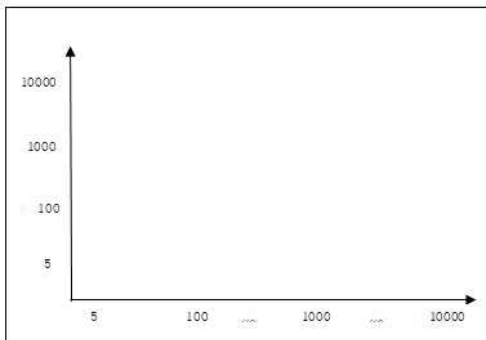
〔그림 Ⅱ-2〕 탐색방법에 따른 수행시간 비교

〔그림 Ⅱ-2〕 활동의 경우에는 선형탐색과 이진탐색이 각각 몇 번의 탐색으로 원하는 결과를 얻는지에 대하여 표를 그려보며 비교, 분석하게 된다. 학생들은 각

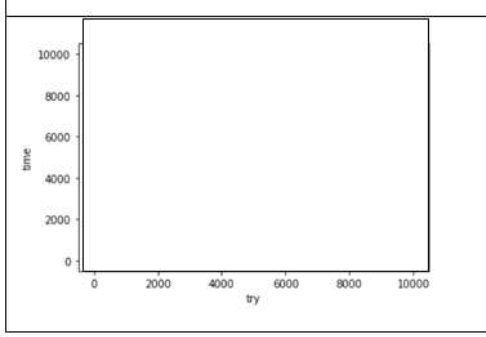
각의 탐색방법이 어떠한 원리로 작동하는 지 먼저 알 , 이에 따라 최적의 경우와 최악의 경우를 나누어서 생각해볼 수 있으며, 이 활동을 통해 선형탐색과 이진탐색의 장단점을 알 수 있다. 또한 어떠한 상황에서 어떠한 방법을 사용하는 것이 더 효율적인지에 대해 스스로 알 수 있었다.

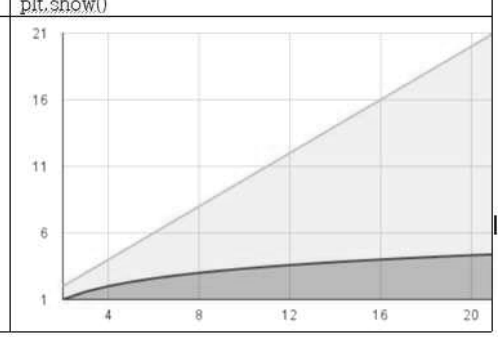
다음 〔그림 Ⅱ-3〕은 교육 프로그램 수업 중 탐색 방법에 따른 수행시간을 그래프로 비교해보는 활동이다.

◎ 선형탐색과 이진탐색의 수행시간 비교 그래프 그려보기




```
import matplotlib.pyplot as plt
import numpy as np
time = [5, 100,1000, 10000]
linear = [5, 100, 1000, 10000]
binary = [3, 7, 10, 14]
plt.plot(time, linear, 'b', label='linear')
plt.plot(time, binary, 'r', label='binary')
plt.legend(loc='best')
plt.ylabel("time")
plt.xlabel("try")
plt.title("linear&binary search time")
plt.show()
```





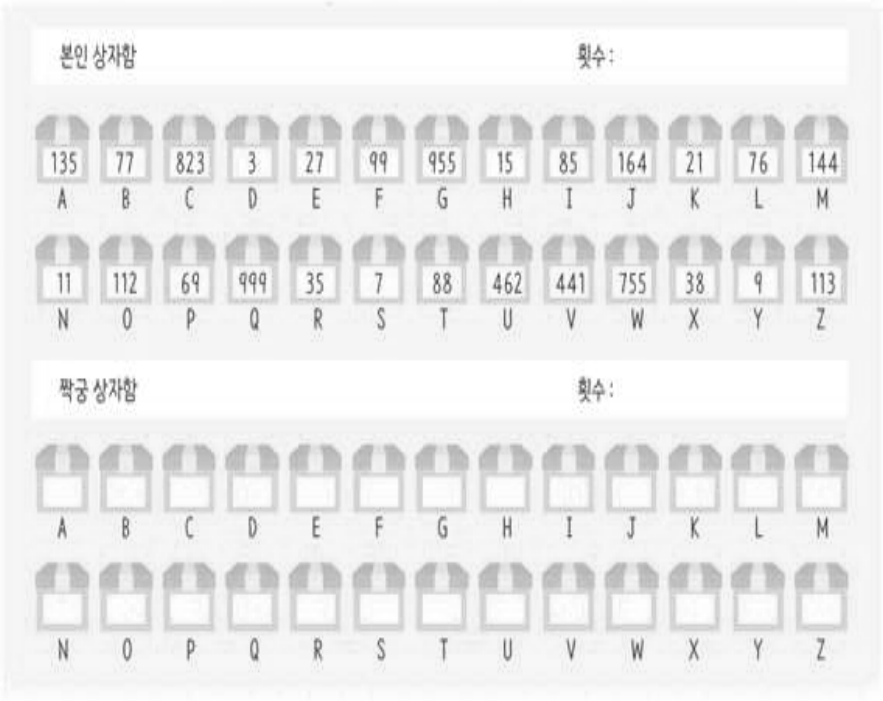
〔그림 Ⅱ-3〕 탐색방법에 따른 수행시간 분석 그래프

〔그림 Ⅱ-3〕 활동은 선형탐색과 이진탐색을 비교해보았던 표를 바탕으로 그래프를 그리게 된다. 표 속에 담긴 숫자보다는 시각적인 그래프로 표현하는 것이 '수행시간 분석'이라는 개념을 이해하는 데 더욱 효과적이기 때문이다. 초등학교 수

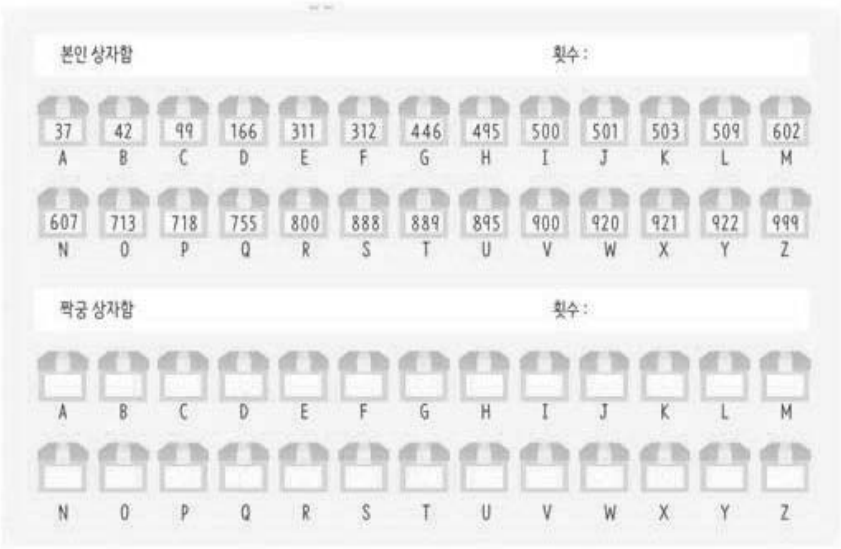
준에서 스스로 프로그래밍을 하여 그래프를 표현하는 것은 어려울 수 있기 때문에 미리 명령어를 제시해주었고, 그래프를 해석하는 데 초점을 두었다. 그래프를 해석하다보면 어떠한 탐색 방법이 더 효율적인지, 그 이유는 무엇인지에 대해서 학생 스스로 판단할 수 있게 된다. 이러한 일련의 과정 속에서 논리적 사고력 향상이 일어나게 된다.

〔그림 Ⅱ-4〕는 활동지를 통해 선형탐색과 이진탐색을 이해해보는 활동이다.

〈1회 A〉



〈1회 A〉



절 취 선

〈1회 B〉

〔그림 Ⅱ-4〕 탐색 알고리즘 수행시간 분석 활동지

짝과 함께 먼저 보물을 찾는 게임으로써 선형탐색일 경우에는 무작위로 선택해서 운이 좋으면 빨리 찾게 되고, 운이 나쁘면 늦게 찾는다는 것을 알 수 있게 된다. 반면 이진탐색의 경우에는 숫자가 오름차순으로 정렬되어 있다는 힌트를 통해 정답을 찾는 범위를 좁혀나갈 수 있다는 점을 깨닫게 된다. 그리고 보물을 몇 회째 찾았는지 확인해보면서 수행시간의 중요성을 인지할 수 있게 구성하였다. 학생들에게 '탐색알고리즘의 수행시간 분석' 이라는 개념을 조금 더 쉽고 재미있게 이해할 수 있는 활동이 되었다.

〈활동2〉

강 릉	거 위	고 전	교 육	기 억
나 비	노 인	다 리	더 위	도 로
두 목	라 면	루 비	리 듬	마 왕
머 루	명 주	무 대	바 위	봉 사
부 자	비 옷	사 랑	서 울	소 원
수 학	스 키	시 인	여 행	요 리
우 유	자 석	조 수	차 표	청 주
초 급	추 천	치 과	카 드	커 피
코 털	탄 생	텃 새	토 끼	피 자
허 락	호 두	효 과	휴 식	힌 트

〔그림 Ⅱ-5〕 탐색 알고리즘 수행시간 분석 활동지

다음 활동은 보물찾기 활동이 숫자로 이루어졌다고 하면 그림 1-6활동은 낱말로 이루어져있다. 또한 가나다 순서로 정렬되어 있으며 문제 출제자가 미리 생각해둔 정답을 최대한 적은 횟수로 맞히는 활동이다.

실제 교육현장에서 학생들은 수준별로 차이가 있었지만, 수행시간 중심이라는 개념에 대해서 인지하게 되었고, 문제해결에만 집중한 것이 아닌 더 효율적인 문제해결에 집중하는 경향을 보이게 되었다. 즉, 정답에 만족하는 것이 아니라 다른 방식으로 정답을 구해보려는 시도를 반복하고, 시행착오를 겪는 과정에서 학생들은 주요 개념들을 배우게 되었다.

4. 검사도구

검사도구는 논리적 사고 검사인 GALT를 사용하였다. GALT(Group Assessment of Logical Thinking; Roadrangka et al., 1983)은 1983년 Vantipa Roadrangka, Russell H.yeany, Michael J.Padilla가 공동으로 개발한 논리적 사고력 검사지이다.

총 문항은 21문항으로 각 문항에는 답과 그 이유를 선택할 수 있는 선다형 형식으로 6가지 논리 유형을 측정할 수 있게 구성되었다. 보존논리 4문항, 확률논리 2문항, 상관논리 2문항 그리고 조합논리는 3문항으로 대부분의 문항에는 문제 상황을 설명하는 짧은 문장과 그림이 제시되어있다. GALT개발자들은 시간과 같은 제약이 있을 경우를 대비해 각 논리 유형별로 2문항씩을 뽑아서 총 6유형 12문항으로 구성한 축소본 GALT로 구성하기도 했다. GALT 검사지는 각 논리 유형들이 태도를 비롯한 과학에 관련된 또 다른 변인을 구성하는 요인과의 관계를 확인하는 연구에서 이용하기 용이하며, 여러 선행연구들에서도 각 논리유형들과 학업성취도, 과학탐구능력, 등과 같은 또 다른 요인들과의 관계를 확인하기 위해 GALT를 이용한다.(유춘열・이용근, 2010)

본 연구에서는 한정된 교육시간으로 인하여 GALT검사지 축소본을 활용하여

논리적 사고 검사를 실시하였으며 〈표 Ⅵ-1〉는 각 문항이 측정하고자 하는 논리 유형을 나타낸 것이다.(심지현, 2018)

〈표 Ⅵ-1〉 논리적 사고 검사지 문항별 논리 유형

논리 유형	완본 문항	축소본 문항
보존 논리 (Conservation Reasoning)	1~4번	1, 2번
비례 논리 (Proprotional Reasoning)	5~10번	3, 4번
변인 통제 논리 (Controlling Varuavles Reasoning)	11~14번	5, 6번
확률 논리 (Probabilistic Reasoning)	15~16번	7, 8번
상관 논리 (Correlational reasoning)	17~18번	9, 10번
조합 논리 (Combinational Reasoning)	19~21번	11, 12번

5. 연구 설계 및 처치

본 연구에서는 도내 초등학생 4~6학년 25명을 대상으로 오리엔테이션, 사전•사후의 논리성 검사, 프로젝트 발표 등을 포함하여 총 6일 간 42차시 수업을 진행하였다. 교육은 외부 변인 통제가 보다 용이하도록 하기 위해서 학기 중 주말반 수업 형식을 지양하고 방학 기간에 집중교육의 형태로 실시하였다. 강사 1인이 전체 학습을 진행하고 보조 강사 2인이 학생들에게 도움을 주었다. 개인 별로 데스크탑 컴퓨터를 사용했으며, 파이썬을 별도로 설치하지 않고 웹페이지(Jupyter.org)에서 사용할 수 있도록 환경을 구축하였다.

교육 프로그램의 효과를 검증하기 위해 논리적 사고력 검사인 GALT를 이용하여 사전•사후검사를 실시하였으며 연구절차를 도식화 하면 〈표 Ⅶ-1〉와 같다.

〈표 Ⅶ-1〉 논리적 사고력 검사 실험 설계

	사전 검사	적용	사후 검사
실험집단(25명)	O1	X	O2

X : 탐색 알고리즘 기반 학습 실시
O1, O2 : 사전•사후 검사(GALT test)

Ⅳ. 연구 결과

1. 교육프로그램 효과 검증

파이썬의 탐색 알고리즘 기반 수행시간 SW교육으로 인해 학생들의 논리성에 어떠한 영향을 끼쳤는가를 분석하기 위한 검증을 실시하였다. 검증을 위하여 사용법이 쉽고 복잡한 자료를 편리하게 처리·분석할 수 있는 SPSS(Statistical Packages for Social Science)14.0 프로그램을 이용하였다.

가. 논리성 검사 정규성 검정

표본의 크기가 30명 이상을 만족하지 못하였기에 검증 전에 실험 집단의 논리성 검사 결과가 정규성을 만족하고 있는지를 정규성 검정을 통해 알아보았다. 정규성 검정은 보편적으로 실시하는 샤피로-윌크(Shapiro-Wilks) 검사를 실시하였고, 그 결과는 〈표 Ⅵ-1〉과 같다.

〈표 Ⅷ-1〉 정규성 검정 결과

하위 요소	실험 집단(N=25)				유의확률
	평균	표준편차	최댓값	최솟값	
보존논리	.3600	.8602	3	-1	.001**
비율논리	.5200	1.228	3	-2	.167
변인통제논리	.1600	.6879	2	-1	.000**
확률논리	.0400	1.059	2	-2	.056
상관논리	.1600	.5537	1	-1	.000**
조합논리	.4000	.7071	2	-1	.001**

*p〈.05 **p〈.01

정규성 검정을 한 결과, '보존논리', '변인통제논리', '상관논리', '조합논리'는 유의확률(p)이 각각 0.001, 0.000, 0.000, 0.001으로 나타나면서 귀무가설을 기각해서 정규성을 만족하지 않은 것으로 나타났다. 반면, '비율논리', '확률논리'에서는 유의확률(p)이 각각 0.167, 0.056으로 유의수준인 .05보다 크기에 귀무가설이 채택되어 정규분포임을 알 수 있었다.

나. 논리성 사전·사후 검사

사전·사후 검사 결과, 논리성의 변화를 알아보기 위해서 정규성을 확보한 요소들은 대응표본 t검정을 실시하고, 정규성을 확보하지 못한 요소들은 비모수검정 중 하나인 Wilcoxon의 부호-순위 검정을 실시하였다.

〈표 Ⅷ-2〉 논리성 사전·사후 검사 결과(대응표본 t검증)

하위 요소	학생수	사전 검사		사후 검사		t	유의확률
		평균	표준편차	평균	표준편차		
비율논리	25	2.56	1.895	3.08	2.178	-2.116	.045*
확률논리	25	1.16	.943	1.20	.764	-.189	.852

*p〈.05 **p〈.01

대응표본 t검정 결과, '비율논리'의 평균점수는 2.56에서 3.08로 0.52점 상승하였으며, 유의확률은 0.045로 평균점수에 있어 유의미한 상승이 있음을 알 수 있다. 반면에 '확률논리'의 경우 평균 1.16에서 1.20으로 0.04점 상승하였으나 유의확률이 0.854로 유의미하지 않은 것으로 나타났다.

〈표 Ⅷ-3〉 논리성 사전·사후 검사 결과(Wilcoxon's 검사)

하위 요소	학생수	사전 검사		사후 검사		t	유의확률
		평균	표준편차	평균	표준편차		
보존논리	25	2.88	1.054	3.24	.879	-2.000b	.046*
변인통제논리	25	1.48	1.418	1.64	1.497	-1.155b	.248
상관논리	25	.240	.436	.400	.577	-1.414b	.157
조합논리	25	1.20	.707	1.60	.645	-2.500b	.012*

*p〈.05, b. 음의 순위를 기준으로

본 〈표 Ⅷ-3〉에서는 정규성을 만족하지 못한 요소들에 대해 부호-순위 검정을

한 결과로 '보존논리'의 평균점수는 2.88에서 3.24로 0.36점 상승하였고 유의확률이 .046으로 나타났다. '조합논리'의 평균점수는 1.20에서 1.60으로 0.40점 상승하였으며 유의확률은 .012로 나타났다. 즉, '보존논리'와 '조합논리'에서는 평균점수가 유의미하게 상승했음을 알 수 있다. 반면, '변인통제논리'의 평균점수는 1.48에서 1.64로 0.16점 상승하고 '상관논리'의 평균점수는 0.24에서 0.4로 0.16점 상승하였으나 $p<.05$를 만족하지 못하여 유의미하지 않은 것으로 나타났다.

2. 연구 결과 분석

논리적 사고 검사(GALT) 결과 중 정규성을 확보한 논리성 요소들은 대응표본 t검정을 실시하였고, 정규성을 확보하지 못한 논리성 요소들에 대해서는 wilcoxon의 부호 순위 검정을 실시하였다.

정규성을 만족한 두 가지 논리성 요소에 대해 대응표본 t검정 결과 '비율논리'에서 유의미한 향상이 있었고, 정규성을 만족하지 못한 네 가지 논리성 요소에 대해 부호-순위 검정을 한 결과 '보존논리', '조합논리'에서 유의미한 향상을 보였다. 정리하자면 탐색 알고리즘을 기반으로 한 교육 프로그램을 통해 GALT 논리적 사고 검사의 논리유형에서 확률논리, 변인통제논리, 상관논리의 경우에는 평균점수는 향상되었으나 유의 확률을 충족하지는 못하였으나 비율논리, 보존논리, 조합논리에서는 유의미한 향상을 가져왔다.

왜 비율논리, 보존논리, 조합논리 부분에서 유의미한 향상이 일어났는지에 대해 추측해보면, 보존논리와 비율논리의 경우 파이썬의 명령어를 익히는 예제 속에 등호, 부등호, 총합, 나머지 등 다양한 연산을 활용했고, 수행시간에 대한 분석을 하는 과정에서 향상되었다고 본다. 조합논리 역시 반복, 리스트, 탐색알고리즘을 학습하면서 모든 경우를 빠짐없이 세거나, 중복하지 않도록 하는 과정을 반복하였기 때문에 향상되었다고 생각한다. 그 외 확률논리, 변인통제논리, 상관논리의 경우 알고리즘의 수행시간을 분석하는 데 있어 확률적 요소의 필요성이 상대적으로 적었던 것 같고, 변인통제가 필요한 예제 및 과제가 부족했던 것 같다. 상관관계의 경우 선형탐색과 이진탐색을 서로 비교·분석하면서 유의미한 향상이 있을 것으로 예상했으나, 4-5학년 학생의 경우 두 탐색방법의 상관성을 파악하는 것을 매우 어려워하였고 이에 영향을 미친 것 같다.

Ⅴ. 결론 및 제언

본 연구에서는 도내에 있는 초등학생을 대상으로 논리성을 향상시키기 위해 파이썬을 활용하여 탐색 알고리즘을 기반으로 한 교육 프로그램을 선보이고 ADDIE 모형의 개발 단계에 따라서 교육 프로그램을 개발하고 적용해보았다. 교육을 진행하면서 외부의 변인 통제가 보다 용이하고 연구의 신뢰성을 높이기 위하여 학기 중 주말반 수업 형식을 지양하고 방학 기간을 활용하여 총 6일, 42차시 집중 프로젝트 교육을 실시하였다. 또한 배운 내용 및 앞으로 배울 내용과 관련된 과제를 매일 제시하고 수행하도록 지도하여 짧은 기간이지만 밀도 있는 학습을 할 수 있게 프로젝트를 진행하였다. 논리성 사전 사후 검사를 해본 결과, 본 연구에서 개발한 교육 프로그램을 통한 SW교육이 초등학생의 논리성에 긍정적인 영향을 줄 수 있다는 것을 알 수 있었다.

하지만, 본 연구에서 실험집단은 상관연구에 필요한 30명 이상의 참여자를 투입하지 못하였기 때문에 일반화하는 데에는 한계가 있다. 그리고 본 연구에서 개발한 교육 프로그램에 비교집단 없이 실험집단에만 투입하였기 때문에 논리성에 긍정적인 영향을 미친다는 점이 본 연구에서 개발한 교육 프로그램의 영향 때문인지 상관관계를 분석할 수 없다는 문제가 있다. 또한 한정된 시간으로 인하여 탐색 알고리즘 교육에서 선형 탐색과 이진 탐색만을 대상으로 수행시간 비교 및 분석을 하였는데, 학생들이 직접 구현에 있어 한계점을 가지고 있었다. 추후의 연구에서는 30명 이상 다수의 참여자를 대상으로 실험집단과 비교집단을 구성하고 연구결과에 대해서 각 요인들 간의 상관관계를 비교, 분석할 필요가 있다.

참고문헌

강자영. (2018). **초등학교 소프트웨어 교육이 논리적 사고력에 미치는 영향 분석**. 목포대학교 교육대학원.

고준강. (2018). **3D 모델링을 이용한 초등 소프트웨어 교육 프로그램 개발**. 서울교육대학교 교육전문대학원.

교육부. (2015). **교육부 소프트웨어 교육 정책**.

곽병선. (1995). **논리력 신장을 위한 CAI프로그램 연구 개발**. 서울: 한국교육개발원.

김경미. 김현숙. (2014). 융합인재 양성을 위한 컴퓨터 프로그래밍 교육의 필요성에 대한 사례연구. **한국디지털정책학회 논문지, 12**(11). 339-348.

김종훈. (2015). **순서도 & C언어**. 서울: 다올미디어.

김종훈. 김종진. (2013). **컴퓨터개론**. 서울: 한빛아카데미.

김진동. 양권우. (2010). 실생활 속 사례를 통한 알고리즘 학습이 논리적 사고력에 미치는 영향. **정보교육학회 논문지, 14**(4). 555-560.

김한성. (2016). 소프트웨어 교육 운영지침 개발을 위한 방향 탐색. **학습자중심교과교육학회 논문지, 16**(8). 529-548.

류춘열・이용근. (2010). 일반화가능도 이론을 이용한 집단논리적사고력검사(GALT)의 신뢰도 분석. **한국지구과학회 학술저널, 31**(1). 95-105.

문미예・김갑수. (2018). 초등학생을 위한 Python 프로그래밍 언어 교육방안 연구. **정보교육학회 학술논문집, 9**(1). 33-41.

미래창조 과학부. (2014). **ICT and Future Planning**.

박대륜. (2018). **초등학생을 위한 로봇 활용 파이썬 학습 프로그램 개발**. 대구교육대학교 교육대학원.

박응용. (2001). **Jump To Python**. 정보게이트.

서성원. (2010). **TPL과 VPL을 활용한 로봇프로그래밍 교육이 정보과학적 사고 능력에 미치는 영향**. 한국교원대학교 교육대학원.

성정숙. 김현철. (2015). 국외 컴퓨터 교육과정의 변화 분석. **컴퓨터교육학회**

논문지, 18(1). 45-54.

소홍렬. (1982). **논리와 사고**. 이화여자대학교.

심지현. (2018). **논리적 사고력 향상을 위한 놀이 중심의 언플러그드 프로그램 개발 및 적용**. 공주교육대학교 교육대학원.

유인환. (2018). 파이썬과 로봇을 활용한 초등학교 SW 교육의 설계. **정보교육학회 학술논문집, 9**(1). 149-155.

유진아. (2008). **공개소프트웨어 Python을 이용한 프로그래밍 교육에 관한 연구**. 단국대학교 교육대학원.

이민영. (2016). **엔트리와 스크래치를 활용한 초등학생의 논리적 사고력 신장에 관한 연구**. 서울교육대학교 교육전문대학원

전산용어사전 편찬위원회. (2005). **컴퓨터 인터넷・IT용어 대사전**. 서울: 일진사.

정구원. (2015). **마이크로컨트롤러 교육을 위한 비주얼과 텍스트 프로그래밍 언어 적용 분석**. 서울교육대학교 교육전문대학원.

정인기. (2002). 탐색 알고리즘 교육을 위한 S/W 컴포넌트의 개발. **정보교육학회논문지, 6**(2). 179-186.

Python Software Foundation. (2017). Python about. https://www.python.org/about/

Roadrangka, V., Yeany, R.H., and Padilla. M.J. (1983). The construction and validation of group assesment of logical thinking(GALT). Paper presented at the Annual Meeting of the National Association for Research in Science Teaching, 5p.

Torrance, E. P. (1978). Giftedness in solving furture problems. **Journal of Creative Behavior, 12**(2). 75-86.

Wing, J. M. (2008). Computational thinking and thinking about computing. **Royal Society of London Philosophical Transactions, 366**(1881). 3717-3726.

ABSTRACT

Development and application of Search algorithm execution education program for elementary school students using Python

Kong, gipyo

Major in Elementary Computer Education
Graduate School of Education Jeju National University

Supervised by Professor Kim, jonghoon

This study developed and applied a Python software education program focused on search algorithm execution time to improve the logical thinking of elementary school students. This educational program was developed based on the process of ADDIE model, utilizing the results of pre-demand analysis conducted for 133students in elementary school in Jeju. In order to verify the effectiveness of the developed SW educational program, 25 students who participated in the education donation program at ○○ University conducted 42hours of classes during 6days. The GALT test was used to analyze the educational effects of the pre- and post-test. According to the results of the analysis, it was found that the educational program developed in this study can positively influence the logical thinking of the elementary school students.

Keywords : Python, Search algoritm, Software education, logical thinking

* A thesis submitted to the committee of Graduate School of Education, Jeju National University in partial fulfillment of the requirements for the degree of Master of Education conferred in August, 2019.

부록

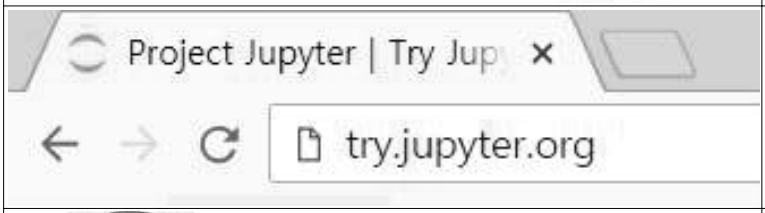
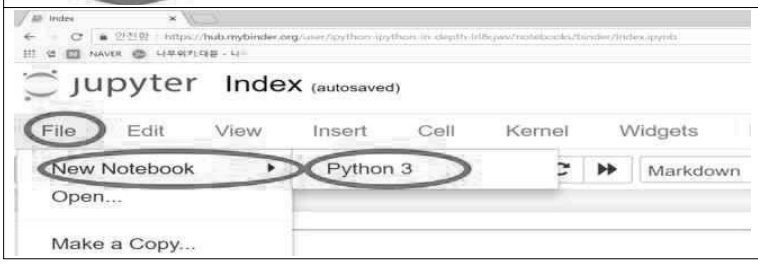
〔부록 1〕 교육 프로그램 교재

〔부록 2〕 교육 프로그램 과정안

제0강 파이썬의 기초

파이썬의 세계에 오신 걸 환영합니다.

파이썬 시작하기

	**Step1.** 구글 크롬 브라우저를 설치합니다.
	**Step2.** 주소창에 try.jupyter.org를 입력합니다.
	**Step3.** 빨간색 동그라미로 표시된 'try jupyter with Python'을 클릭합니다,
	**Step4.** File탭에서 New Notebook Python을 클릭합니다.

들여쓰기는 코드를 읽기 쉽도록 일정한 간격을 띄워서 작성하는 방법입니다. 특히 파이썬은 들여쓰기 자체가 문법입니다. 예를 들어 if의 다음 줄은 항상 들여쓰기를 해야 합니다.

```
if a == 10:
print('10입니다.')     # 들여쓰기 문법 에러
```

```
실행결과
IndentationError: expected an indented block
```

올바른 코드는 다음과 같이 if의 다음 줄은 들여쓰기를 해주어야 합니다.

```
          if a == 10:
공백 4칸  ␣␣␣␣print('10입니다.')
```

단, 같은 블록은 들여쓰기 칸 수가 같아야 하고, 공백과 탭을 섞어 쓰면 안 됩니다.

```
          if a == 10:
공백 2칸  ␣␣print('10')
공백 4칸  ␣␣␣␣print('입니다.')                    ✗

          if a == 10:
공백 4칸  ␣␣␣␣print('10')
 탭 1칸   ├──→print('입니다.')                    ✗

          if a == 10:
공백 4칸  ␣␣␣␣print('10')
공백 4칸  ␣␣␣␣print('입니다.')                    ○
```

Q. 다음 다섯 가지 표현 중 정상적으로 값이 출력 되는 것을 찾아보세요.

```
1) print("I love you")
2) print("I like you')
3) print('Korea')
4) print{Hello}
5) print[Hello]
```

제 1 강 파이썬 기초문법

◎ 연산자 기호

	<실행>
10+ 4	14
10-4	6
10*4	40
10/4	2.5
10//4	2
10%4	2
10**4	10000

Q. 실행결과를 보고 연산 기호를 정리해봅시다.

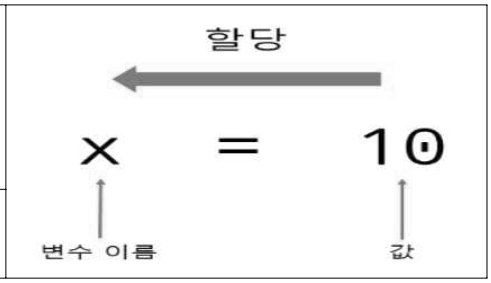
+		//	
-		%	
*		**	
/			

◎ 변수

x = 10이라고 입력하면 10이 들어있는 변수 x가 만들어집니다. 즉, '변수이름 = 값' 형식이죠. 이렇게 하면 변수가 생성되는 동시에 값이 할당(저장)됩니다.

```
x = 10
print(x)      # print로 변수의 값 출력
```

<실행>
10



Q. 실행결과로 알맞은 것은 무엇일까요?

```
korean = 90
english = 90
mathematics = 80
science = 80
average = (                                    )
print("평균점수:", average)
```

<실행>
평균점수:85

Q. 두 개의 변수에 자신의 몸무게, 키를 저장하고 합을 구하는 프로그램을 만드시오.

◎ 입력

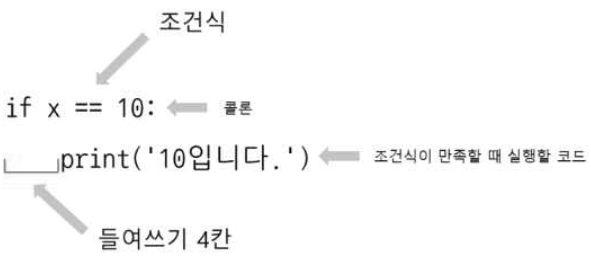
input은 표준 입력을 받는 함수이며 주로 입력 값을 변수에 저장해서 사용합니다.

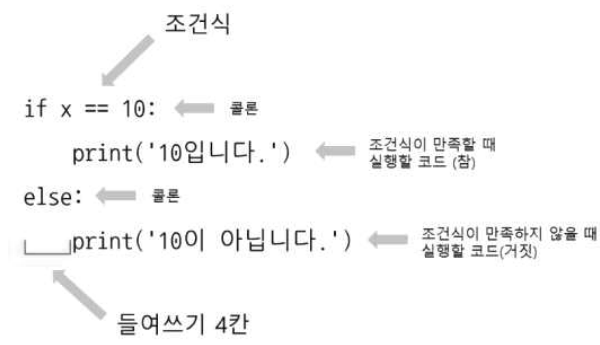
s = input('문자열을 입력하세요: ') print(s)	<실행> 문자열을 입력하세요: 안녕 <-입력 안녕

Q. 세 개의 변수에 자신의 나이와 몸무게 키를 입력하고 더한 뒤 출력해보는 프로그램을 만드시오. [int 설명]

◎ 조건문 이해하기(if, else, elif)

조건문은 특정 조건일 때 코드를 실행하는 문법입니다.

	조건식은 x == 10과 같은 형식으로 지정해 주는데 ==는 두 값이 "같을 때"라는 뜻이며 수학의 =(등호)와 같습니다. 즉, if x == 10:은 x가 10과 같은지 비교한 뒤 같으면 다음에 오는 코드를 실행하라는 뜻이 됩니다.

	x가 10일 경우에는 '10입니다.'를 출력하고 x가 10이 아닐 경우에는 '10이 아닙니다.'가 출력됩니다.

Q. 두 수 중 큰 수를 찾는 프로그램을 완성하시오.

```
a= int(input('정수:'))
b= int(input('정수:'))
if (                    ):
    print('큰 수:', a)
else:
    print('큰 수:', b)
```

```
<실행>
정수: 15 <-입력
정수: 20 <-입력
큰 수: 20
```

Q. 다음 알고리즘을 보고 놀이기구 탑승 조건에 대한 설명을 쓰시오.

설명: (_____)

```
소스 :
height = int(input("키: "))
if height<120 or height>=200:
 print('제한')
else:
 print('통과')
```

```
<실행>
키: 155 <-입력
통과
```

Q. 놀이공원 입장료 구하기 알고리즘을 완성하시오.

8살 미만, 60세 이상은 무료이고, 그 이외는 10000원의 입장료인 놀이공원

```
a=int(input('나이:'))
if (                              ):
    print('무료')
else:
   print('10000원')
```

```
<실행>
나이: 13 <-입력
10000원
```

Q. 입력된 세 가지 수의 최댓값을 출력하는 알고리즘을 완성하시오.

```
<방법1>
a= int(input('정수'))
b= int(input('정수'))
c= int(input('정수'))
if a>b:
        if a>c:
            print('가장 큰 수:', a)
        else:
            print('가장 큰 수:', c)
else:
        if b>c:
            print('가장 큰 수:', b)
```

```
        else:
            print('가장 큰 수:', c)
```

<방법2>

```
a= int(input('정수'))
b= int(input('정수'))
c= int(input('정수'))
print(max(a,b,c))
```

*max: 최댓값 함수

☞ **<방법1>과 <방법2>의 차이점은 무엇일까요?**

Q. 입력된 세 가지 수의 최솟값을 출력하는 알고리즘을 완성하시오.

제2강 반복 구조 - 수행시간 분석 예제

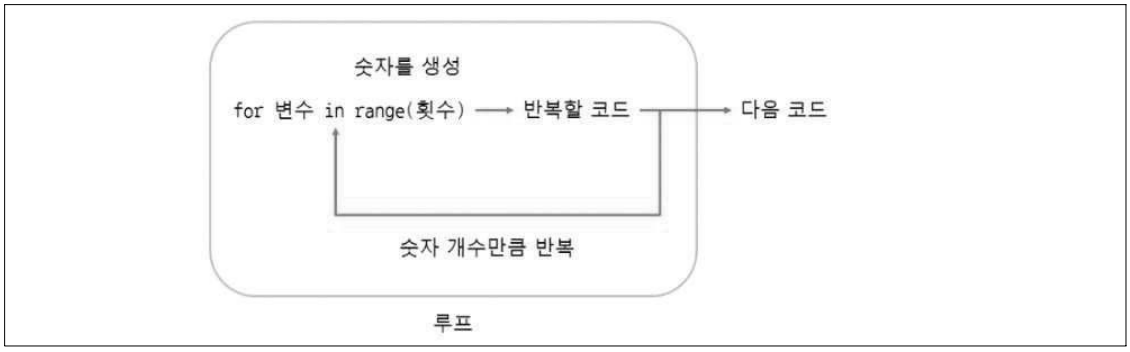
◎ 반복문 이해하기

Q. '컴퓨터'를 100번 출력한다면?

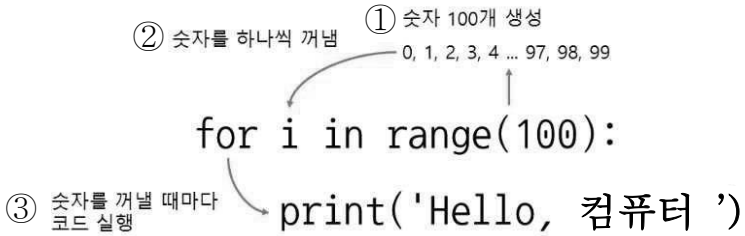
print('컴퓨터') print('컴퓨터') print('컴퓨터') . . . print('컴퓨터') print('컴퓨터') print('컴퓨터')	<실행> '컴퓨터' '컴퓨터' '컴퓨터' . . . '컴퓨터' '컴퓨터' '컴퓨터'

☞이 프로그래밍의 문제점은 뭘까요?

◎for 반복문의 원리



Q. for문을 이용해서 '컴퓨터'를 100번 출력하면 어떤 점이 좋은 지 생각해보세요.



<실행>
'컴퓨터'
'컴퓨터'
'컴퓨터'
.
.
.
'컴퓨터'

Q. 1부터 10까지 출력하기(for를 사용한 반복)

```
for i in range(10):
  print(i+1, end=' ')
```

[실행]
1 2 3 4 5 6 7 8 9 10

```
for i in range(1, 11, 1):
  print(i, end=' ')
```

[실행]
1 2 3 4 5 6 7 8 9 10

range(5): 0, 1, 2, 3, 4	range(a)는 0부터 'a-1'까지의 수
range(1, 5, 2): 1, 3, 5, 7	range(a, b, c)은 'a'부터 'b-1'까지 'c'만큼 수

[연습] 각 범위에 해당하는 숫자를 적어보세요.
range(4):
range(0, 4):
range(1, 5, 2):

Q. 10부터 5까지 출력하는 프로그램을 작성하시오.

Q. for문을 이용해서 1부터 10까지의 합 구하기

```
sum= (      )
for i in range(1, _____ ):
    sum=sum+(_____)
print(sum)
```

<실행>

Q. for문을 이용해서 1부터 100까지의 합을 구해보세요.

```
sum= (      )
for i in range(1, _____ ):
    sum=sum+(_____)
print(sum)
```

<실행>

Q. for문을 이용해서 1부터 100까지의 수 중 홀수의 합을 구해보는 식을 완성하고, 어떤 알고리즘이 효율적인지 생각해보세요.

sum=( )
for i in range(1, 100):
  if i%2==1:
  sum=sum+ i
print(sum)
	<실행>
	2500

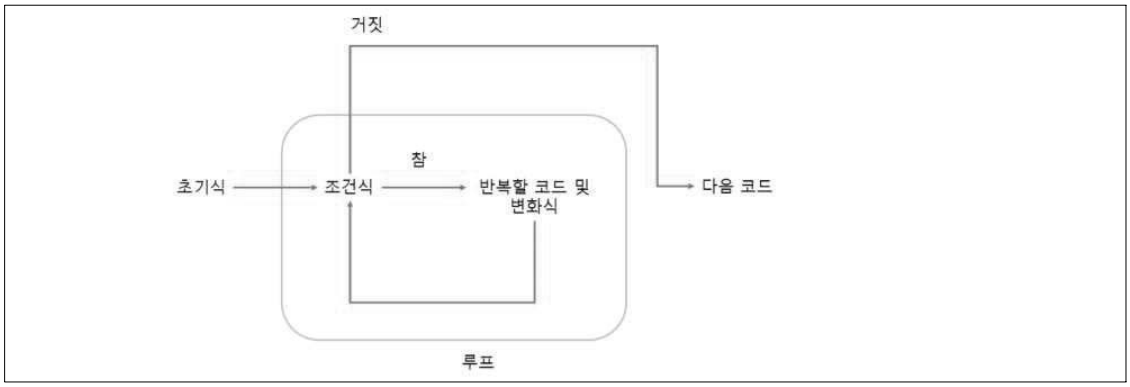
sum= ( ) for i in range(1, _____): sum=sum+ (_____) print(sum)	

Q. for문을 이용해서 1부터 100까지의 수 중 짝수의 합을 구해보는 식을 완성해보고 실행결과를 구해보시오.

sum=( )
for i in range(1, 101):
  if i%2==0:
  sum=sum+ i
print(sum)
	<실행>

sum= ( ) for i in range(___, _____): sum=sum+ (_____) print(sum)	

◎ while 반복문 원리



Q. 1부터 10까지 출력하기(while을 사용한 반복)

```
i = 1
while i<=10:
   print(i, end=' ')
   i=i+1
```

[실행]
1 2 3 4 5 6 7 8 9 10

```
i = 0
while _____:
   i=i+1
   print(i, end=' ')
```

[실행]
1 2 3 4 5 6 7 8 9 10

```
while 조건:
      문장
```

while은 조건이 참인 동안 아래 문장을 반복해서 실행한다.

Q. 10부터 5까지 출력하는 프로그램을 작성하시오.

Q. while문을 이용해서 '컴퓨터'를 100번 출력하기 위한 알고리즘을 만들어 보시오.

```
i = 100
while (        ):
     print('컴퓨터')
     i=i-1
```

<실행>
'컴퓨터'
'컴퓨터'
'컴퓨터'
.
.
.
'컴퓨터'
'컴퓨터'
'컴퓨터'

Q. while문을 활용하여 다음 실행결과가 나오도록 알고리즘을 만들어 보시오.

```
num = 0
while (                    )
     if num % 2 == 1:
          print(num)
     num = num+1
```

<실행>
1
3
5
7
9

Q. while문을 이용해서 1부터 10까지의 합 구하기

```
sum= (        )
```

<실행>

a=1
while a<= ( ):
  sum=sum+ a
  a=a+ 1
print(sum)	

Q. <u>while문을 이용해서</u> 1부터 100까지의 합을 구해보세요.

Q. 숫자 맞추기 게임(random의 활용)

```
import random
n = random.randint(1, 10)
while True:
    a = int(input("이 숫자는?: "))
    if a == n:
        print("정답")
        break
    if a < n:
        print("UP")
    if a > n:
        print("DOWN")
```

```
[실행]
이 숫자는?: 5
DOWN
이 숫자는?: 3
UP
이 숫자는?: 4
정답
```

while True: 문장	조건이 참인 동안 문장을 계속 실행 → 영원히 반복
break	반복을 멈추어 주는 명령어
단순하게 정해진 횟수를 반복할 때는 for 명령어가 유용하지만, 반복을 계속할지 중단할지 판단하거나 무한 반복을 해야 할 때는 while 명령어가 유용하다.	

<정리>

같은 문항을 해결할 때, 같은 결과가 나오더라도 어떠한 알고리즘을 사용하느냐에 따라서 그 모양도 (같고 / 다르고) 걸리는 수행시간도 (같아질 / 달라질) 수 있습니다. 따라서 프로그래머는 ㅎㅇㅈ인 알고리즘을 구성하는 것이 중요합니다. 교제에 있는 예시 알고리즘 이외에도 다른 알고리즘으로는 해결할 방법은 없는지, 있다면 어떤 것이 더욱 효율적인가를 찾아보는 것이 매우 중요합니다.

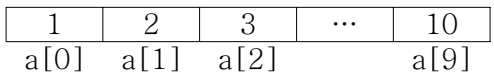
제3강 리스트와 탐색

◎리스트란?

비슷한 특성을 가진 자료들을 연결해 놓은 것을 **리스트**라고 합니다. 일반적인 변수에는 1가지 값밖에 저장하지 못하지만, 리스트에는 여러 개의 데이터를 저장할 수 있습니다.

```
a = []
for i in range(1, 11):
    a.append(i)
print(a)
```

[실행]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]



a[0] a[1] a[2] a[9]

* append란?
리스트에 항목을 추가하는 것으로 'a append(i)' a라는 리스트에 i를 추가한다는 의미함.

◎리스트 요소 선택하기

```
a = ['a', 'b', 'c', 'd']
print(a[1])
print(a[-1])
print(a[:2])
print(a[2:])
print(a[1:3])
print(a[1:-1])
```

[실행]
b
d
['a', 'b']
['c', 'd']
['b', 'c']
['b', 'c']

Q. 위 리스트에서 ['b', 'c', 'd']가 출력되도록 명령어를 작성해 보시오.

◎리스트 수정하기

```
a = [‘a’, ‘d’, ‘c’]
a[1] = ‘b’
print(a)
----------------------------------
-------
del(a[1])
print(a)
----------------------------------
-------
a.insert(1, ‘b’)
print(a)
----------------------------------
--------
a.remove(‘b’)
print(a)
```

[실행]
['a', 'b', 'c']

['a', 'c']

['a', 'b', 'c']

['a', 'c']

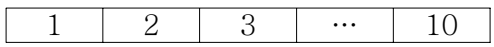
Q. 위 리스트의 마지막에서 ‘a’만 남도록 명령어를 작성해 보시오.

◎리스트 활용하기

Q. 1부터 10까지의 수를 리스트에 저장하고 10부터 1까지 거꾸로 출력해보시오.

```
a = []
for i in range(1, 11):
    a.append(i)
for i in range(9, -1, -1):
    print(a[i], end=’ ‘)
```

[실행]
10, 9 , 8, 7, 6, 5, 4, 3, 2, 1



Q. 리스트 항목을 거꾸로 뒤집어 출력해보시오.

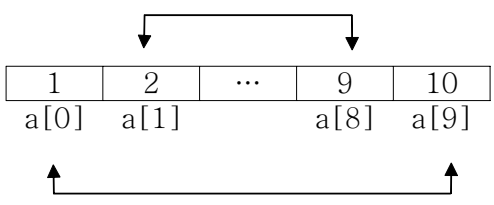
<방법1>

```
a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
for i in range(0, 5):
    temp=a[i]
    a[i]=a[9-i]
    a[9-i]=temp
print(a)
```

<방법2>

```
a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
a.reverse()
print(a)
```

[실행]
[10, 9 , 8, 7, 6, 5, 4, 3, 2, 1]



☞ 2가지 방법 중 더 효율적인 것은 무엇일까요?

Q. a = [5, 10, 7, 1, 3]라는 리스트를 작은 수 순서대로 정렬하시오.

```
a = [15, 10 ,7, 1, 3]
a.sort()                                    *a.sort() = 작은 수 순서대로 정렬
print (        )
```

<실행>
12

Q. a = [10, 4, 67, 92, 54, 12, 2, 79, 55, 43, 82, 15] 라는 리스트에서 3번째로 작은 숫자를 출력하시오. 빈칸에 알맞은 알고리즘을 쓰시오.

```
a = [10, 4, 67, 92, 54, 12, 2, 79, 55, 43, 82, 15]
a.sort()                                    *a.sort() = 작은 수 순서대로 정렬
print (        )
```

<실행>
12

Q. 리스트 속 0~100사이 랜덤 숫자 10개의 총합을 구하는 프로그램을 만들어보시오.

```
import random
a= [random.randint(0,100) for x in range(10)]
print (a)
sum = 0
for  i in a:
    sum = i +  sum
print(sum)
```

<실행>
[42, 38, 53, 20, 84, 36, 37, 26, 70, 9]
415

Q. 리스트 속 중국집 메뉴들을 무작위로 선택하는 프로그램을 만들어보시오.

답안예시)

```
import random                                   *import random = 무작위모듈
food = ['짜장면', '짬뽕', '볶음밥', '간짜장', '짬뽕밥', '짜장밥']
select= (random.choice(food))
print(select)
```

<실행>
짬뽕

Q. 친구들의 이름을 랜덤하게 출력하는 프로그램을 만들어보시오.

답안예시) import random friends = [‘mingyu’, ‘minwoo’, ‘minbeom’, ‘gipyo’] random.shuffle(frineds) print(friends[0])	<실행> gipyo
답안예시) import random friends = [‘mingyu’, ‘minwoo’, ‘minbeom’, ‘gipyo’] select = random.choice(frineds) print(select)	<실행> gipyo

Q. 리스트를 이용한 퀴즈 만들기

```
pro = ['동물의 왕은?', '새 중의 왕은?', '바다의 왕은?']
answer = ['사자', '독수리', '고래']
for i in range(len(pro)):
    ans = input(pro[i])
    if ans == answer[i]:
        print('정답!')
    else:
        print('땡')
```

[실행]
동물의 왕은?사자
정답!
새 중의 왕은?독수리
정답!
바다의 왕은?뱀
땡

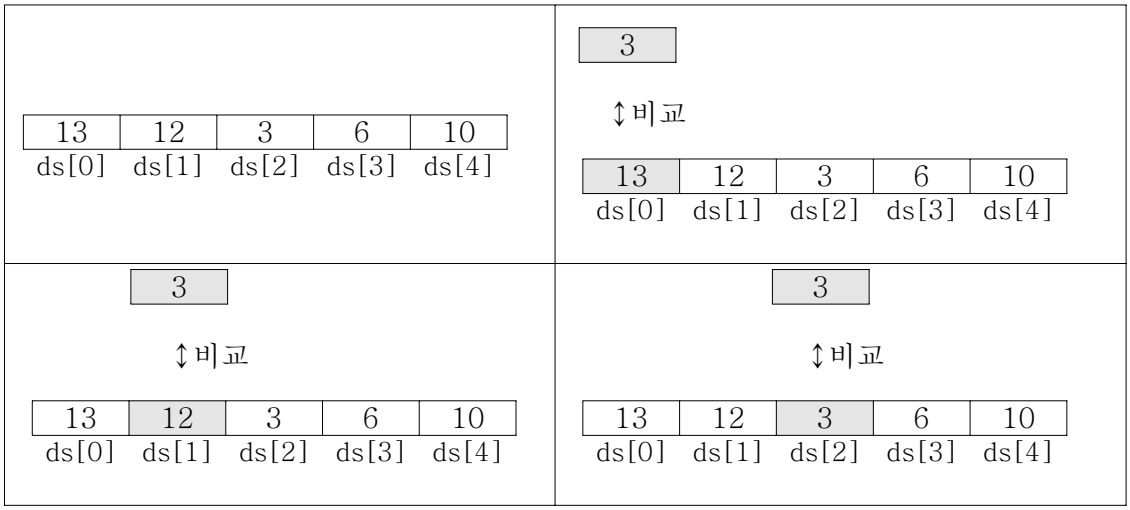
Q. 무작위모듈을 이용하여 무작위로 선택하는 또 다른 프로그램을 만들어보시오.

Q. 리스트 탐색 프로그램을 완성하시오.

```
a=[x*10 for x in range(1,11)
key=int(input('키:'))
if key in a:
    print(a.index(key), '에서 탐색')
else:
    print('탐색 실패')
```

```
<실행>
키:90 <-입력
8에서 탐색
```

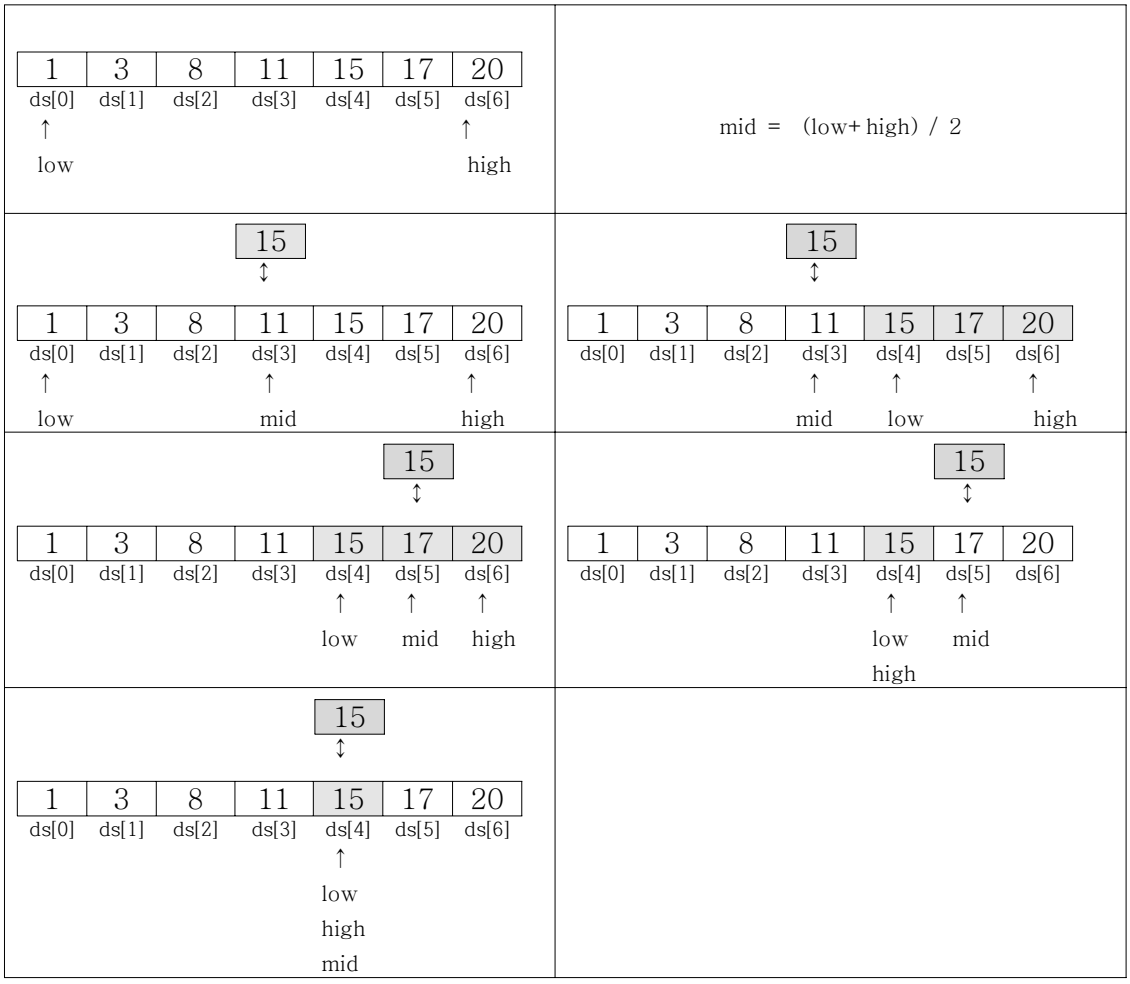
◎선형탐색의 원리




```
ds=[13, 12, 3, 6, 10]
found= (            )
key=1
for a in range(0,           ):
    if key==ds[a]:
        found=1
        break
if found==1:
    print(a)
else:
    print('없음')
```

```
<실행>
2
```

◎이진탐색




```
ds=[1, 3, 8, 11, 15, 17, 20]
found=0
key=15
low=0
high=len(ds)-1
while (                    ):
    mid=(low+ high)//2
    if key==ds[mid]:
        found=1
        break
    elif key<ds[mid]:
        high=(              )
    else:
```

<실행>

4

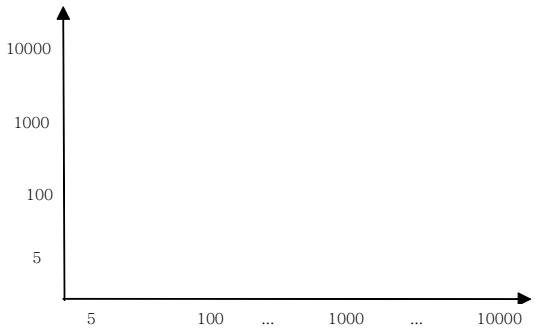
low=( )
if found==1:
  print(mid)
else:
  print('없음')	

◎ 선형탐색과 이진탐색의 수행시간 비교

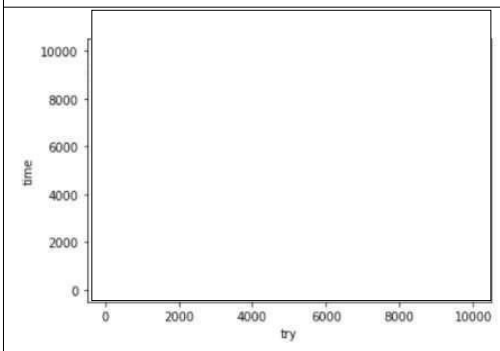
	선형탐색		
	최적의 경우	최악의 경우	평균
5개의 숫자 탐색	1번 째 탐색	5번 째 탐색	(1+ 2+ 3+ 4+ 5)/5 = 3번 째 탐색
100개의 숫자 탐색			
1000개의 숫자 탐색			
10000개의 숫자 탐색			

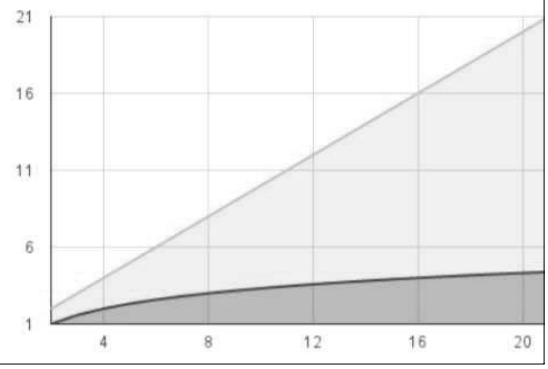
	이진탐색	
	최적의 경우	최악의 경우
5개의 숫자 탐색	1번 째 탐색	3번 째 탐색
100개의 숫자 탐색		
1000개의 숫자 탐색		
10000개의 숫자 탐색		

◎ 선형탐색과 이진탐색의 수행시간 비교 그래프 그려보기




```
import matplotlib.pyplot as plt
import numpy as np
time = [5, 100,1000, 10000]
linear = [5, 100, 1000, 10000]
binary = [3, 7, 10, 14]
plt.plot(time, linear, 'b', label='linear')
plt.plot(time, binary, 'r',
label='binary')
plt.legend(loc='best')
plt.ylabel("time")
plt.xlabel("try")
plt.title("linear&binary search time")
plt.show()
```





제4강 자율주제 탐구

2018 파이썬 알고리즘 개인별 프로젝트 최종과제 계획서

(　　)초등학교　(　　)학년　이름 (　　)

1. 주제:

(ex. 이름을 가지고 탐색하기, 도서관 책 탐색하기, 성적 탐색하기, 전화번호부 탐색하기 등.)

2. 최종과제 설명:

3. 알고리즘 기초 설계 :

4. 최종과제 프로그래밍 (코딩)

5. 최종과제 실행 결과

〔부록 2〕

<table>
<tr><td>일시</td><td>2018.8.13.-2018.8.14.</td><td>대상</td><td>파이썬 B반</td></tr>
<tr><td>주제</td><td>파이썬 기초 기능</td><td>차시</td><td>1~11/36</td></tr>
<tr><td>활동명</td><td>파이썬의 기초 기능 익히기</td><td>소요시간</td><td>550분</td></tr>
<tr><td>학습목표</td><td colspan="3">파이썬의 기초기능을 익히고 사용할 수 있다.</td></tr>
<tr><td>학습
요소</td><td colspan="2">교수·학습 활동</td><td>차시</td></tr>
<tr><td>도입</td><td colspan="2">오리엔테이션 및 학습내용 안내
동기유발 활동
기본학습 능력 및 논리성 검사를 위한 GALT test실시</td><td>1~3</td></tr>
<tr><td rowspan="3">전개</td><td colspan="2">●파이썬이란?
- 텍스트형 프로그래밍 언어 중 하나
- 파이썬 접속 방법 및 기초 문법 설명
●파이썬 기초문법1
1. 들여쓰기 문법
2. 공백 사용법
3. 출력 방법</td><td>4~5</td></tr>
<tr><td colspan="2">● 파이썬 기초문법2
1. 연산자 기호 사용법
2. 변수 사용법
3. 입력 구문
-다양한 문제 및 퀴즈를 통해 문법 익히기
-간단한 프로그램 모방 및 제작</td><td>5~7</td></tr>
<tr><td colspan="2">● 파이썬 기초 문법 실습
-배운 유형 총 복습
-과제 제시 및 피드백</td><td>8~10</td></tr>
<tr><td>정리</td><td colspan="2">● 학습한 내용 복습 및 정리
● 다음 차시 배울 내용 문제와 퀴즈 제시</td><td>11</td></tr>
</table>

일시	2018.8.15.	대상	파이썬 B반
주제	파이썬 조건문 & 반복문	차시	12~17/36
활동명	파이썬 조건문 & 반복문 익히기	소요시간	300분
학습목표	파이썬 조건문 & 반복문을 익히고 사용할 수 있다.		

학습 요소	교수·학습 활동	차시
도입	학습내용 안내 동기유발 활동 파이썬 기초 기능 복습	12
전개	**●조건문이란?** - 특정한 조건일 때 코드를 실행하는 문법 **●조건문의 종류** 1. If 2. else 3. elif	13
	●조건문의 관련 문제 제시 1. 큰 수를 찾는 프로그램 제작 2. 놀이기구 탑승 조건을 보고 탑승 여부 판단하는 프로그램 제작 -다양한 문제 및 퀴즈를 통해 문법 익히기 -간단한 프로그램 모방 및 제작	14
	● 반복문 이해하기 -컴퓨터를 100번 출력해보기 -더 쉬운 방법 혹은 더 짧은 코드로 나타낼 수 없을 지 생각해보기 **● 반복문의 종류** 1. for 반복문 2. while 반복문 두 반복문의 차이점을 코딩을 통해 알아보기	15~16
정리	**● 학습한 내용 복습 및 정리** **● 다음 차시 배울 내용 문제와 퀴즈 제시**	17

일시	2018.8.16.	대상	파이썬 B반
주제	파이썬 반복문 & 리스트	차시	18~23/36
활동명	파이썬 반복문 &리스트 익히기	소요시간	300분
학습목표	파이썬 반복문& 리스트를 익히고 사용할 수 있다.		

학습 요소	교수·학습 활동	차시
도입	학습내용 안내 동기유발 활동 파이썬 조건문 & 반복문 복습 및 학습지를 통한 전시학습 상기	18
전개	**●반복문 원리 알아보기** - for 반복문과 while 반복문 원리 알아보기 **●반복문 관련 과제 풀어보기** 1. 10부터 5까지 출력하는 프로그램 제작 2. 1 3 5 7 9가 출력되도록 하는 프로그램 제작 3. 숫자 맞추기 게임 프로그램 제작	19
	●반복문과 수행시간 분석의 관계 알아보기 **수행시간** 문제를 해결하기 위해 실행해야하는 단계의 길이 -반복문을 통해 문제를 해결하기 위해 실행해야하는 단계의 길이를 짧게 만드는 것의 중요함을 강조 -수행시간이 극단적으로 긴 경우와 극단적으로 짧은 경우 비교	20
	● 리스트 이해하기 -비슷한 특성을 가진 자료들을 연결해 놓은 것을 **리스트** **● 리스트의 특징 및 예제 풀이** - 일반적인 변수에는 1가지 값밖에 저장하지 못하지만, 리스트에는 여러 개의 데이터를 저장할 수 있다 - 리스트 관련 기본 예제 풀이	21~22
정리	**● 학습한 내용 복습 및 정리** **● 다음 차시 배울 내용 문제와 퀴즈 제시**	23

일시	2018.8.17.	대상	파이썬 B반
주제	파이썬 탐색 알고리즘	차시	24~29/36
활동명	파이썬 탐색 알고리즘 익히기	소요시간	300분
학습목표	파이썬 탐색 알고리즘을 익히고 사용할 수 있다.		

학습 요소	교수·학습 활동	차시
도입	학습내용 안내 동기유발 활동 파이썬 리스트 학습지 및 과제 피드백을 통한 전시학습 상기	24
전개	**●탐색의 원리 알아보기** - 컴퓨터가 자료를 찾는 것과 사람이 자료를 찾는 원리가 다르다는 점을 강조 **●탐색의 종류** - 다양한 탐색 종류가 있지만, 초등학생 수준에 맞게 선형탐색과 이진탐색을 학습 1. 선형탐색 2. 이진탐색 - 각각 탐색 방법의 원리를 프로그래밍 하면서 알아보기	25~26
	●선형탐색과 이진탐색의 비교(수행시간을 중심으로) 1. 보물찾기 게임으로 각각 탐색방법에 따른 수행시간 알아보기 2. 선형탐색과 이진탐색의 최선&최악의 수행시간을 비교 3. 비교 결과를 바탕으로 그래프 작성	27~28
정리	**● 학습한 내용 복습 및 정리** **● 개인별 프로젝트 최종과제 계획서 작성**	29

일시	2018.8.18.	대상	파이썬 B반
주제	개별 프로젝트 과제 발표	차시	30~36/36
활동명	파이썬 개별 프로젝트 과제 발표	소요시간	350분
학습목표	파이썬을 활용한 수행시간 기반 탐색 알고리즘 개별 프로젝트 과제를 발표할 수 있다.		

학습 요소	교수·학습 활동	차시
도입	학습내용 안내 동기유발 활동 개별 프로젝트 과제 발표 연습	30
전개	● **개인 최종 과제 발표** - 파이썬을 활용한 수행시간 기반 탐색 알고리즘 개별 프로젝트 과제 발표 - 프로젝트 과제 피드백 및 느낀 점 공유	31~33
	● **학습한 모든 내용 총정리** - 학생 나름의 방법으로 그동안 학습한 내용 정리	34
정리	● **설문 및 논리성 검사(GALT) 사후검사 실시**	35~36