컴퓨팅사고를 위한 파이썬 워크북

컴퓨팅사고를 위한 파이썬 워크북

– 파이썬 기초 중심으로

발행 2023년 11월 01일
지은이 최미선, 최지영, 구진희
발행인
발행처
주소
전화
팩스
전자우편
홈페이지
ISBN

본 교재는 2023년도 과학기술정보통신부 및 정보통신기획평가원의 'SW중심대학사업' 지원을 받아 제작되었습니다.

컴퓨팅사고를 위한 파이썬 워크북

파이썬 기초 중심으로

최미선, 최지영, 구진희 지음

충남대학교 소프트웨어중심대학사업단
Chungnam National University National Center of Excellence in Software

배재대학교 AI·SW중심대학사업단
PAI CHAI UNIVERSITY

일러두기

1. 이 책은 파이썬 프로그래밍 입문자를 위한 기초 실습용 워크북이다.

이 책은 프로그래밍 입문자를 위한 용도로 개발되어 프로그래밍 개념 활용 연습 위주로 구성되어 파이썬 프로그래밍 기초 부분만을 다루고 있다.

2. 이 책에서는 파이썬 언어 문법의 기초적인 부분만을 다루고 있으므로, 자세한 문법은 파이썬 공식 홈페이지 문서를 참조하기 바란다.

파이썬 공식 홈페이지(http://python.org)의 Python Docs 에는 파이썬 버전별로 튜토리얼 및 라이브러리 설명 등의 다양한 문서 자료가 공개되어 있다. 이 책에서 다루는 내용 이외의 추가 내용은 Python Docs 에서 찾아보기 바란다.

contents

1장

컴퓨팅사고와 프로그래밍

STEP 01

컴퓨팅사고란?

컴퓨팅 사고력(Computational Thinking)은 컴퓨팅 분야에서 사용되는 기본적인 개념과 원리를 기반으로 문제를 효율적으로 해결할 수 있는 사고 능력을 말한다. 이 절에서는 컴퓨팅사고 주요 개념에 대하여 알아보도록 하자.

❑ 컴퓨팅 사고(Computational Thinking)란?

그림 1 지넷 윙 교수

카네기 멜론 대학의 지넷 윙(Jeannette Wing) 교수는 2006 년에 처음으로 'Communications of ACM' 학술지에 컴퓨팅 사고력(Computational Thinking) 개념을 처음으로 발표하였다. 지넷 윙 교수는 '컴퓨팅 사고란 문제의 해결책을 만드는 사고 과정으로서, 이 방법으로 생성된 해결책은 컴퓨터에 의해 효율적으로 실행될 수 있다'고 하면서, 읽고 쓰기, 셈하기 처럼 누구나 가지고 있어야 할 필수 역량이라고 주창하였다. 지넷 윙 교수의 주장 이후에 전 세계에서 컴퓨팅 사고 열풍이 시작되어, 우리나라에서도 2018 년부터 초등학교부터 컴퓨팅 사고를 다루는 교육이 필수화되어 실시되고 있다.

❑ 컴퓨팅 사고력 개념

컴퓨팅 사고는 우리에게 당면한 문제를 컴퓨터의 도움을 받아서 해결하는데 필요한 사고력으로, 일반적으로 사용되는 컴퓨팅 사고의 주요 개념으로는 문제분해, 패턴인식, 추상화, 알고리즘이 있다.

- 문제 분해(Problem Decomposition) : 주어진 문제를 해결 가능한 좀 더 작은 문제로 나누어가는 과정이다.
- 패턴 인식(Pattern Recognition) : 문제 내부나 문제들 간에 존재하는 반복되는 패턴이나 규칙을 찾는 과정이다.

- 추상화(Abstraction) : 문제 해결에 관련 없는 세부 사항은 제거하고 중요한 정보를 추출하여 문제를 단순하게 표현한다.
- 알고리즘(Algorithm) : 문제 해결을 위한 절차를 순차, 선택, 반복을 이용하여 구조적으로 표현한다.

결국 컴퓨팅 사고는 주어진 복잡한 문제를 해결하기 위하여 문제를 분석하고 분해하여, 패턴인식과 추상화를 통해 풀어야 할 문제를 단순화한 후, 문제를 해결하는 절차, 즉 알고리즘을 만드는 일련의 과정으로, 알고리즘을 프로그램으로 작성하여 컴퓨터로 자동화할 수 있다.

그림 2 컴퓨팅 사고와 프로그래밍을 통한 문제해결 과정

❑ 알고리즘과 프로그래밍

컴퓨팅 사고의 결과로 문제 해결을 위한 절차인 알고리즘이 만들어지면, 프로그램으로 구현하여 컴퓨터에서 실행되도록 함으로써 자동화할 수 있다. 아래 그림은 정수를 하나 입력받아 짝수인지 홀수인지를 판별하여 출력하기 위한 알고리즘을 순서도로 표현한 후, 파이썬 프로그램으로 구현한 것이다. 이와 같이 알고리즘은 파이썬 언어뿐만 아니라 C, Java 등 다양한 프로그램 언어로 구현될 수 있다. 이 교재에서는 컴퓨팅 사고의 결과로 나온 알고리즘을 자동화 하는데 필요한 기초 파이썬 프로그래밍 방법에 대해서 알아보기로 한다.

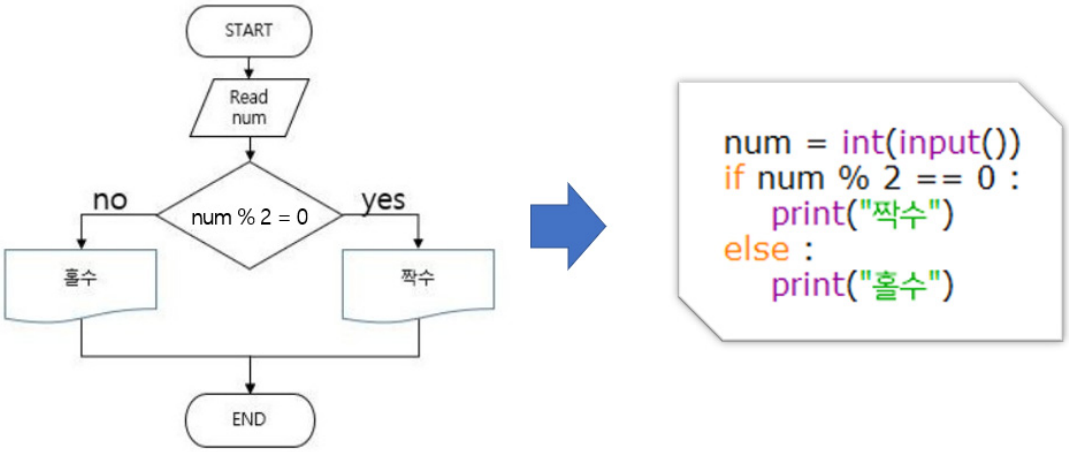


그림 3 순서도로 표현된 알고리즘과 파이썬 프로그램

STEP 02

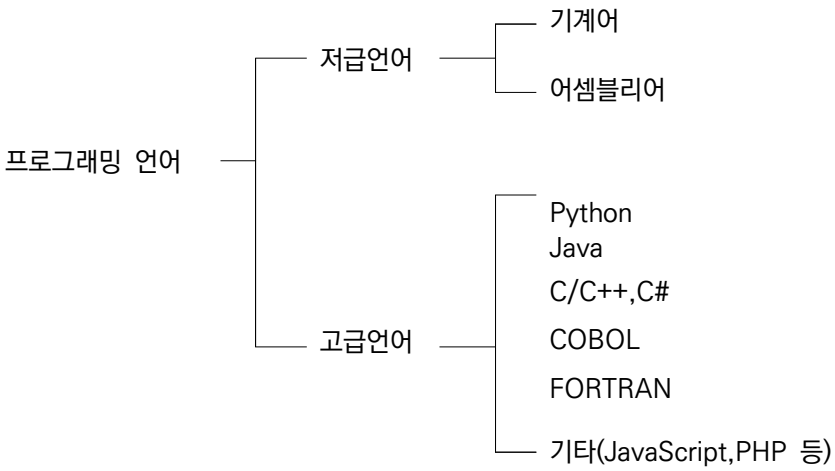
프로그래밍이란?

프로그램(program)은 컴퓨터에게 작업을 지시하는 명령어들의 집합이다. 파이썬과 같은 프로그래밍 언어로 컴퓨터를 위한 코드(명령어)를 작성하는 작업을 프로그래밍(programming) 또는 코딩(coding)이라고 한다.

❑ 프로그래밍 언어(Programming languages)의 필요성

컴퓨터는 내부적으로 '전기가 흐름'과 '전기가 흐르지 않음'의 두가지 상태만을 구분할 수 있는 소자로 구성되어 있어, 모든 정보는 이진수로 저장되고, 처리되어야만 한다. 한글, 영어와 같은 자연어(natural language)로 대화하는 사람이 이진수를 사용하는 컴퓨터에게 작업을 지시하려면 어떻게 해야할까? 사람과 컴퓨터가 대화하기 위해서는 사람과 컴퓨터가 이해할 수 있는 공통의 언어가 필요하고, 우리는 이러한 언어를 프로그래밍 언어(programming language)라고 부른다. 프로그래밍 언어를 통해 사람이 컴퓨터에게 작업 지시 명령을 내리면, 컴퓨터가 프로그래밍 언어로 작성된 명령을 이해하여 작업을 수행하면 된다.

❑ 프로그램 언어의 분류



프로그래밍 언어는 크게 저급언어(low-level language)와 고급언어(high-level language)로 나뉜다. 저급 언어는 컴퓨터에 가까운 언어로 기계어와 어셈블리어가 있다. 기계어는 기계(컴퓨터)가 사용하는 언어로 모든 명령을 이진수로 직접 작성해야 하기 때문에 사람이 사용하기에는 매우 어렵다. 어셈블리어는 기계어에서 명령어로 사용되는 이진수 조합을 간단하게 이해할 수 있는 영단어(LOAD X, STORE X 등)로 대체한 것으로 기계어보다는 프로그램 코드를 작성하기 쉬우나 이 역시, 프로그램 작성 시 상당한 노력이 필요하다.

고급언어는 사람들이 사용하는 일상적인 언어와 기호들을 그대로 이용하여 명령어들이 구성되어 있다. 따라서 사람이 프로그램을 작성하고 이해하기에는 편리하나, 0 과 1 만을 이해하는 컴퓨터가 고급언어로 된 프로그램을 실행하기 위해서는 기계어로 번역하는 과정이 필요하다. 프로그래밍 언어마다 그 언어로 작성된 프로그램을 컴퓨터가 이해할 수 있도록 해주는 번역기 프로그램이 제공된다. 번역기는 고급언어 프로그램을 0 과 1 로 된 기계어 프로그램으로 자동으로 변환해주는 역할을 하며, 번역하는 방식에 따라 컴파일(compile) 방식과 인터프리트(interpret) 방식으로 나뉜다.

컴파일 방식은 고급언어로 된 프로그램이 저장된 소스파일(source file)을 한꺼번에 기계어로 번역하여 컴퓨터에서 실행 가능한 이진수로 된 실행파일을 생성한다. 반면에 인터프리트 방식은 프로그램을 한줄 단위로 기계어로 번역한 후 바로 실행시켜 그 결과를 보여주는 방식으로 컴파일 방식과는 달리 실행파일을 생성하지 않는다. 앞으로 배우게 될 Python 언어와 Java, JavaScript, Perl 등의 언어들은 인터프리트 방식을 사용한다. 컴파일 방식을 사용하는 대표적인 프로그래밍 언어들로 C/C++ 등이 있다. 컴파일 방식과 인터프리트 방식을 사용하는 번역기 프로그램을 각각 컴파일러(compiler), 인터프리터(interpreter)라고 부른다.

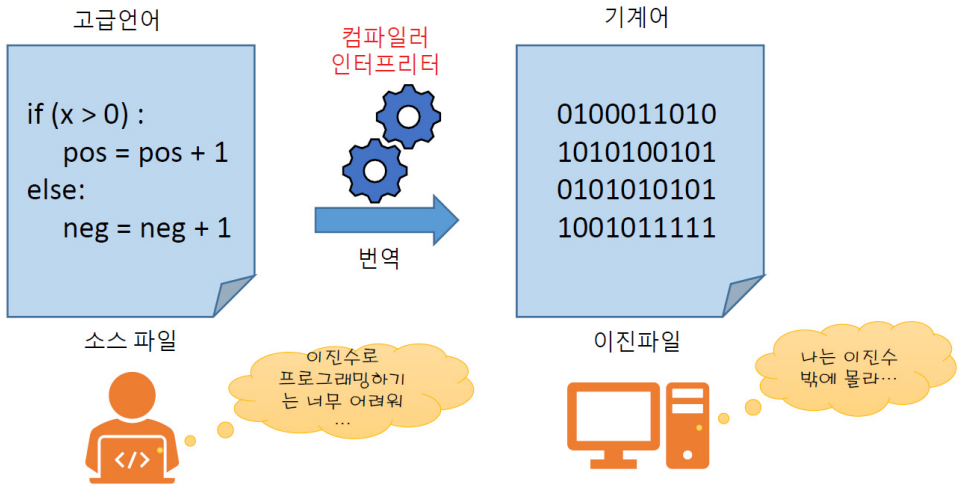


그림 4 번역기 프로그램의 필요성

❑ 프로그래밍 과정

컴퓨터를 이용하여 문제 해결을 자동화하기 위해서는 먼저 해당 작업을 수행할 수 있는 최적의 알고리즘을 설계하고, 문제 해결에 적합한 고급 프로그램 언어를 선택하여 알고리즘을 프로그램으로 구현해야 한다. 고급언어로 작성된 프로그램은 언어의 번역 방식에 따라 컴파일러 또는 인터프리터에 의해 기계어로 번역되고, 컴퓨터에서 실행되어 결과가 나오게 된다.

소스 파일에서 프로그래밍 언어에서 정해진 문법 규칙에 어긋나는 부분이 있는 경우, 기계어로 번역하는 과정에서 오류가 발생한다. 이런 오류를 **문법 오류(syntax error)**라고 부른다. 번역기는 문법 오류에 대한 메시지를 출력해주므로, 오류 메시지에서 잘못된 부분에 대한 힌트를 얻어서 문법 오류를 바로 잡은 후, 다시 번역 과정을 거치면 된다.

문법 오류를 수정한 후에도, 실행 과정에서 오류가 발생할 수 있는데, 이러한 오류를 실행시간 오류(run-time error)라고 부른다. 예를 들어서 나눗셈 연산을 할 때 0 으로 나눌 경우 실행시간 오류가 발생한다. 실행시간 오류도 오류 메시지를 잘 살펴보고 오류가 난 곳을 찾아서 해결하면 된다.

또 다른 오류의 유형은 논리 오류(logic error)로 문법도 잘못되지 않았고, 실행도 되나, 논리적으로 정확하지 않은 결과가 나오는 경우를 말한다. 예를 들어서 1~10 까지의 자연수의 합을 구하는 프로그램에서 55 가 아니라 100 과 같이 잘못된 답이 나온다면, 논리 오류가 발생한 것이다. 논리 오류는 알고리즘을 잘못 설계해서 나오는 경우이므로 프로그램 제어 흐름을 하나하나 따라가면서 잘못된 곳이 있는지 일일이 확인하는 과정을 거쳐서 오류를 찾아내야 한다. 이와 같이 프로그램에서 다양한 유형의 오류를 찾아서 수정하는 과정을 디버깅(debugging)이라고 한다.

그림 5 COBOL 언어 창시자 그레이스 호퍼가 MarkII 컴퓨터 오류를 찾는 과정에서 회로 사이에 낀 나방을 제거하여 오류를 수정하게 된 일화에서 '디버깅' 용어가 유래되었다.

STEP 03

파이썬 설치하기

파이썬(Python)은 1990 년 네덜란드의 Guido Van Rossum 이 개발한 인터프리터 언어이다. 파이썬은 문법이 간결하여 배우기 쉽고, 사람이 생각하는 방식을 그대로 표현할 수 있을 뿐만 아니라 개발 속도가 빠르며, 영상처리, 데이터분석, 인공지능 등 최신 애플리케이션(application) 개발에 필요한 풍부한 라이브러리들이 무료로 제공되고 있어 가장 인기있는 프로그래밍 언어 중의 하나로 각광받고 있다.

❑ 파이썬으로 할 수 있는 일

파이썬은 범용 프로그래밍 언어이므로 대부분의 응용 프로그램 개발에 적합하다. 무료로 제공되는 NumPy(수치연산 라이브러리), Pandas(데이터분석 라이브러리), Matplotlib(데이터 시각화 라이브러리), Tensorflow/Keras(딥러닝 라이브러리), OpenCV(영상처리 라이브러리) 등 다양한 오픈소스 라이브러리를 활용하면 수치연산, 데이터분석, 인공지능, 사물인터넷 등 최신 기술 분야의 응용 프로그램을 어렵지 않게 개발할 수 있다.

그림 6 다양한 파이썬 라이브러리

❑ 파이썬 개발 환경

파이썬 프로그램을 사용하려면, 파이썬 프로그램을 작성하고, 번역해서 실행할 수 있는 파이썬 개발환경이 필요하다. 파이참(PyCharm) 등 파이썬 전용 개발 환경 또는 Visual Studio Code 등과 같은 범용 프로그래밍 개발환경을 컴퓨터에 설치하거나, 별도의 프로그램 설치없이 웹브라우저에서 구글 코랩(Google Colab) 메모장과 같은 클라우드 환경을 이용할 수 있다.

(1) Python IDLE(Integrated Development and Learning Environment)

파이썬에서 기본적으로 제공하는 통합 개발 및 학습 환경으로 파이썬 홈페이지(www.python.org)에서 제공되는 파이썬 설치 파일을 이용하면 같이 설치된다. 파이썬 쉘창(Shell Window)에서 직접 대화식으로 명령을 입력하거나, 윈도우즈 메모장앱과 유사한 에디터창(Editor Window)에서 프로그램을 작성하고 실행시킬 수 있다.

(2) 파이참(PyCharm)

파이참은 JetBrains 가 개발한 파이썬 프로그래밍 언어에 특화된 통합 개발 환경으로 스마트 코드 완성, 코드 검사, 즉석 오류 강조 표시 및 빠른 수정, 자동 코드 리팩터링 등을 제공하여 전문 개발자용으로 많이 사용된다. JetBrains 홈페이지(www.jetbrains.com)에서 무료 버전인 Commumity Edition 을 다운로드받아 사용할 수 있다.

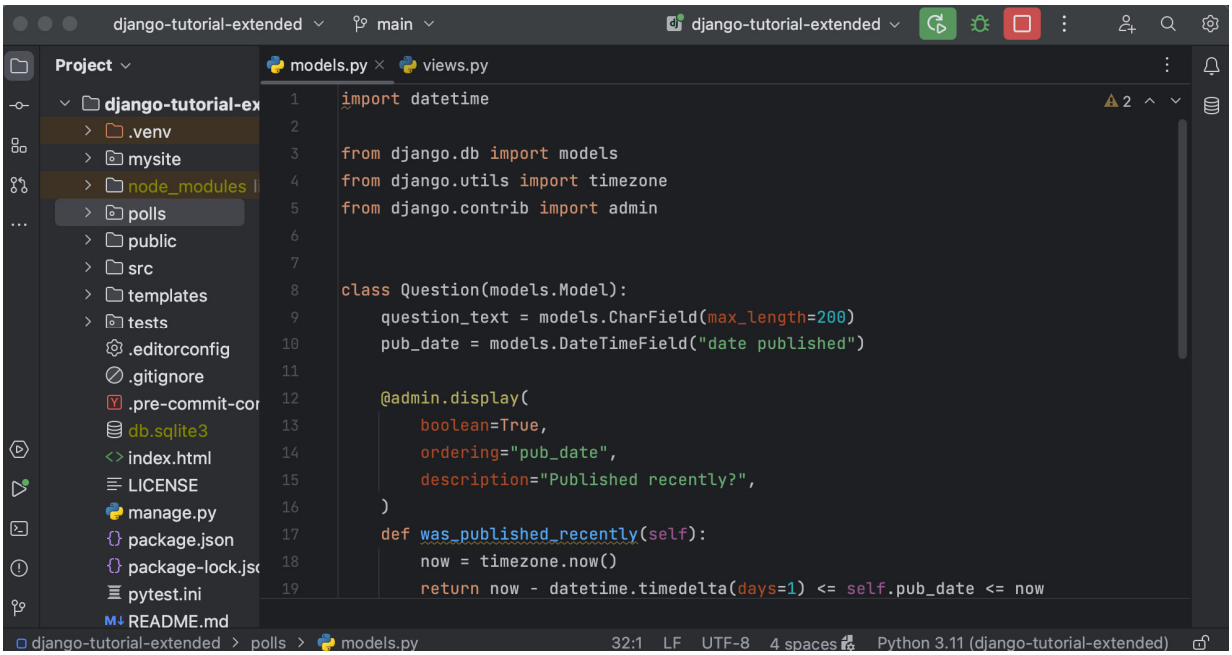


그림 7 파이참 개발환경(www.jetbrains.com)

(3) 주피터 노트북(Jupyter Notebook)

주피터 노트북은 컴퓨팅 문서를 만들고 공유하기 위한 웹 애플리케이션으로 파이썬 등 40 개 이상의 프로그래밍 언어 코드를 셀 단위로 작성해 바로 실행 가능하고, 결과를 바로 확인할 수 있다. 주피터 노트북을 사용하면 실행 코드와 서식 있는 텍스트를 이미지, HTML, LaTeX 등과 함께 하나의 문서로 통합할 수 있다. 주피터 노트북은 주피터 홈페이지(www.jupyter.org)에서 다운로드받아 설치하거나, 데이터분석이나 머신러닝이나 데이터분석 등에 사용하는 여러가지 패키지가 기본적으로 포함되어 있는 파이썬 배포판인 아나콘다(Anaconda, www.anaconda.com) 를 다운로드받아 설치하면 주피터 노트북도 함께 설치된다.

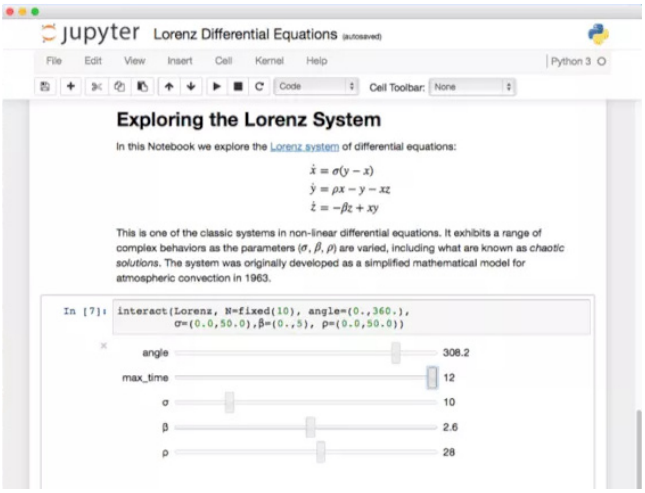


그림 8 주피터 노트북 사용 예

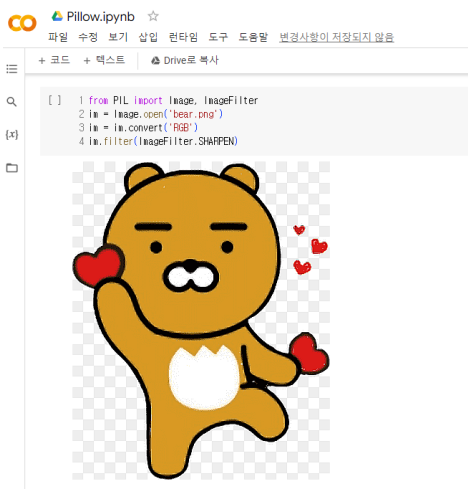


그림 9 구글 코랩 메모장 사용 예

(4) 구글 코랩(Google Colab)

구글 코랩은 컴퓨터에 파이썬을 설치하지 않아도, 구글 클라우드 서버 환경에서 Colaboratory(줄여서 'Colab'이라고 함)를 통해 브라우저 내에서 파이썬 스크립트를 작성하고 실행할 수 있다. 파이썬 스크립트를 작성하는데 사용되는 코랩 메모장은 주피터 노트북 환경을 사용하며, 작성된 파일은 사용자 컴퓨터가 아닌 사용자의 구글 드라이브 계정에 저장된다.

❑ 파이썬 기본 개발 환경 설치

이 교재에서는 초보자를 위한 교재이므로 파이썬 기본 IDLE 을 이용하여 프로그래밍 실습을 진행할 것이다. 파이썬 공식 홈페이지에서 설치 파일을 다운로드 받아 파이썬 기본 개발 환경을 컴퓨터에 설치하여 보자.

(1) 파이썬 설치 파일 다운로드

파이썬 홈페이지(www.python.org) [Downloads] 메뉴에서 사용자 컴퓨터 운영체제 유형에 맞는 Python 3.x 최신 설치 파일을 다운로드 받는다. 이 교재에서는 Python 3.11 버전을 기준으로 설치 방법과 프로그래밍 방법을 설명할 것이다.

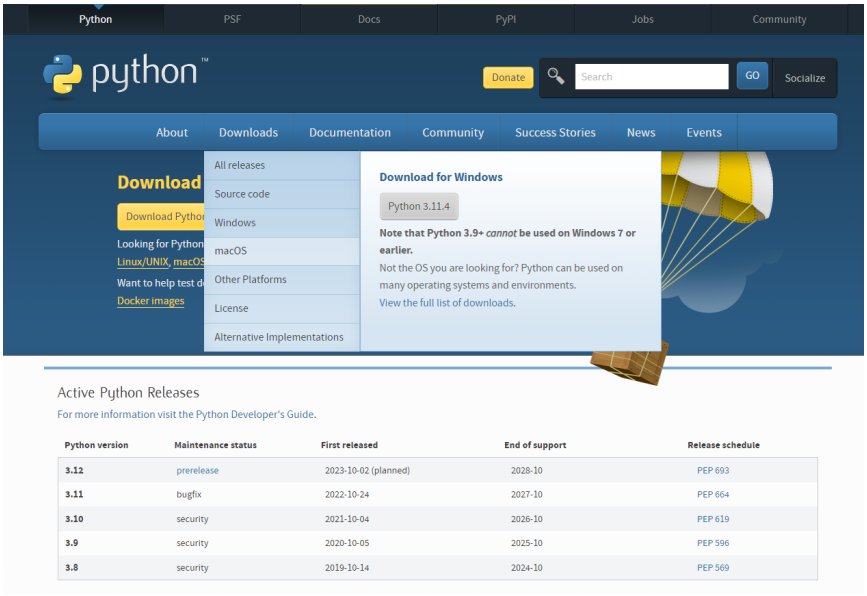


그림 10 파이썬 홈페이지 다운로드 메뉴 화면

(2) 윈도우에 파이썬 설치하기

이 교재에서는 가장 많이 사용되는 Windows OS를 기준으로 파이썬 설치과정을 설명한다. 다운로드 받은 설치파일을 더블클릭하여 실행시키면 아래와 같은 대화상자가 나타난다. 파이썬이 컴퓨터의 어느 폴더에서도 실행될 수 있도록 하단의 'Add python.exe to PATH' 체크박스를 추가로 선택한 후, [Install Now]를 클릭하여 설치과정을 진행한다.

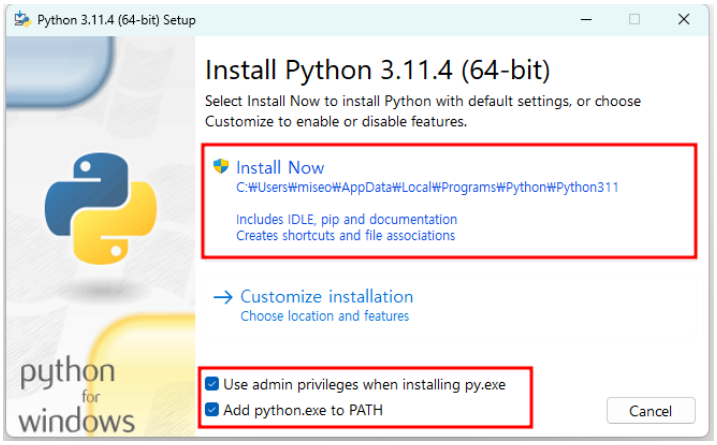
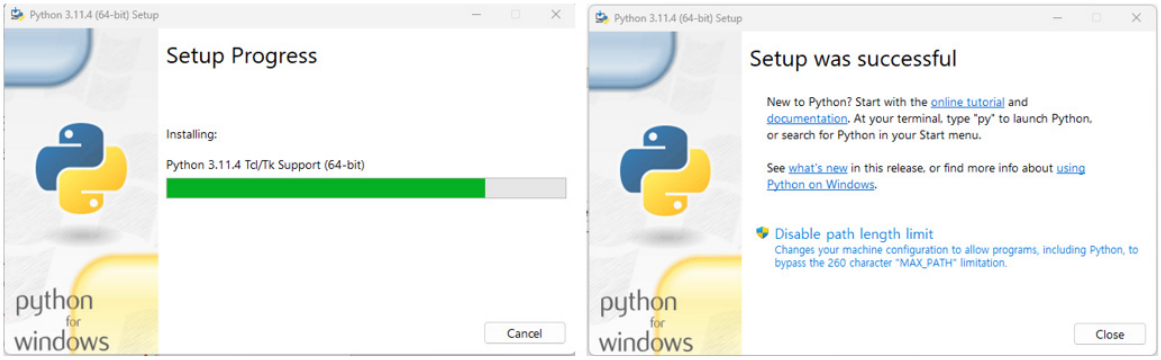


그림 11 윈도우용 설치 대화상자

이후의 설치 과정은 아래 왼쪽 그림과 같이 자동으로 진행되며. 오른쪽 그림과 같이 설치가 성공되었다는 대화상자가 나오면 설치가 종료된 것이므로 [Close]버튼을 클릭한다.



(3) 파이썬 설치 확인

파이썬 설치가 성공적으로 끝나면, 윈도우즈 메뉴의 모든 앱에서 방금 설치한 Python 3.11 폴더가 생성되어 있고, 하단에 관련 프로그램들이 설치되어 있는 것을 확인할 수 있다. Python 실행파일(Python 3.11)과 함께 통합개발환경(IDLE), 매뉴얼 등이 함께 설치되어 있다.

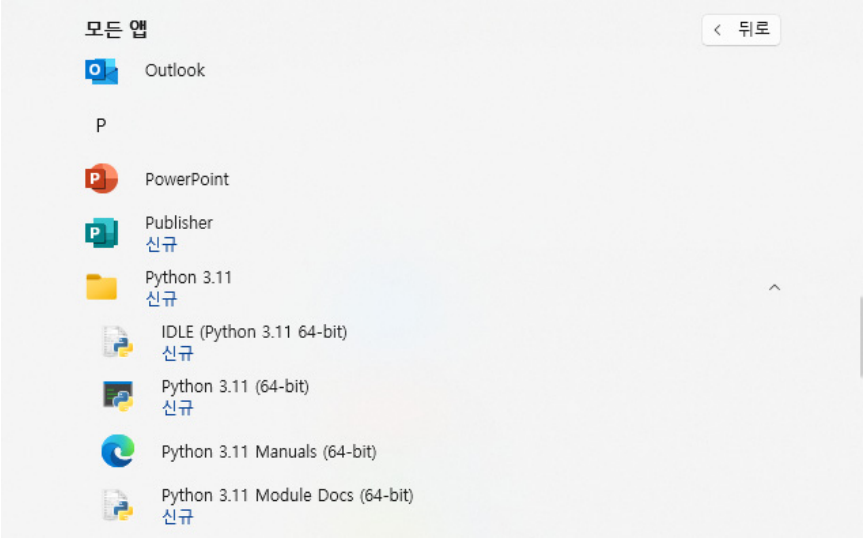


그림 12 파이썬 설치 화면(윈도우즈 11 기준)

첫 번째 파이썬 프로그램

STEP 04

파이썬 IDLE 에서 간단한 파이썬 프로그램을 작성해보면서 파이썬 프로그램의 기본 구조를 익히고, IDLE 에서 프로그램을 작성하고, 실행시키는 방법을 배워보자.

❑ 파이썬 IDLE 실행하기

윈도우즈 메뉴에서 [모든 앱] - Python 3.11 을 찾아 IDLE 을 더블클릭하여 실행시키면 오른쪽과 같은 파이썬 쉘(Shell) 창이 나타난다.

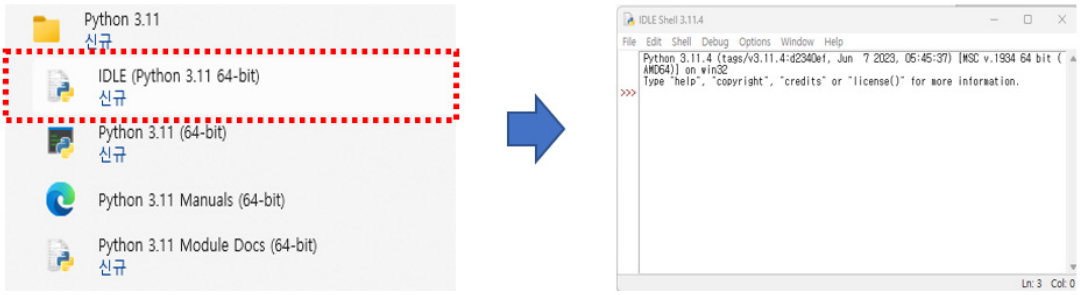


그림 13 파이썬 IDLE 실행 시 나타나는 파이썬 쉘창

파이썬 쉘창은 파이썬 프로그램 실행 결과가 출력되는 창으로 쉘창에서 한줄씩 명령어를 대화식으로 입력해서 결과를 바로 확인해볼 수 있으나, 여러줄 프로그램을 작성하기에 불편하고 코드 수정이 어려우므로 에디터창에서 프로그램을 작성하는 것이 편리하다.

아래 그림과 같이 파이썬 쉘창에서 [File]-[New File] 메뉴를 선택하면 에디터창이 나타난다.

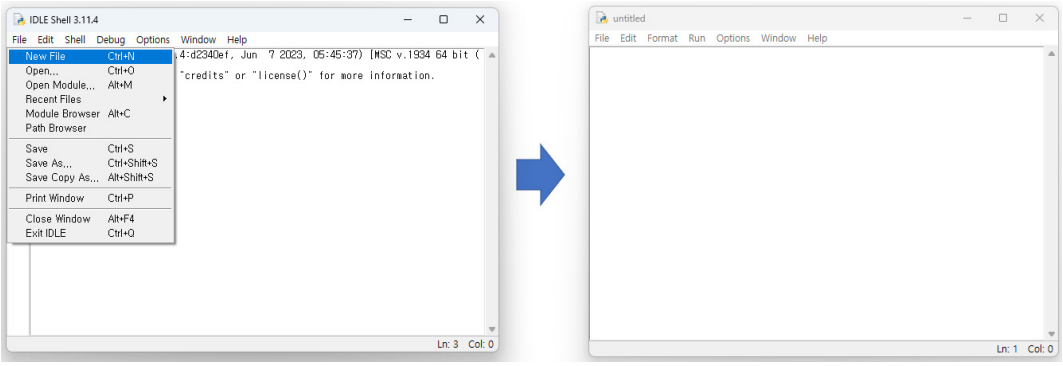


그림 14 파이썬 쉘 창에서 [New File] 메뉴 선택해서 에디터창 띄우기

에디터 창에서 프로그램을 입력하면 기본적으로 .py 확장자가 붙은 파이썬 소스 파일이 하나 생성된다. 파이썬에서 프로그램 소스파일은 모듈(module)이라고도 부른다.

❑ 첫 번째 파이썬 프로그램 작성하기

이제 파이썬 프로그램을 작성할 준비가 되었으니 에디터창을 열고 아래 그림과 같이 코드를 입력하여 화면에 한줄의 문장을 출력하는 첫 번째 파이썬 프로그램을 작성하여 보자.

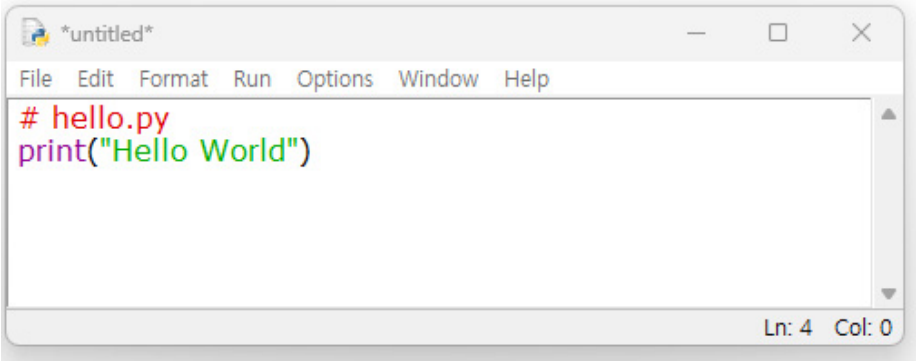


그림 15 에디터창에 첫 번째 프로그램 입력

첫 번째 줄의 '#'으로 시작하는 문장은 프로그램 명령문이 아닌 주석(comment)으로 프로그램 각 부분에 대한 설명에 해당한다. 주석은 프로그램의 가독성(readability)을 증진시켜 이해를 돕기위한 목적으로 작성하는 것으로 프로그램의 필수 요소는 아니다. 두 번째 줄이 파이썬 명령어 코드 부분이다. print("Hello World")는 화면에 따옴표 안에 있는 문장(Hello World)를 그대로 출력하는 기능을 가진 함수 호출문이다. print()문에 대해서는 2 장 STEP04 에서 자세히 알아볼 것이다.

이제 작성한 프로그램을 실행해보도록 하자.

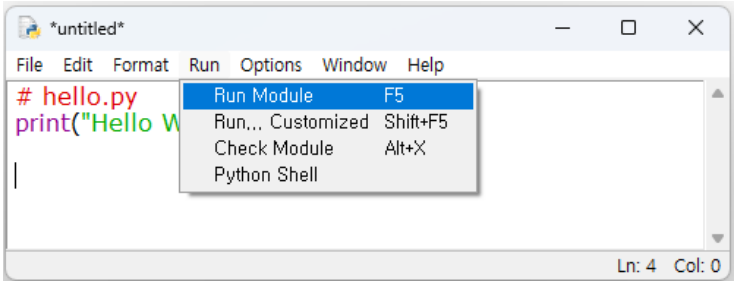


그림 16 파이썬 프로그램 실행하기

위와 같이 [Run]-[Run Module]을 선택하면 파일을 저장할지 묻는 대화상자가 나온다. 파일 저장위치와 파일명을 저장하면 파이썬 소스파일이 생성되고, 프로그램이 실행되어 파이썬 쉘창에서 실행 결과를 확인할 수 있다.

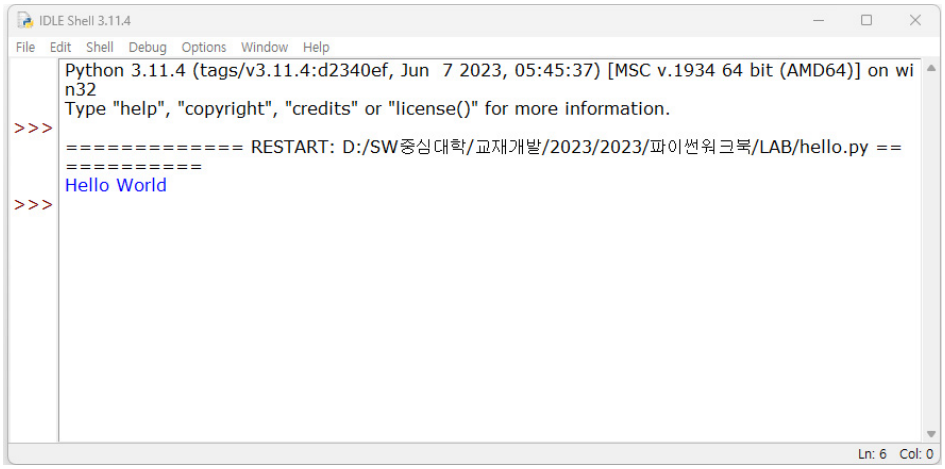


그림 17 파이썬 쉘창에 출력된 프로그램 실행 결과

프로그램을 잘못 작성한 경우에는 오류가 발생하여 프로그램 실행이 되지 않는다. 예를 들어, Hello World 문자열 뒤에 따옴표를 빠뜨린 경우, 아래와 같이 SyntaxError(구문 오류) 창이 뜨고 문자열이 제대로 끝나지 않았음을 의미하는 "unterminated string literal" 이라는 오류 메시지가 출력된다.

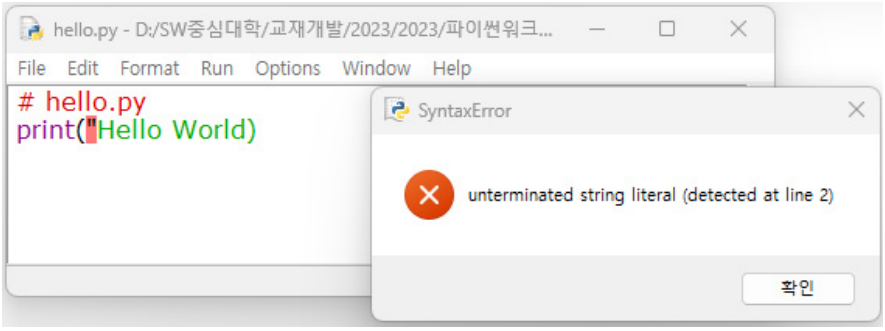


그림 18 구문 오류 발생 예

구문 오류가 발생한 경우, 오류가 없어질 때까지 오류를 수정하는 디버깅(debugging) 작업이 필요하다. 오류 메시지를 잘 해석하면, 오류 수정을 위한 힌트를 얻을 수 있다.

2장

파이썬 기본 구성요소

파이썬 기초 자료형

STEP 01

컴퓨터 프로그램에서 다루는 데이터의 종류는 문자, 정수, 실수, 리스트 등 매우 다양하다. 다양한 종류의 데이터를 효율적으로 다루기 위해서 프로그래밍 언어마다 자료형(data type) 개념을 사용하고 있다. 파이썬에도 모든 프로그램에서 일반적으로 사용되는 데이터의 종류를 표준 자료형 체계로 구분하고 있다.

❑ 파이썬 표준 자료형

파이썬 표준 자료형은 다음 그림과 같이 크게 숫자, 시퀀스, 부울형으로 구분할 수 있다.

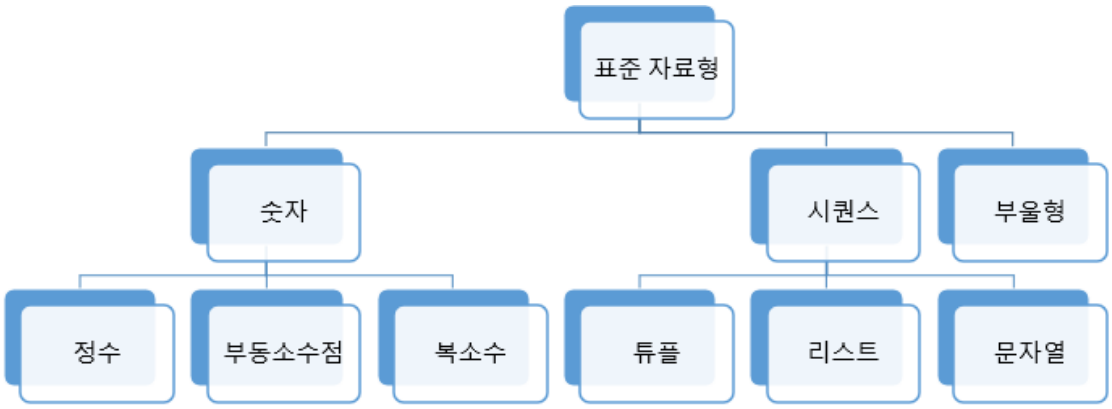


그림 19 파이썬 표준 자료형

숫자 자료형으로는 정수형(integer)과 소수점이 있는 실수형 표현을 위한 부동소수점형(float), *a+bi* 형태의 복소수 표현을 위한 복소수형(complex number)이 있다. 시퀀스(sequence)형은 자료들의 순서있는 모임으로 문자들의 연속된 모임인 문자열(string), 여러 항목들의 모임인 리스트(list), 튜플(tuple) 등이 있다. 부울(boolean)형은 참, 거짓을 나타내는 True, False 값을 표현할 수 있다.

자료형	값 예시
정수형(Integer)	-1500, 1500, 0, 100, 0o10, 0x1A
부동소수점형(Float)	-1.5, 3.141592, 123.456, 9000.0, 1.3E-5
문자열(String)	“Hello World!”, ‘안녕’, “123”, ‘@!#$%’, ‘A’
부울형(Bool)	True, False

파이썬 표준 자료형 중에서 가장 기본적인 자료형인 정수형, 부동소수형, 문자열, 부울형에 대해서 조금 더 알아보도록 하자.

❑ 정수형(Integer)

정수형은 수학에서 다루는 양의 정수, 0, 음의 정수를 모두 표현할 수 있으며, 숫자 표현에 주로 사용하는 10 진수(Decimal) 외에도 8 진수(Octal) 및 16 진수(Hexadecimal), 2 진수(binary)로도 표현할 수 있다.

아래 예는 정수를 10 진수, 8 진수, 16 진수로 표현하고 값을 화면에 출력한 것이다. ‘>>>’는 파이썬 쉘에서 맨 앞에 나오는 프롬프트로, 프롬프트 다음 커서 위치에 파이썬 명령문을 입력하고 엔터키를 치면 명령이 실행된다.

10진수	8진수	16진수

>>> a = 123
>>> a = -123
>>> a = 0
	>>> a = 0o123
	>>> print(a)
	83
		>>> a = 0x7ff
		>>> print(a)
		2047
		>>> b = 0xABC
		>>> print(b)
		2748

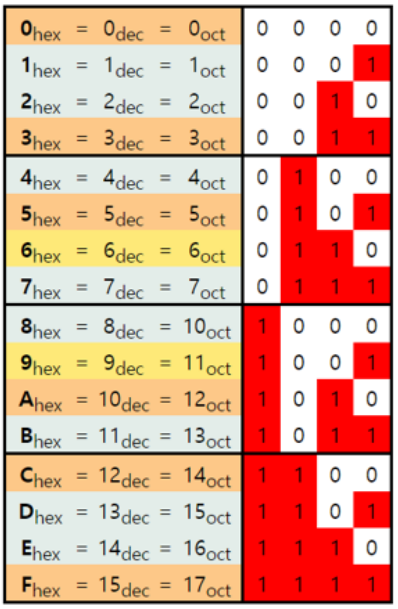


그림 20 여러 진법으로 표기된 숫자 (https://ko.wikipedia.org)

예시에서 a=123 명령문은 a 라는 변수에 정수값 123 을 대입하라는 의미이다. 변수와 대입문에 대해서는 **STEP02. 변수 알아보기**와 **STEP03. 수식과 연산자**에서 자세히 설명할 것이다.

숫자 앞에 0o(숫자 0 + 알파벳 ‘o’ 또는 ‘O’)가 붙어 있으면, 그 뒤에 나오는 숫자는 10 진수가 아닌 8 진수로 해석된다. 8 진수는 하나의 자리에 0~7 까지의 숫자를 표현할 수 있으며, 위의 예에서 0o123 의 값을 출력해보면, 123 이 아닌 $1 \times 8^2 + 2 \times 8^1 + 3 \times 8^0 = 83$ 이 계산되어 출력되게 된다.

16 진수는 숫자 앞에 0x(숫자 0 + 알파벳 'x' 또는 'X')를 붙여서 16 진수임을 나타낸다. 16 진수는 0~15 까지의 숫자를 표현할 수 있으나, 1 개의 자리에 숫자를 표현하기 위해 10~15 까지의 값은 문자 'A'~'F' 또는 'a'~'f'로 대신하여 나타낸다. 위의 예시에서 0x7ff 를 출력하면, $7 \times 16^2 + 15 \times 16^1 + 15 \times 16^0 = 2047$ 와 같이 10 진수로 계산되어 출력된다.

❑ 부동소수점형(Floating-point)

부동소수형은 소수점이 있는 실수를 표현한다. 1.5 와 같이 소수점을 붙여서 표현할 수도 있으며, 1.3E-20 과 같이 실수값이 너무 크거나 작을 때, 컴퓨터식 지수 표현 방식을 사용하여 간단하게 표현할 수 있다.

일반적인 실수 표현 방식	컴퓨터식 지수 표현 방식

>>> a = 1.5
>>> a = 15000000000.0
>>> a
15000000000.0
>>> a = 0.15
>>> a
0.15
	>>> a = 1.5E10
	>>> a
	15000000000.0
	>>> a = 1.5e-1
	>>> a
	0.15

❑ 문자열(String)

문자열은 문자들의 순서있는 집합으로 따옴표 안에 문자들을 나열하여 표현한다. 문자열을 구성하는 문자는 알파벳 대문자/소문자 뿐만 아니라 특수문자와 숫자도 따옴표로 묶으면 문자로서 사용할 수 있다. 따옴표는 큰따옴표(" ")와 작은 따옴표(' ') 모두 사용 가능하므로 상황에 따라 따옴표를 선택해서 편리하게 사용할 수 있다. 예를 들어서, '엄마가 "철수야" 하고 부르셨다'와 같이 문자열 중에 대화 내용이 포함되는 경우에 전체 문자열을 작은 따옴표로 묶고, 문자열 안에 포함된 부분을 큰따옴표로 묶음으로써 문자열 따옴표가 매치되지 않아 발생하는 문법 오류를 방지할 수 있다.

"Hello World!", "안녕하세요^^", '엄마가 "철수야" 하고 부르셨다', "I'm happy"

따옴표를 연속적으로 세 개 사용하면 아래와 같이 여러줄로 된 문자열을 쉽게 만들 수 있다. 여러줄 문자열은 여러 줄 주석(comments)을 작성할 때도 사용된다.

큰 따옴표로 여러줄 문자열 만들기	작은 따옴표로 여러줄 문자열 만들기
"""How are you? I'm fine. Thank you."""	'''동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라 만세 무궁화 삼천리 화려강산 대한사람 대한으로 길이 보전하세 '''

문자열 안에 역슬래시('\') 문자와 조합해서 특수한 기능을 하는 문자들이 있는데 이를 이스케이프 시퀀스(Escape Sequence) 제어문자라고 한다. 아래 표에 파이썬에서 자주 사용되는 이스케이프 시퀀스 문자들의 이름과 의미를 정리해놓았다. 이들은 2 개의 문자처럼 보이지만, 역슬래시와 결합하여 하나의 문자로 취급된다.

제어문자	이름	의미
\a	경고(bell)	"삐"하는 경고음 발생
\b	백스페이스(backspace)	커서를 현재 위치에서 한 글자 앞으로 옮긴다.
\t	수평탭(horizontal tab)	커서의 위치를 현재 라인에서 설정된 다음 탭 위치로 옮긴다.
\n	줄바꿈(newline)	커서를 다음 라인의 시작 위치로 옮긴다.
\"	큰 따옴표	큰 따옴표 문자 그 자체를 의미
\'	작은 따옴표	작은 따옴표 문자 그 자체를 의미
\\	역슬래시(back slash)	역슬래시 문자 그 자체를 의미

아래의 예는 문자열을 작은 따옴표로 묶는 경우, 중간에 나오는 아포스트로피(Apostrophe)로 사용되는 작은 따옴표를 표현하기 위해 이스케이프 시퀀스 \'를 사용하고, \n 제어문자를 사용하여 줄바꿈 위치를 지정하고 있다. 줄바꿈을 위해서는 \n 제어문자 대신에 따옴표 3 개로 문자열을 묶어서 여러줄 문자열을 정의할 수도 있다.

```
>>> mline = 'It\'s time for everyone to learn programming\nYou need python'
>>> print(mline)
It's time for everyone to learn programming
You need python
```

❑ 부울형(bool)

부울형은 참(True)과 거짓(False)을 나타내는 자료형으로 True 또는 False 상수값을 변수에 직접 대입하거나, 부울형으로 계산되는 조건식의 결과값을 이용하여 논리 연산을 할 수 있다. 논리 연산에 대해서는 **STEP03. 수식과 연산자**에서 다루게 될 것이다,

부울형 변수를 사용한 대입문	관계식의 결과는 부울형으로 계산됨

>>> a = True
>>> a
True
>>> a = False
>>> a
False
>>> type(a)
<class 'bool'>
	>>> 1 == 1
	True
	>>> 1 != 1
	False
	>>> 1 >= 2
	False
	>>> 1 < 2
	True

STEP 02

변수 알아보기

컴퓨터 프로그램에서는 많은 자료를 읽고, 쓰고, 연산하는 작업들을 포함한다. 프로그램 실행 중에 메모리에 임시로 저장되는 자료들을 프로그램에서 쉽게 접근할 수 있도록, 메모리 공간에 이름을 붙여놓고, 이름을 통해 쉽게 값에 접근할 수 있도록 하는 방법이 변수(Variable)이다.

❑ 파이썬 변수의 특징

변수는 자료값을 담는 상자라고도 할 수 있다. 자료값을 담는 상자를 생성하기 위해서 파이썬에서는 아래 왼쪽 그림과 같이 변수명을 써주고 대입연산자('=')을 이용해서 값을 넣어주기만 하면 자동으로 변수가 생성된다.

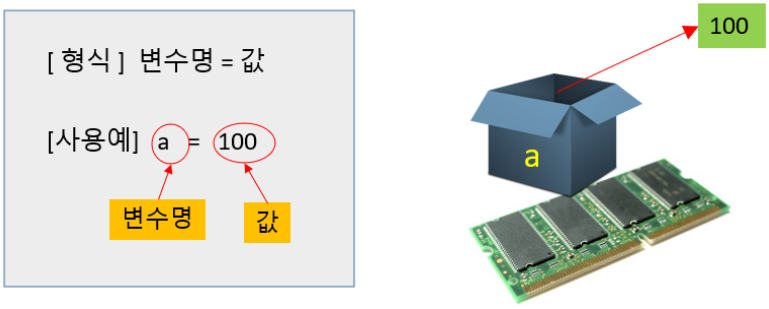


그림 21 파이썬 변수 사용 방법

C/C++나 Java 등의 전통적인 프로그래밍 언어들에서는 변수를 사용하기 전에 변수의 자료형과 변수명을 같이 써주는 변수 선언 과정이 필요하다. 파이썬에서는 이러한 변수 선언 과정 없이 변수에 값을 대입하는 것으로 변수가 자동으로 만들어진다. 변수 선언 과정이 없으므로, 파이썬 변수는 특정 자료형에 한정되지 않고, 모든 유형의 자료를 저장하는데 사용될 수 있다.

❑ 변수명 짓기

변수명을 지을 때는 다음과 같은 파이썬 변수 명명 규칙을 따라야 한다.

- 변수 이름은 대소문자를 구분한다.
- 변수 이름을 시작할 때는 숫자, 특수문자의 사용을 피해야 한다.
  (즉, 시작글자는 영문자(대소문자), 언더스코어(_)로만 가능)
- return, not, try, while, for, if, import 와 같은 파이썬 내부에서 사용하는 키워드는 변수 이름으로 사용할 수 없다.

파이썬에서는 한글 변수 이름도 가능하지만, 피하는 것이 좋으며, 변수에 저장되는 값과 관련있는 단어를 사용하여 변수 이름을 만드는 것이 기억하기 쉽고 프로그램 유지보수에 도움이 된다. 예를 들어, 높이값을 저장하기 위한 변수명으로 'a'라고 짓는 것보다 '높이'에 대한 영단어인 'height'로 변수명을 지으면 기억하기가 쉽다.

잘된 변수 이름 예	잘못된 변수 이름 예
width, student_id, studentName, id1, max_number, value, num5, not1, ...	1student, money$, Hello#, not, ...

❑ 변수 사용하기

변수를 사용하여 사각형의 면적을 구하는 간단한 예제를 살펴보도록 하자.

```
>>> width = 5                  # width 변수에 5를 저장
>>> height = 3                 # height 변수에 3을 저장
>>> area = width * height      # width와 height에 저장된 값을 읽어서 연산 수행 후,
                               # 결과값을 area 변수에 저장
>>> print(area)                # area 변수에 저장된 값을 출력
15
```

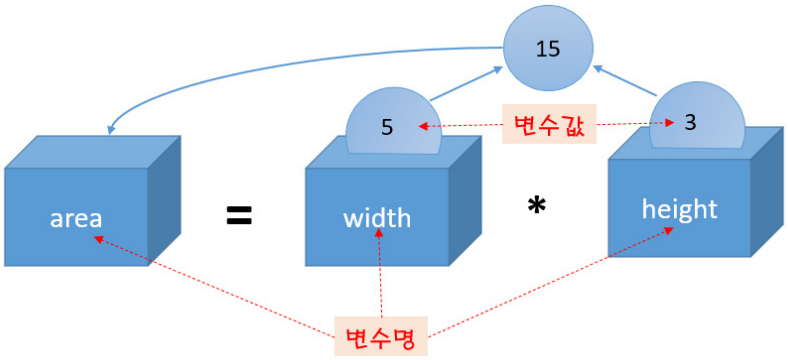


그림 22 사각형 면적 구하기 연산 실행 과정

먼저 가로, 세로 길이를 저장할 변수인 width, height 를 만들고 각각의 변수에 값을 저장한다. 사각형 면적을 구하기 위해 width, height 에 저장된 값을 읽어서 두 개의 값을 곱한 후, 결과값을 면적에 해당하는 area 변수에 저장한 후, print() 함수를 이용하여 area 값을 읽어서 출력한다.

다음은 문자형과 부울형 자료값을 변수에 저장하고 출력하는 간단한 예이다. 파이썬 변수는 자료형이 지정되어 있지 않으므로 같은 변수에 어떤 유형의 데이터도 대입문으로 저장할 수 있으나, 변수의 이름과 값 사이의 연관성을 고려하여 구분하여 사용하는 것이 좋다.

변수에 문자형 자료 사용하기	변수에 부울형 자료 사용하기

>>> str = "Hello"
>>> print(str)
Hello
>>> str = "안녕하세요"
>>> print(str)
안녕하세요
	>>> bvalue = True
	>>> print(bvalue)
	True
	>>> bvalue = False
	>>> print(bvalue)
	False

왼쪽 예제에서는 str 변수에 영어 문자열을 저장하고 출력 후에, 다시 한글 문자열을 새로 대입한 후 값을 출력하였다. 변수는 'variable'이라는 단어의 의미와 같이 변수에 저장된 값이 변하는 특징을 가지므로 필요할 때 새로운 값을 대입하여 사용할 수 있다. 오른쪽 예제에서는 bvalue 라는 변수에 True, False 값을 대입한 후 출력하고 있다.

파이썬 변수는 대입문으로 값을 변수에 저장함으로써 자동으로 생성되지만, 값이 정의되지 않은 변수를 읽으려고 시도하는 경우에는 아래와 같이 오류가 발생한다.

변수를 잘못 사용한 경우 오류 발생 예

```
>>> a = 10                                  # 변수 a에 10 대입
>>> b = 20                                  # 변수 b에 20 대입
>>> result = a + b + c                      # 변수 a, b 값을 읽고, 변수 c를 읽을 차례에서,
Traceback (most recent call last):            생성되지 않은 변수 c 가 정의되지 않아 오류 발생
  File "<pyshell#23>", line 1, in <module>
    result = a + b + c
NameError: name 'c' is not defined
```

❑ 변수 값 맞교환하기

파이썬 변수는 대입문으로 값을 맞교환할 수 있는 편리한 기능이 있다. 오른쪽 예에서 변수 a, b 의 값을 a, b = b, a 문장으로 바로 교환할 수 있다. C 언어와 같은 경우에는 하나의 문장으로 처리가 불가능해서 temp = a, a = b, b = temp 와 같이 임시 변수를 이용한 3 개의 대입문을 통해야지만 a, b 변수 값을 맞교환할 수 있다.

```
>>> a = 1
>>> b = 2
>>> a, b = b, a
>>> a
2
>>> b
1
```

STEP 03

수식과 연산자

컴퓨터 프로그램은 컴퓨터에게 연산을 지시하는 다양한 형태의 수식(expression)들을 포함한다. 수식은 연산자(operator)와 피연산자(operand)들로 구성되며, 연산을 수행하여 수식의 결과값을 생성한다.

❑ 수식이란?

수식이란 프로그램에서 컴퓨터에게 내리는 작업 지시 명령의 최소 단위로, 연산자와 연산에 필요한 피연산자로 구성되어 있으며, 연산자의 의미에 맞는 연산을 수행하여 수식의 결과값을 생성한다.

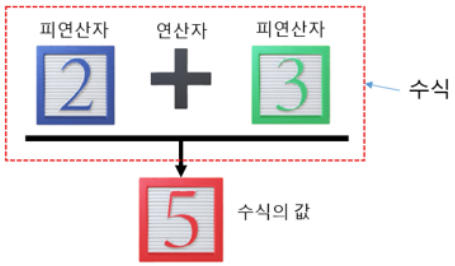


그림 23 수식의 구성요소

❑ 연산자의 종류

연산자는 피연산자의 개수에 따라 피연산자의 개수가 1 개인 단항 연산자(Unary operator), 피연산자의 개수가 2 개인 이항 연산자(Binary operator), 피연산자의 개수가 3 개인 삼항 연산자(Ternary Operator)로 구분할 수 있다.

연산자는 기능에 따라서 산술 연산자(Arithmetic Operator), 관계 연산자(Relational Operator), 논리 연산자(Logical Operator), 비트 연산자(Bit Operator)등으로 구분할 수 있다. 산술 연산자는 사칙 연산을 수행하는 연산자로, 대부분 이항 연산자로서 2 개의 피연산자를 가진다. 수학에서 사용되는 사칙 연산자와의 주요 차이점은 곱셈 연산자가 '*'(별표, asterisk) 기호로, 나눗셈 연산자가 '/'(슬래쉬, slash) 기호로 표시된다는 것이다. 거듭제곱을 계산하는 지수 연산자는 '**'와

같이 별표 두 개를 붙여서 사용하고, 정수로 나눈 나머지 값을 계산하는 연산을 추가로 제공하여, '%' 기호로 나타낸다. 파이썬 언어는 다른 프로그래밍 언어들과 다르게, 나눗셈의 피연산자 유형에 따라 정수 나눗셈은 '//' 기호로, 실수 나눗셈은 '/' 기호로 구분하여 표기하고 있다.

연산자	기호	사용 예	결과값
덧셈	+	2 + 3	5
뺄셈	-	2 - 3	-1
곱셈	*	2 * 3	6
나눗셈(정수)	//	3 // 2	1
나눗셈(실수)	/	3 / 2	1.5
나머지	%	3 % 2	1
지수	**	2 ** 3	8

관계 연산자는 피연산자들의 값을 비교하여 부울형 값(True/False)으로 평가하는 연산자들로 주로 조건식에 많이 사용된다. 조건식에 대해서는 3 장에서 자세히 다루게 될 것이다.

연산자	기호	사용 예	결과값
크다	〉	2 〉 3	False
작다	〈	2 〈 3	True
같다	==	2 == 3	False
같지 않다	!=	2 != 3	True
크거나 같다	〉=	2 〉= 3	False
작거나 같다	〈=	2 〈= 3	True

논리 연산자는 부울형 피연산자들의 값에 대해 논리 연산을 수행하여 결과를 True/False 로 평가하는 연산자들로 조건식을 연결하는데 많이 사용된다.

연산자	기호	사용 예	결과값
AND 연산	and	x and y	x와 y가 모두 True일 때 True, 그 외에는 False
OR 연산	or	x or y	x와 y가 모두 False일 때 False, 그 외에는 True
NOT 연산	not	not x	x가 True 이면 False, x가 False 이면 True

비트 연산자는 0/1 의 두가지 정보를 나타낼 수 있는 비트(bit) 단위로 연산을 수행하는 연산자로 파이썬 기초를 다루는 이 교재의 범위를 벗어나므로 자세히 다루지는 않겠다.

❑ 연산자 우선 순위

파이썬 언어에는 지금까지 살펴본 것들 외에도 다양한 연산자들이 있다. 수학에서와 마찬가지로 프로그램에서도 수식에 여러 연산자들이 나올 때 어떤 연산이 먼저 수행되는지에 따라서 결과값이 달라지게 된다. 파이썬 언어는 아래와 같은 우선순위에 따라 연산 순서를 정하여 결과를 평가한다.

순위	연산자	설 명
1	**	지수 연산자
2	~, +, -	단항연산자
3	*, /, %, //	곱셈, 나눗셈, 나머지 연산자
4	+, -	덧셈, 뺄셈
5	&	비트 AND 연산자
6	^, |	비트 XOR 연산자, 비트 OR 연산자
7	<=, <, >, >=	비교 연산자
8	<>, ==, !=	동등 연산자
9	=, %=, /=, //=, -=, +=, *=, **=	대입, 복합 연산자
10	is, is not	동등 연산자
11	in, not in	멤버 연산자
12	not, or, and	논리 연산자

연산자 우선순위는 표의 위쪽으로 갈수록 높으며 우선순위가 가장 높은 연산자가 먼저 계산된 후, 다음 우선순위의 연산자가 순차적으로 계산되어 최종 수식값을 평가한다.

❑ 숫자 연산하기

파이썬 산술 연산자들을 이용하여 간단한 연산식들을 작성한 예시이다. 파이썬 IDLE 쉘에 입력하여 결과를 확인하여 보자.

덧셈, 곱셈 연산자 활용 예

```
>>> a = 3          # a에 3 대입
>>> b = 5          # a에 5 대입
>>> a + b          # a + b 의 값을 계산
8
>>> a * b          # a * b 의 값을 계산
15
```

지수 연산자 활용 예

```
>>> a = 2          # a에 2 대입
>>> b = 10         # b에 10 대입
>>> a ** b         # a^b 승 계산
1024
```

다음은 나눗셈과 나머지 연산을 실행한 예시이다. 실습을 통해 확인하여 보자.

나눗셈 연산자 활용 예	나머지 연산자 활용 예

>>> a = 3 # a에 3 대입
>>> b = 5 # a에 5 대입
>>> a / b # 실수 나눗셈
0.6
>>> a // b # 정수 나눗셈, 몫만 계산
0
	>>> a = 10 # a에 10 대입
	>>> b = 3 # b에 3 대입
	>>> a // b # a를 b로 나눈 몫
	3
	>>> a % b # a를 b로 나눈 나머지
	1

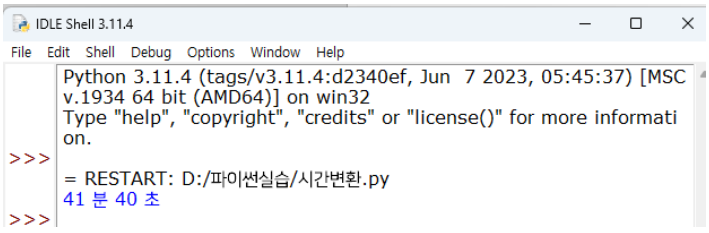
💻 실습 예제 1. 시간 변환하기

초단위로 주어진 시간을 '몇분 몇초'와 같은 형식으로 분과 초단위로 구분하여 출력하는 프로그램을 작성해보자.

프로그램 작성 전에 문제해결을 위한 절차를 생각해보고 간단하게 알고리즘을 정리해보자.

① 초단위 시간을 변수 sec 에 저장
② sec를 60으로 나눈 몫을 분단위 시간 변수 minute에 저장
③ sec를 60으로 나눈 나머지를 sec에 저장
④ min, sec 값을 몇분 몇초 형식으로 출력

이번에는 프로그램 수정이 편하도록 파이썬 IDLE 에디터를 사용하여 소스파일에 알고리즘에 해당하는 프로그램을 입력한 후 파일을 저장하고, [Run Module]을 선택하여 프로그램을 실행해보도록 한다.

프로그램	실행 결과
sec = 2500 minute = sec // 60 sec = sec % 60 print(minute, "분", sec, "초")	

💻 실습 예제 2. 거스름돈 계산하기

내가 가진 돈에서 최대로 살 수 있는 상품의 개수를 계산하고, 거스름돈을 계산해주는 프로그램을 작성해보자. 먼저 문제해결을 위한 알고리즘을 간단하게 정리해보자.

① money 변수에 가진 돈의 금액을 저장
② item_price에 상품 1개의 가격을 저장
③ money로 살 수 있는 item_pirce 상품의 개수를 계산하여 num_items 변수에 저장
④ item_price에 num_items 값을 곱해서 총 상품 금액을 계산하여 total_price 변수에 저장
⑤ money에서 total_price 값을 빼서 출력한다.

소스파일에 프로그램을 입력한 후 파일을 저장하고, 프로그램을 실행하여 결과를 확인해보자.

프로그램

change.py - D:/파이썬실습/change.py (3.11.4)

File Edit Format Run Options Window Help

```
money = 5000          # 가진 돈
item_price = 200      # 물건 1개 가격

# 내가 가진돈으로 살 수 있는 최대 개수
num_items = money // item_price
print("최대로 살 수 있는 개수 : ", num_items)

# 최대로 사고, 남는 거스름돈
total_price = num_items * item_price
change = money - total_price
print("거스름돈 : ", change)
```

실행 결과

IDLE Shell 3.11.4

File Edit Shell Debug Options Window Help

```
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun  7 2023, 0
n32
Type "help", "copyright", "credits" or "license()" for m
>>>
= RESTART: D:/파이썬실습/change.py
최대로 살 수 있는 개수 :  25
거스름돈 :  0
>>>
```

❑ 문자열 연산하기

문자열은 문자들의 순서있는 모임으로 숫자형과 함께 파이썬 프로그램에서 가장 많이 다루게 되는 자료형이다. 파이썬 문자열에 대해서 문자열 연결, 문자열 곱하기, 문자열 인덱싱, 문자열 슬라이싱 등 다양한 연산들이 가능하다.

(1) 문자열 연결과 반복 연산

산술 연산자인 '+'와 '*'는 숫자 연산과는 다르게, '+'는 문자열이 피연산자로 올 경우, 문자열 연결(concatenation) 연산자로서 동작하며, '*'는 문자열 곱하기 연산자로 동작한다.

문자열 연결하기	문자열 곱하기

>>> firstname = “Harry”
>>> lastname = “Potter”
>>> firstname + “ ” +lastname
‘Harry Potter’
>>> firstname = “김”
>>> lastname = “철수”
>>> firstname + lastname
‘김철수’
	>>> a = “ding-dong ”
	>>> a * 2 # a의 내용을 10회 반복
	‘ding-dong ding-dong ’
	>>> str = “짝”
	>>> str * 10 # str 내용을 10회 반복
	‘짝짝짝짝짝짝짝짝짝짝’
	>>> 5 * str # 숫자가 먼저 나와도 가능
	‘짝짝짝짝짝’

(2) 문자열 인덱싱(Indexing)

문자열은 문자들의 순서있는 모임으로 문자열을 구성하는 각 문자에 번호가 매겨져 있다. 프로그래밍 언어에서는 문자열의 각 요소에 부여되는 번호를 인덱스(index)라고 부른다. 파이썬에서는 양의 인덱스와 음의 인덱스를 모두 사용할 수 있는데, 양의 인덱스는 맨앞부터 0 에서 시작하며 1 씩 증가하는 방식으로 부여된다. 음의 인덱스는 문자열 맨 끝부터 앞쪽으로 -1 부터 시작해서 -1 씩 값이 증가하는 방식으로 매겨진다.

문자열에서 []와 인덱스를 사용하여 특정 문자를 뽑아내는 연산을 문자열 인덱싱(indexing)이라고 한다. 아래의 왼쪽 그림은 “HELLO” 문자열의 각각의 문자에 해당하는 양의 인덱스와 음의 인덱스를 보여주고 있다. 오른쪽 예시는 문자열 인덱싱 연산을 통해서 “HELLO”로부터 문자를 추출하는 프로그래밍 예제이다.

“HELLO”	H	E	L	L	O
양의 인덱스	0	1	2	3	4
음의 인덱스	-5	-4	-3	-2	-1

```
>>> word = “HELLO”
>>> word[0]          # 양의 인덱스
‘H’
>>> word[4]
‘O’
>>> word[-1]         # 음의 인덱스
‘O’
>>> word[-5]
‘H’
```

(3) 문자열 슬라이싱(Slicing)

문자열 인덱스를 이용하여 문자열의 일부분을 잘라내는 연산을 말한다. 문자열명[*시작번호 : 끝번호 : 간격*] 형태로 사용하며, *시작번호*부터 *끝번호*까지의 문자를 *간격* 만큼 건너뛰면서 뽑아내는

연산이다. 이때, 끝 번호는 슬라이싱된 결과에 포함되지 않으므로 주의해야 한다. *간격*은 생략되는 경우가 많으며, 생략될 시 디폴트값은 1 이다.

문자열 슬라이싱에는 양의 인덱스는 물론이고, 음의 인덱스도 사용할 수 있다. “Hello World” 문자열에 대한 인덱싱 결과는 아래 표와 같다.

	H	e	l	l	o		W	o	r	l	d
양의 인덱스	0	1	2	3	4	5	6	7	8	9	10
음의 인덱스	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

문자열 슬라이싱 연산 시, [시작번호:끝번호:간격] 지정 방법에 따라 결과가 어떻게 차이가 있는지를, 문자열 s 변수 슬라이싱 예를 통해 확인해보자.

슬라이싱 예	의미
s[a : b]	인덱스 a부터 b-1 까지의 문자열(a 〈 b)를 말한다.
s[a : b : c]	a 〈 b 이고 c 〉 0 이면, a부터 b-1까지의 c 간격의 문자열을, a 〉 b 이고, c 〈 0 이면 a부터 b + 1까지의 c 간격의 문자열을 말함
s[0 : : 1]	s[0]에서부터 문자열 끝까지의 문자열, 즉 s 문자열 전체를 의미
s[-1 : : -1]	s[-1] 에서부터 문자열 처음까지 -1 간격의 문자열, 즉 s 문자열 거꾸로 전체를 의미

양의 인덱스를 이용한 슬라이싱	음의 인덱스를 이용한 슬라이싱

>>> a = “Hello World”
>>> a[0:5] # 앞에서 5개의 문자열 추출
‘Hello’
>>> a[6:] # 6번에서 맨 마지막까지 추출
‘World’
>>> a[:5] # 맨 앞에서 4번까지 추출
‘Hello’
>>> a[::] # 처음부터 전체 문자열 추출
'Hello World’
>>> a[0::2] # 처음부터 1개 간격으로 추출
'HloWrd'
	>>> a = “Hello World”
	>>> a[-11:-6] # 앞에서 5개의 문자열 추출
	‘Hello’
	>>> a[-5:] # 6번에서 맨 마지막까지 추출
	‘World’
	>>> a[:-6] # 맨 앞에서 4번까지 추출
	‘Hello’
	>>> a[-1:-6:-1] # -1~-5번까지 거꾸로 추출
	'dlroW’
	>>> a[-1: :-1] #전체 문자열 거꾸로 추출
	'dlroW olleH'

💻 실습 예제 3. 문자열에서 정보 추출하기

주민등록번호에서 생년월일을 추출하는 프로그램을 작성해보자. 먼저 문제해결을 위한 알고리즘을 간단하게 정리해보자.

① pid 변수에 주민등록번호 저장
② 문자열 슬라이싱으로 첫 번째 문자부터 5번째 문자까지 추출하여 birthday 변수에 저장
③ birthday 출력

프로그램	실행 결과

birthday.py - D:/파이썬실습/birthday.py (3.11.4)
File Edit Format Run Options Window Help
pid = "000726-4401234"
birth = pid[:6] # 맨 앞부터 5번 인덱스까지 추출
print(birth)
	IDLE Shell 3.11.4
	File Edit Shell Debug Options Window Help
	Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023
	934 64 bit (AMD64)] on win32
	Type "help", "copyright", "credits" or "license()" fo
	>>>
	= RESTART: D:/파이썬실습/birthday.py
	000726
	>>>

❑ 문자열 포맷팅(Formating)

미리 내용이 정해진 문자열은 따옴표로 묶어서 문자열을 바로 정의할 수 있지만, 변수나 수식이 포함된 문자열을 만들기 위해서는 문자열 포맷팅 기능을 이용해야만 한다. 문자열 포맷팅은 **%서식** 또는 **f 문자열 포맷팅** 등을 이용할 수 있다.

(1) % 서식을 이용한 포맷팅

'%' 문자를 이용하여 문자열을 원하는 형식으로 포맷팅하는 방법으로 **'형식지정자' % 값** 의 형식으로 사용된다. 아래 예에서 "답은 %d 입니다." 부분이 포맷 문자열로 %d 부분에 정수 데이터가 들어갈 예정이고, result 값이 들어가게 된다.

정수 포맷팅 예	실수 포맷팅 예
x = 21 y = 35 result = x * y print("답은 %d 입니다." % result)	x = 2.1 y = 3.5 result = x * y print("답은 %f 입니다." % result)
실행 결과	**실행 결과**
답은 735 입니다.	답은 7.350000 입니다.

% 서식에서 포맷 문자열에 사용되는 자료형별 형식 지정자는 다음과 같다.

자료형	문자	사용할 수 있는 값
정수	%d	숫자(정수, 실수)
실수	%f	숫자(정수, 실수)
문자열	%s	문자열, 숫자
8진수	%o	정수
16진수	%x	정수
% 문자	%%	문자 %

두 개 이상의 값을 포맷 문자열에 넣으려면, 값을 ( )로 묶어주어야 한다.

두 개 이상 값에 대한 % 서식 포맷팅 예

```
x = 25
y = 35
result = x * y
print("%d X %d = %d 입니다." % (x,y, result))
```

실행 결과

```
25 X 35 = 875 입니다.
```

```
name = "김철수"
age = 20
print("%s는 %d세 입니다." % (name, age))
```

실행 결과

```
김철수는 20세 입니다.
```

(2) f 문자열 포맷팅

문자열 앞에 접두사인 f를 붙이고 중괄호 { } 안에 변수명을 넣어서 문자열을 만드는 방식이다. 아래 예제는 위와 동일한 예제를 f 문자열 포맷팅으로 표현한 것이다. f 문자를 앞에 붙이고 변수가 들어갈 위치에 중괄호 안에 변수명을 넣어주면, 알아서 변수의 값이 치환되어 문자열이 생성된다.

두 개 이상 값에 대한 f 문자열 포맷팅 예

```
x = 21
y = 35
result = x * y
print(f"{x} X {y} = {result} 입니다.")
```

실행 결과

```
21 X 35 = 735 입니다.
```

```
name = "김철수"
age = 20
print(f"{name}는 {age}세 입니다.")
```

실행 결과

```
김철수는 20세 입니다.
```

STEP 04

사용자 입력과 출력

컴퓨터 프로그램은 사용자로부터 데이터를 입력받아 연산을 수행하여 모니터와 같은 출력장치에 결과를 출력한다. 사용자는 키보드나 마우스를 통해 데이터를 입력할 수 있는데, 이 절에서는 가장 기본적인 키보드로부터 데이터를 입력받고 출력하는 방법에 대해서 다루기로 한다.

❑ 사용자 인터페이스

컴퓨터 프로그램과 사람간에 대화를 위해서는 사용자 인터페이스(User Interface)가 필요하다. 사용자 인터페이스는 그래픽 사용자 인터페이스(GUI : Graphical UserInterface)와 텍스트 기반 인터페이스가 있다. 파이썬에서 그래픽 사용자 인터페이스를 사용하기 위해서는 tkinter 등의 추가 모듈을 사용해야만 한다. 파이썬 쉘은 텍스트 기반 인터페이스에 해당하고, 이 책에서는 텍스트 기반 인터페이스를 사용하는 방법에 대해서만 다루기로 한다.

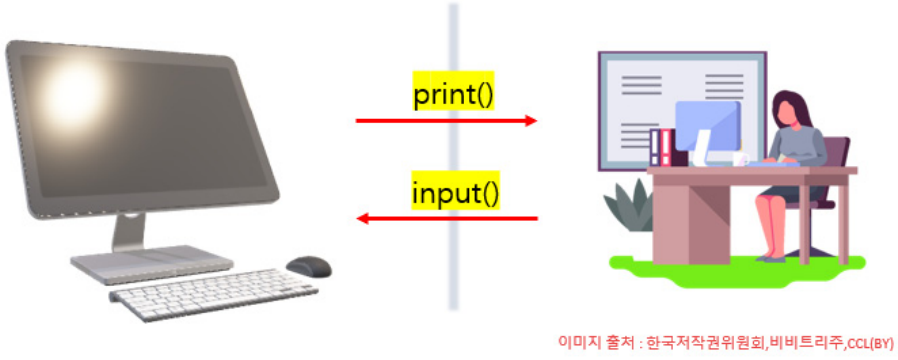


그림 24 텍스트 기반 인터페이스 방식

텍스트 기반 인터페이스에서는 모니터에 결과를 텍스트 형식으로 보여주고, 사용자가 키보드를 통해 입력한 문자 정보를 컴퓨터에 전달하는 방법으로 사용자-컴퓨터간 대화가 이루어진다. 지금까지 실습에서 사용했던 print() 함수가 텍스트로 화면에 결과를 출력해주는 함수이며, 반대로 사용자로부터 키보드 입력을 처리하는 기능은 input() 함수를 통해 제공된다.

❑ print() 함수

이미 앞에서 사용해왔던 print() 함수는 사용자가 원하는 형태로 화면에 텍스트를 출력해주는 함수로 문자열 포맷팅 기능을 이용하면 좀 더 다양한 형태로 출력 형식을 제어할 수 있다.

(1) 기본 출력

print() 함수의 가장 기본적인 사용 형태는 출력하고자 하는 값이나 변수명을 콤마(,)로 구분하여 순서대로 괄호안에 인수로 넣어주는 것이다. 여러개의 값을 콤마(,)로 구분하여 print() 함수를 호출하면, 값들 사이에 한 개의 공백이 자동으로 삽입되어 출력된다.

예제	
print("Hello World!")	# 화면에 "Hello World!" 문자열 출력
print(100)	# 화면에 100 출력
print(x, y)	# 화면에 변수 x의 값과 y의 값을 공백으로 구분하여 출력
print("합은", x+y, "입니다")	# "합은 10 입니다" 와 같이 x+y 연산의 값을 포함하여 출력

(2) 문자열 포맷팅을 이용한 출력 제어

앞절에서 설명한 %서식이나 f 문자열 포맷팅을 이용하여 출력 결과를 원하는 형태로 포맷팅할 수 있다. 다음은 동일한 출력 내용을 서로 다른 문자열 포맷팅을 이용하여 출력하는 예제 코드이다.

% 서식 문자열 포맷팅 예	f 문자열 포맷팅 예
name = "김철수" age = 20 print("%s는 %d세 입니다." % (name, age))	name = "김철수" age = 20 print(f"{name}는 {age}세 입니다.")
실행 결과	
김철수는 20세 입니다.	

❑ input() 함수

input() 함수는 기본적으로 문자열로 데이터를 입력받으므로, 정수나 실수 등 다른 자료형으로 입력 데이터를 다루기 위해서는 반드시 해당 자료형으로 형 변환(type conversion)이 필요하다.

input() 함수 사용법
[형식] 변수 = input("*입력 안내 메시지*") **[설명]** *입력 안내 메시지* 를 출력하고, 사용자가 입력한 값을 문자열로 반환한다. **[주의]** 숫자로 입력받아야 할 경우, 해당 숫자형으로 형 변환이 필요하다.

예제 코드	실행 결과
id = input("학번을 입력하시오 : ") num = int(input("정수를 입력하시오: ")) value = float(input("실수를 하나 입력하시오: ")) print("학번", id) print("정수", num, "실수", value)	학번을 입력하시오 : 2023-12345 정수를 입력하시오: 10 실수를 하나 입력하시오: 10.5 학번 2023-12345 정수 10 실수 10.5

위 예제 코드에서 정수값 입력 시, input()으로 문자열을 입력받은 후, 다시 int() 함수로 감싸주어서 문자열을 정수값으로 변환하는 절차가 수행되어, num 변수에는 정수값이 저장된다. 실수값 입력도 int() 대신 float()로 감싸주는 부분만 다르고 동일한 방식으로 처리된다.

다음은 두 개의 숫자를 입력받아서 덧셈 결과를 출력하는 프로그램으로 정수 변환을 안할 경우 발생하는 문제점을 예제를 통해 확인해볼 수 있다. [예제 1]에는 두 개의 입력 모두 정수가 아닌 문자열로 처리되어 변수 x, y 에 각각 저장된 후, '+' 연산이 문자열 연결 연산자로 해석되어 숫자 덧셈이 아닌 x, y 에 저장된 문자열이 연결된다.

[예제1] 잘못된 input() 사용	실행 결과
x = input("정수1 입력 : ") y = input("정수2 입력 : ") sum = x +y print("합 =", sum)	정수1 입력 : 10 정수2 입력 : 20 합 = 1020

[예제 2]에는 문자열을 입력받은 후, int 변환을 통해 정수로 변환된 값이 x, y 변수에 저장되어 두 수의 합이 계산되어 sum 에 저장되어, 정상적으로 결과가 출력된다.

[예제2] 잘된 input() 사용	실행 결과
x = int(input("정수1 입력 : ")) y = int(input("정수2 입력 : ")) sum = x + y print("합 =", sum)	정수1 입력 : 10 정수2 입력 : 20 합 = 30

💻 실습 예제 4. 이름 입력과 출력

여권에 이름을 출력하기 위해 firstname 과 lastname 을 사용자로부터 각각 입력받아 두 개의 이름을 연결하여 출력하여 보자.

① firstname 입력
② lastname 입력
③ firstname과 lastname을 '+' 연산자로 연결하여 출력

프로그램	실행 결과
firstname = input("영문 이름을 입력하세요:") lastname = input("영문 성을 입력하세요:") print("당신의 여권 이름은", firstname+" " + lastname)	영문 이름을 입력하세요:Miseon 영문 성을 입력하세요:Choi 당신의 여권 이름은 Miseon Choi

3장

프로그램 제어구조

제어문이란?

STEP 01

컴퓨터 프로그램의 실행 흐름을 제어하는 문장을 제어문이라고 하며, 제어문은 순차, 선택, 반복의 3 가지 기본 제어구조를 표현할 수 있다.

❑ 3 가지 기본 제어구조

프로그램의 실행 순서를 제어하기 위한 제어구조는 다음과 같이 순차구조, 선택구조, 반복구조로 나누어진다.

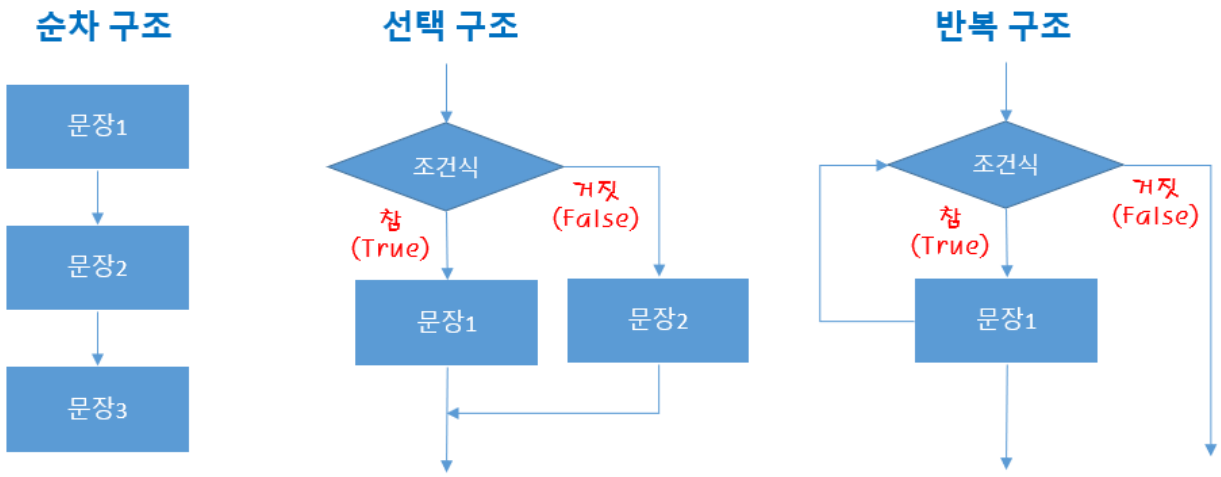


그림 25 3가지 기본 제어 구조

순차구조는 가장 기본적인 제어구조로 프로그램에 작성된 순서대로 문장들을 위에서 아래 방향으로 코드를 실행하는 구조이다. 선택구조는 조건식의 값을 판별하여 참(True)과 거짓(False)인 경우로 나누어 실행될 문장을 선택하는 구조이다. 반복구조는 조건식의 값이 참이면 문장을 반복해서 실행하고, 거짓이면 반복을 종료하는 구조이다.

❑ 3 가지 기본 제어구조와 파이썬 프로그래밍

대부분의 프로그래밍 언어와 마찬가지로 파이썬에서도 순차, 선택, 반복의 3 가지 제어구조를 표현할 수 있는 문장이 제공된다.

<table>
<tr><th>순차 구조 => 문장</th><th>선택 구조 => 조건 선택문</th></tr>
<tr><td>x = 10
print(x)
y = 20
print(y)
average = (x + y) / 2
print(average)</td><td>score = 85
if (score >= 80):
    print("합격")
else:
    print("불합격")</td></tr>
<tr><th colspan="2">반복 구조 => 반복문</th></tr>
<tr><td>age = 0
while (age <= 18):
    print("미성년자입니다.")
    age = age + 1
print("성년이 되신 것을 축하합니다. ")</td><td>for age in range(0, 19):
    print("%d 세=> 아직 미성년자" % age)
print("이제 성년이 되셨습니다.")</td></tr>
</table>

기본적으로 모든 문장은 순차구조로 위에서 아래 방향으로 실행된다. 선택 구조는 조건 선택문에 해당하는 if-else 문과 match 문으로 표현할 수 있다. 반복구조는 while 문이나 for 문으로 표현 가능하다. 각각의 제어문에 대해서 다음절부터 자세히 살펴보기로 하자.

조건식 만들기

STEP 02

조건식은 주어진 조건을 판별하여 참(True) 또는 거짓(False) 중 하나의 결과값을 생성하는 수식으로 프로그램 제어구조 중 선택, 반복 구조에서 실행 흐름을 바꾸는데 사용된다. 이 절에서는 관계연산자와 논리연산자를 사용하여 조건식을 만드는 방법에 대해서 알아보기로 하자.

❑ 조건식(conditional expression)이란?

조건식은 부울값(True/False)으로 평가 결과가 나오는 수식으로, 조건 선택문과 반복문에 사용되어 프로그램의 실행 흐름을 바꾸는데 중요한 역할을 한다. 조건식은 보통 관계연산자나 논리연산자를 포함하지만, 아래와 같이 값 자체로도 True, False 값이 정해지기도 한다.

값 예시	True / False 평가 값	설명
“abc”	True	문자열이 비어 있지 않으면 참
“”	False	문자열이 비어 있으면 거짓
[1,2,3]	True	리스트가 비어 있지 않으면 참
[]	False	리스트가 비어 있으면 거짓
(1, 2, 3)	True	튜플이 비어 있지 않으면 참
()	False	튜플이 비어 있으면 거짓
{1, “kim”,2,“lee”}	True	딕셔너리가 비어 있지 않으면 참
{}	False	딕셔너리가 비어 있으면 거짓
1	True	0이 아닌 모든 수는 참
0	False	0은 거짓
None	False	값이 존재하지 않음을 의미, 거짓

조건식을 만드는데 주로 사용되는 관계연산자나 논리연산자를 사용하는 방법에 대하여 좀 더 자세히 알아보기로 하자.

❑ 관계 연산자(Relational Operator)

피연산자들의 대소 관계에 따라 부울값(True, False)으로 결과값을 계산하는 연산자이다.

관계 연산자	의 미	관계 연산자 사용 예	
x 〉 y	x가 y보다 크다		
x 〈 y	x가 y보다 작다		
x == y	x와 y가 같다		
x != y	x와 y가 같지 않다		
x 〉= y	x가 y보다 크거나 같다		
x 〈= y	x가 y보다 작거나 같다		

```
>>> x = 5
>>> y = 3
>>> x > y
True
>>> x < y
False
```

```
>>> x == y
False
>>> x != y
True
>>> x >= y
True
>>> x <= y
False
```

❑ 논리 연산자(Logical Operator)

부울값들에 대해 연산을 수행하여 결과값으로 부울값(True, False)을 생성하는 연산자들로 관계식들을 연결하여 복잡한 조건을 표현하는데 주로 사용된다.

논리 연산자	사용 예	의 미
AND 연산	x and y	x와 y가 모두 True이면 True, 그렇지 않으면 False
OR 연산	x or y	x나 y중 하나만 True이면 True, 모두 False이면 False
NOT 연산	not x	x가 True 이면 False, False 이면 True

간단한 예시로 대학교에서 성적우수 장학금을 받기 위한 조건이 12 학점 이상 수강하고, 평점 3.7 이상을 다 만족하는 경우만 장학금을 받는 경우에 2 가지 조건을 AND 연산으로 연결하여 아래와 같이 표현할 수 있다. 논리 연산자는 and, or, not 과 같이 영문 소문자로만 사용해야 한다.

직전 학기에 12학점 이상 수강하고 평점이 3.7 이상이면
조건1 → 장학금을 받는다. 조건2

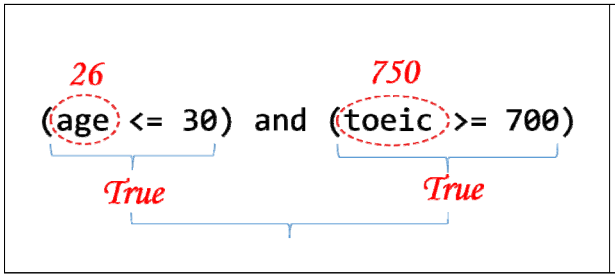
(직전 학기에 12학점 이상 수강) and (평점이 3.7 이상이면)
조건1 → 장학금을 받는다. 조건2

```
>>> credit = 18
>>> gpa = 4.0
>>> x = (credit >= 12)
>>> x
True
>>> y = (gpa >= 3.7)
>>> y
True
>>> x and y
True
```

💻 실습 예제 5. 합격 여부 판별하기

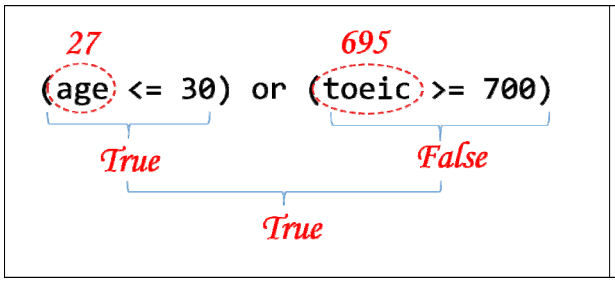
나이와 토익점수 두가지 조건을 AND 와 OR 연산으로 연결하여 합격 여부를 판단하는 간단한 예제를 살펴보자.

(1) 나이가 30 살 이하이고 토익이 700 점 이상이면 합격




```
>>> age = int(input("나이 입력 -> "))
나이 입력 -> 26
>>> toeic = int(input("토익점수 입력 ->"))
토익점수 입력 ->750
>>> (age <= 30) and (toeic >= 700)
True
```

(2) 나이가 30 살 이하이거나 토익이 700 점 이상이면 합격




```
>>> age = 27
>>> toeic = 695
>>> (age <= 30) or (toeic >= 700)
True
```

💻 실습 예제 6. 윤년 여부 판별하기

연도를 입력받아 윤년인지 여부를 판별하는 조건식을 작성해보자. 윤년 판별을 위해서는 우선 윤년 계산 규칙을 알아야 한다. 그레고리력의 윤년 계산 규칙은 다음과 같다.

- 연수가 4 로 나누어 떨어지는 해는 윤년으로 한다.
- 연수가 4, 100 으로 나누어 떨어지는 해는 평년으로 한다.
- 연수가 4, 100, 400 으로 나누어 떨어지는 해는 윤년으로 한다.

위 규칙에 따라 윤년을 계산하는 알고리즘을 먼저 정리해보자.

① 연도를 정수로 입력받아 year 변수에 저장한다.

② year 변수의 값을 4 로 나눈 나머지가 0 인지를 검사하는 조건식 1 을 작성한다.

③ year 변수의 값을 100 으로 나눈 나머지가 0 인지를 검사하는 조건식 2 을 작성한다.

④ year 변수의 값을 400 로 나눈 나머지가 0 인지를 검사하는 조건식 3 을 작성한다.

⑤ 조건식 1 AND NOT(조건식 2) 이면 윤년이다. 또는

⑥ 조건식 3 이 참이면 윤년이다.

위 알고리즘에 대한 프로그램을 아래와 같이 작성해볼 수 있다.

프로그램	실행 결과
year= int(input("연도를 입력하세요:")) cond1 = (year % 4 == 0) cond2 = (year % 100 == 0) cond3 = (year % 400 == 0) print((cond1 and not(cond2)) or cond3)	연도를 입력하세요:2000 True 연도를 입력하세요:2023 False

4장

조건 선택문

단일 if문 사용하기

STEP 01

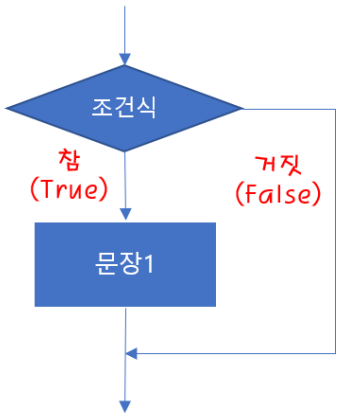
조건 선택문이란 프로그램의 3 가지 기본 제어구조 중에서 선택구조를 구현하는데 사용되는 문장으로 조건식의 값이 True 인지 False 인지에 따라 프로그램의 실행 순서가 결정된다.

❑ 파이썬 조건 선택문 종류

파이썬에서 제공되는 조건 선택문은 if 문과 match-case 문이 있다. if 문은 단일 조건문에 해당하는 if 문, if-else 문과 다중 조건문에 해당하는 if-elif-else 문이 제공된다. 파이썬 3.10 버전부터 새로이 추가된 match-case 문도 다중 조건문을 표현하는데 적합하며, Java 등 다른 프로그래밍 언어에서 제공되는 switch-case 문과 유사하다.

❑ 조건식이 참인 경우만 문장 수행하기 - if 문

조건식이 참일 경우에만 특정 작업을 수행하도록 하려면 아래와 같은 형식으로 if 문을 작성하면 된다. 조건식 다음에는 반드시 콜론(:)이 와야 하며, 다음 줄에는 들여쓰기를 지켜서 조건식이 참일 때 수행할 문장을 작성해야 한다.




```
if 조건식:
    수행할 문장1
```

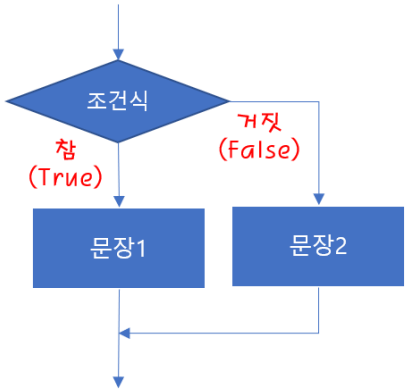
다음 예제는 신호등 색깔이 녹색일때만 길을 건너가는 상황을 간단하게 if 문으로 표현한 것이다. 신호등 색깔을 'R' 또는 'G'로 입력받아 signal 에 저장한 후, signal 이 'G'인 경우에 "건너간다"를 화면에 출력한다. 대문자 'G'와 소문자 'g'를 모두 처리하기 위해 논리 연산자 or 를 이용하여 조건을 연결하였다.

예제	프로그램 구현	실행 결과
신호등이 녹색이면, 건너간다.	signal = input("신호등 색깔(빨강: R 녹색 : G)? ") if signal == 'G' or signal == 'g' : print("건너간다")	신호등 색깔(빨강: R 녹색 : G)? G 건너간다

❑ 조건식이 참인 경우와 거짓인 경우 나누어서 문장 수행하기 - if-else 문

조건식이 참일 경우와 거짓인 경우로 나누어서 다른 작업을 수행하도록 하려면 아래와 같은 형식으로 if-else 문을 작성하면 된다.

```
if 조건식 :
    참일 때 수행할 문장
else :
    거짓일 때 수행할 문장
```



다음 예제는 신호등 색깔이 녹색일 때 길을 건너고, 그렇지 않을 때 멈추는 상황을 간단하게 if-else 문으로 표현한 것이다. 신호등 색깔을 'R' 또는 'G'로 입력받아 signal 에 저장한 후, signal 이 'G'인 경우에 "건너간다", 그렇지 않은 경우에 "멈춘다"를 화면에 출력한다.

예제	프로그램 구현	실행 결과
신호등이 녹색이면, 건너간다. 그렇지않으면, 멈춘다.	signal = input("신호등 색깔(빨강: R 녹색 : G)? ") if signal == 'G' or signal == 'g' : print("건너간다") else : print("멈춘다")	신호등 색깔(빨강: R 녹색 : G)? g 건너간다 신호등 색깔(빨강: R 녹색 : G)? R 멈춘다

3 장에서 살펴보았던 장학금 수여 여부를 판별하는 조건식을 이용하여 if 문으로도 간단하게 구현해보자.

(직전 학기에 12학점 이상 수강) **and** (평점이 3.7 이상이면)
조건1 → 장학금을 받는다. 조건2

프로그램	실행 결과

credit = int(input("직전학기 수강 학점 입력: "))
gpa = float(input("직전학기 평균 평점 입력 : "))
if (credit >= 12) and (gpa >= 3.7) :
  print("장학금을 받습니다.")
else :
  print("좀 더 노력이 필요합니다.")
	직전학기 수강 학점 입력: 15
	직전학기 평균 평점 입력 : 3.85
	장학금을 받습니다.

💻 실습 예제 1. 주민번호 5 부제

주민번호 5 부제(출생년도 끝자리)는 생년월일 끝자리를 5 로 나누어 나머지 값에 따라 요일을 나누어서 코로나 백신 등 국가 예방접종을 신청할 수 있도록 하는 방식이다. 예를 들어, 월요일은 출생년도 끝자리가 1 이거나 6 인 사람만 신청을 할 수 있다. 오늘이 월요일이라고 가정하고, 출생년도를 입력하면 신청 대상인지 아닌지 여부를 알려주는 메시지를 출력해보자.

(출생년도 끝자리가 1) **or** (출생년도 끝자리가 6)
조건1 → 오늘 백신 접종 예약을 할 수 있다. 조건2

① 출생년도를 문자열로 입력받아 year 변수에 저장한다.
② year 변수에 저장된 문자열에서 인덱싱을 이용하여 끝자리를 추출하여 digit 변수에 저장한다.
③ digit 가 “1” 또는 “6”이면 “오늘 백신 접종 예약을 할 수 있다”를 출력
④ 그렇지 않으면, “오늘 백신 접종 예약을 할 수 없다”를 출력

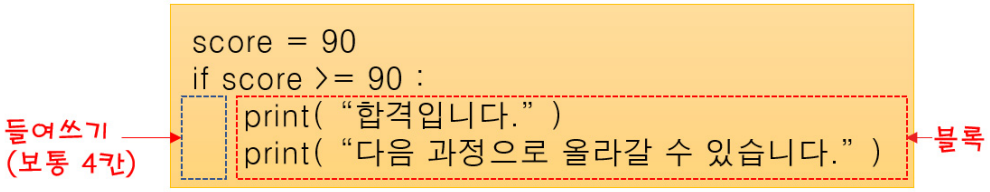
이를 다음과 같이 프로그램으로 구현할 수 있다. 입력받은 출생년도가 저장된 year 에서 끝자리를 추출하기 위해 year[-1]과 같이 음의 인덱싱을 이용하였다.

프로그램	실행 결과

year= input("출생년도 4자리를 입력하세요:")
digit = year[-1]
if digit == "1" or digit == "6" :
  print("오늘 백신 접종 예약을 할 수 있습니다.")
else :
  print("오늘 백신 접종 예약을 할 수 없습니다.")
	출생년도 4자리를 입력하세요: 2000
	오늘 백신 접종 예약을 할 수 없습니다.
	
	출생년도 4자리를 입력하세요:2001
	오늘 백신 접종 예약을 할 수 있습니다.

❑ 블록(Block)과 들여쓰기

if 문 사용 시, 조건이 참이거나 거짓일 경우 실행해야 할 대상 문장이 두 개 이상인 경우는 if 문의 영향을 받는 범위를 블록으로 묶어주어야 한다. 블록이란 관련있는 문장들의 집합으로 다른 프로그래밍 언어들과는 다르게 파이썬에서는 들여쓰기를 이용하여 블록의 범위를 표시한다. 다음 예제와 같이 같은 들여쓰기 레벨에 있는 문장들은 같은 블록에 속한다. 한 레벨을 나타내는 들여쓰기 칸수는 보통 4 칸을 많이 사용한다.



파이썬에서 들여쓰기는 단순한 가독성 향상 목적이 아닌 블록의 범위를 표시하는 중요한 역할을 수행하므로 들여쓰기를 잘못하면 프로그램의 실행 흐름이 달라질 수 있어 주의해야만 한다.

다음은 들여쓰기를 잘못했을 때 예상치 못한 결과가 나오는 예시이다. 첫 번째 프로그램에서는 2 개의 print 문의 하나의 블록에 포함되어 score 가 90 이상일 때 두 문장이 모두 출력된다. 두 번째 프로그램에서는 들여쓰기를 잘못하여 if 문이 참일 때 실행하는 블록의 범위가 첫 번째 print 문만 포함하고 있다. 그 결과 두 번째 print 문이 if 문 블록에 포함되지 않아 score 값에 상관없이 항상 출력되는 잘못된 결과가 나온다.

프로그램	실행 결과

score = 90
if score >= 90 :
  print("합격입니다.")
  print("다음 과정으로 올라갈 수 있습니다.")
	합격입니다.
	다음 과정으로 올라갈 수 있습니다.

score = 80 if score >= 90 : print("합격입니다.") print("다음 과정으로 올라갈 수 있습니다.") 들여쓰기가 잘못되어 블록이 잘못 지정됨	다음 과정으로 올라갈 수 있습니다.

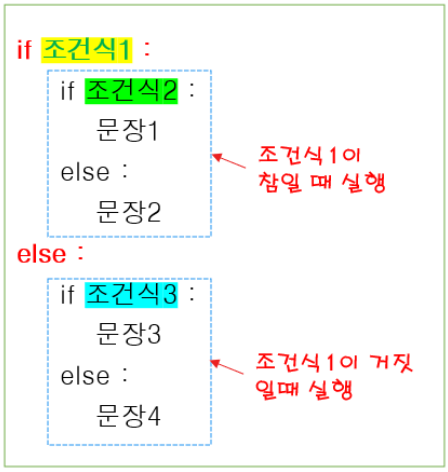
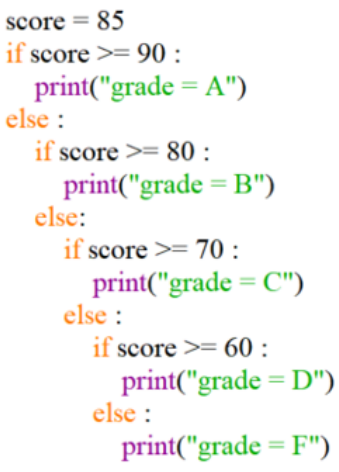
중첩 if문과 연속 if문

STEP 02

조건 선택문에서 여러 가지 조건을 판별하여 다른 처리를 해야하는 경우에, 중첩 if 문 또는 연속 if 문을 사용하여 복합적인 조건이 포함된 다중 선택 구조를 표현할 수 있다.

❑ 중첩 if 문(Nested if Statement)

if 문도 하나의 문장이므로 문장이 올 수 있는 자리이면 if 문도 올 수 있으므로 if 문안에 if 문을 중첩시켜서 복잡한 조건을 표현하는 것도 가능하다. 다만, if 문 중첩이 많아지면 구조가 복잡해지는 문제가 있으므로 가능하면 다음절에 나오는 연속 if 문(if~elif)으로 해결하는 것이 좋다.

중첩 if문 형식	중첩 if문 예제
	


```
score = 85
if score >= 90 :
   print("grade = A")
else :
   if score >= 80 :
      print("grade = B")
   else:
      if score >= 70 :
         print("grade = C")
      else :
         if score >= 60 :
            print("grade = D")
         else :
            print("grade = F")
```

왼쪽은 중첩 if 문의 형식을 간단하게 표현한 것으로 중첩 if 문의 구조가 잘 나타나 있다. 가장 바깥쪽 조건식 1 의 값을 판별하여 참인 경우에 다시 if 문을 만나서 조건식 2 의 값을 판별하여 참/거짓에 따라 문장 1 또는 문장 2 를 실행한다. 조건식 1 이 거짓인 경우에는 조건식 3 의 값에 따라 문장 3 또는 문장 4 를 수행한다.

오른쪽은 성적에 따라 절대평가 방식으로 학점이 부여되는 절차를 중첩 if 문으로 표현한 것이다. 가장 먼저 score 의 값이 90 점 이상인지를 판별하여 A 학점을 출력하고, 90 점 미만인 경우를 또

다시 80 점 이상과 80 점 미만으로 구분한다. 80 점 이상인 경우는 B 학점을 출력하고, 80 점 미만인 경우를 다시 70 점 이상과 70 점 미만으로 나눈다. 70 점 이상인 경우 C 학점을 출력하고, 70 점 미만인 경우 60 점 이상과 그 외의 경우로 나누어서 각각 D 학점과 F 학점을 출력한다.

❑ 연속 if 문 : if~elif 문

중첩 if 문은 if 문 중첩이 많아지면 구조가 매우 복잡해진다. 이를 개선할 수 있는 방법이 연속 if 문으로 조건식의 값에 따라 여러 케이스로 나누어서 다른 문장을 실행하도록 만들 수 있다.

연속 if 문은 if 문 다음에 첫 번째 조건식을 적고 나머지 조건식들은 elif 문 다음에 적는 방식으로 여러 가지 조건을 나누어서 표현한다. 마지막으로 어느 조건에도 해당 안되는 경우는 else 문으로 처리한다. else 문은 필요없는 경우에는 생략 가능하다. 연속 if 문 형식에서 if 문 안에 여러 문장이 블록으로 올 수 있는 것을 표현하기 위해 수행할 문장 1-1, 수행할 문장 1-2 와 같이 표현하였다.

연속 if문 형식	연속 if문 예제	중첩 if문 예제

if 조건식1 :
  수행할 문장1-1
  수행할 문장1-2
  …
elif 조건식2 :
  수행할 문장2-1
  수행할 문장2-2
  …
…
else :
  수행할 문장n-1
  수행할 문장n-2
  …
	score = 85
	if score >= 90 :
	  print("grade = A")
	elif score >= 80 :
	  print("grade = B")
	elif score >= 70 :
	  print("grade = C")
	elif score >= 60 :
	  print("grade = D")
	else :
	  print("grade = F")
		score = 85
		if score >= 90 :
		  print("grade = A")
		else :
		  if score >= 80 :
		  print("grade = B")
		  else:
		  if score >= 70 :
		  print("grade = C")
		  else :
		  if score >= 60 :
		  print("grade = D")
		  else :
		  print("grade = F")

앞에서 들었던 학점 부여 예시를 연속 if 문과 중첩 if 문으로 구현한 결과를 비교해보자. 중첩 if 문보다 연속 if 문이 훨씬 간단하고 구조적으로 표현된 것을 확인할 수 있을 것이다.

💻 실습 예제 2. 음료 주문하기

카페에서 키오스크를 통해 음료를 선택하여 주문하는 프로그램을 텍스트 기반 메뉴로 간단하게 구현하여 보자. 카페의 메뉴는 아메리카노(3000 원), 카페라떼(4000 원), 카페모카(4000 원),

흑당밀크티(5000 원) 4 종이 제공된다고 가정하자. 고객이 여러 음료 중에서 어떤 것을 선택했는지를 검사하는 다중 조건문 문제이므로 if~elif 문으로 해결할 수 있다.

프로그램 구현 전에 알고리즘을 먼저 정리해보자.

① 화면에 메뉴를 출력한다.
② 사용자에게 주문할 메뉴번호를 정수로 입력 받아 menuid 변수에 저장한다.
③ menuid 가 1 번이면 아메리카노 주문 확인 메시지와 가격을 출력한다.
④ menuid 가 2 번이면 카페라떼 주문 확인 메시지와 가격을 출력한다.
⑤ menuid 가 3 번이면 카페모카 주문 확인 메시지와 가격을 출력한다.
⑥ menuid 가 4 번이면 흑당밀크티 주문 확인 메시지와 가격을 출력한다.
⑦ 1~4 이외의 값이 입력된 경우, 잘못 주문했다는 메시지를 출력한다.

위 알고리즘에 대한 프로그램을 아래와 같이 작성해볼 수 있다. 단, 이 프로그램은 한번 실행 시, 단 한번의 주문만 가능하다. 연속적으로 주문이 가능하도록 하려면 다음 장에서 배울 반복문을 활용하여 예시된 프로그램 전체를 감싸주어야 한다.

프로그램

```
# order_drink.py

print("******* 메뉴 *******")
print("1. 아메리카노 : 3000원")
print("2. 카페라떼 : 4000원")
print("3. 카페모카 : 4000원")
print("4. 흑당밀크티 : 5000원")

menuid = int(input("주문할 음료 번호를 말씀해주세요 > "))
if menuid == 1 :
    print("아메리카노를 주문하셨습니다.")
    print("3000원을 결제합니다");
elif menuid == 2 :
    print("카페라떼를 주문하셨습니다.")
    print("4000원을 결제합니다");
elif menuid == 3 :
    print("카페모카를 주문하셨습니다.")
    print("4000원을 결제합니다");
elif menuid == 4 :
    print("흑당밀크티를 주문하셨습니다.")
    print("5000원을 결제합니다");
else :
    print("잘못된 주문입니다.")
    print("처음부터 다시 시작해주세요.")
```

실행 결과

```
******* 메뉴 *******
1. 아메리카노 : 3000원
2. 카페라떼 : 4000원
3. 카페모카 : 4000원
4. 흑당밀크티 : 5000원
주문할 음료 번호를 말씀해주세요 > 2
카페라떼를 주문하셨습니다.
4000원을 결제합니다


******* 메뉴 *******
1. 아메리카노 : 3000원
2. 카페라떼 : 4000원
3. 카페모카 : 4000원
4. 흑당밀크티 : 5000원
주문할 음료 번호를 말씀해주세요 > 4
흑당밀크티를 주문하셨습니다.
5000원을 결제합니다


******* 메뉴 *******
1. 아메리카노 : 3000원
2. 카페라떼 : 4000원
3. 카페모카 : 4000원
4. 흑당밀크티 : 5000원
주문할 음료 번호를 말씀해주세요 > 5
잘못된 주문입니다.
처음부터 다시 시작해주세요.
```

STEP 03

match 문

파이썬 3.10 버전부터 추가된 match 문은 패턴 매칭과 관련된 기능을 제공하는 구문으로, if-elif 블록을 사용하여 다양한 조건을 처리했던 상황을 더 간결하고 가독성 좋게 처리할 수 있다. C, Java, JavaScript 등의 switch 문과 유사하지만 좀 더 강력하고 다양한 기능을 제공한다.

❑ match 문(match statement)이란?

match 문은 주어진 값과 패턴을 비교하여 어떤 블록을 실행할지 결정한다. 값이 여러 패턴 중 어느 패턴과 일치하는지 확인하고 해당되는 코드 블록을 실행한다. match 문은 값을 비교하는 데 여러 가지 패턴을 사용할 수 있는데, 이 패턴들은 값의 구조나 특성에 따라 다양한 형태로 정의될 수 있다. 이 책에서는 match 문의 기본적인 사용법 위주로 살펴보기로 하자.

❑ match 문 기본 사용 예제

match 문의 가장 간단한 형태는 다음 예와 같이 조건을 검사하고자 하는 값을 한 개 이상의 상수와 비교하는 것이다.

프로그램	실행 결과

grade = int(input("초등학교 학년을 입력하세요 : "))
match grade :
  case 1:
  print("저학년 입니다.")
  case 2:
  print("저학년 입니다.")
  case 3:
  print("저학년 입니다.")
  case 4:
  print("고학년 입니다.")
  case 5:
  print("고학년 입니다.")
  case 6:
  print("고학년 입니다.")
  case _:
  print("잘못 입력하셨습니다")
	초등학교 학년을 입력하세요 : 1
	저학년 입니다.
	
	초등학교 학년을 입력하세요 : 4
	고학년 입니다.
	
	초등학교 학년을 입력하세요 : 7
	잘못 입력하셨습니다

grade 변수값을 case 다음 상수값과 비교하여 매치되는 블록을 실행하고 match 문을 빠져나온다. 매치되는 case 가 없으면 match 문 내의 어떤 문장도 실행되지 않는다. 모든 경우를 처리하려면, 위의 예제와 같이 '_'(언더스코어)를 case 상수로 사용하면 된다. '_'(언더스코어)는 모든 값과 매치되는 와일드카드(wildcard) 문자로 앞의 case 에 매치되지 않는 모든 값과 매치된다.

case 다음에 나오는 상수는 |("or") 기호를 사용하여 여러 상수를 하나로 연결할 수 있다. 위의 예시에서 1~3 학년은 저학년, 4~6 학년은 고학년에 해당하므로, 여러 케이스를 연결하여 아래와 같이 좀 더 간결하게 코드를 작성할 수 있다.

프로그램	실행 결과

grade = int(input("초등학교 학년을 입력하세요 : "))
match grade :
  case 1 | 2 | 3:
  print("저학년 입니다.")
  case 4 | 5 | 6:
  print("고학년 입니다.")
  case _:
  print("잘못 입력하셨습니다")
	초등학교 학년을 입력하세요 : 2
	저학년 입니다.
	
	초등학교 학년을 입력하세요 : 5
	고학년 입니다.
	
	초등학교 학년을 입력하세요 : 0
	잘못 입력하셨습니다

💻 실습 예제 3. match 문으로 구현한 음료 주문하기

[실습 예제 2]에서 다루었던 음료 주문하기 예제를 match 문으로도 구현해보자.

프로그램	실행 결과

print("******* 메뉴 *******")
print("1. 아메리카노 : 3000원")
print("2. 카페라떼 : 4000원")
print("3. 카페모카 : 4000원")
print("4. 흑당밀크티 : 5000원")
 	
menuid = int(input("주문할 음료 번호를 말씀해주세요 > "))
match menuid :
  case 1 :
  print("아메리카노를 주문하셨습니다.")
  print("3000원을 결제합니다");
  case 2 :
  print("카페라떼를 주문하셨습니다.")
  print("4000원을 결제합니다");
  case 3 :
  print("카페모카를 주문하셨습니다.")
  print("4000원을 결제합니다");
  case 4 :
  print("흑당밀크티를 주문하셨습니다.")
  print("5000원을 결제합니다");
  case _ :
  print("잘못된 주문입니다.")
  print("처음부터 다시 시작해주세요.")
	******* 메뉴 *******
	1. 아메리카노 : 3000원
	2. 카페라떼 : 4000원
	3. 카페모카 : 4000원
	4. 흑당밀크티 : 5000원
	주문할 음료 번호를 말씀해주세요 > 2
	카페라떼를 주문하셨습니다.
	4000원을 결제합니다
	
	******* 메뉴 *******
	1. 아메리카노 : 3000원
	2. 카페라떼 : 4000원
	3. 카페모카 : 4000원
	4. 흑당밀크티 : 5000원
	주문할 음료 번호를 말씀해주세요 > 4
	흑당밀크티를 주문하셨습니다.
	5000원을 결제합니다
	
	******* 메뉴 *******
	1. 아메리카노 : 3000원
	2. 카페라떼 : 4000원
	3. 카페모카 : 4000원
	4. 흑당밀크티 : 5000원
	주문할 음료 번호를 말씀해주세요 > 5
	잘못된 주문입니다.
	처음부터 다시 시작해주세요.

5장

반복문

STEP 01

반복문이란?

반복문(Repitition statement)은 프로그램 제어구조 중 반복구조를 표현하는데 사용되는 문장으로 주어진 조건이 만족하는 동안 블록안의 문장을 반복해서 실행한다. 프로그램에서 반복되는 패턴을 묶어서 반복문으로 표현하면 보다 간결한 프로그램을 작성할 수 있다.

❑ 반복구조의 필요성

반복구조가 필요한 이유를 아래의 꽃 그리기 예제를 통해 살펴보자. 이 예제는 파이썬코드가 아닌 의사코드로 작성되었다. 순차구조만을 사용할 경우 20 픽셀 길이 선분 8 개로 이루어진 꽃잎 1 장을 그리려면 24 줄의 코드가 필요하다. 3 줄의 코드가 8 번 반복되므로 오른쪽 상단과 같이 반복구조를 사용하여 4 줄로 줄일 수 있다. 여러 장의 꽃잎을 그리는 것 역시 꽃잎 한 장 그리는 반복구조를 꽃잎의 개수만큼 반복하면 쉽게 그릴 수 있다. 순차구조만을 사용할 경우 6 장의 꽃잎을 그리려면 적어도 24 줄 X 6 장 만큼의 코드가 필요한 반면, 반복구조는 6 줄의 코드로 미션을 해결할 수 있다.

<table>
<tr><th>꽃 그리기</th><th>꽃잎 한 장 그리기(순차구조)</th><th>꽃잎 한 장 그리기(반복구조)</th></tr>
<tr><td rowspan="3">  </td><td rowspan="3">색 바꾸기
앞으로 20픽셀 이동하기
오른쪽으로 30도 돌기
색 바꾸기
앞으로 20픽셀 이동하기
오른쪽으로 30도 돌기
색 바꾸기
앞으로 20픽셀 이동하기
오른쪽으로 30도 돌기
색 바꾸기
앞으로 20픽셀 이동하기
오른쪽으로 30도 돌기
색 바꾸기
앞으로 20픽셀 이동하기
오른쪽으로 30도 돌기
색 바꾸기
앞으로 20픽셀 이동하기
오른쪽으로 30도 돌기
색 바꾸기
앞으로 20픽셀 이동하기
오른쪽으로 30도 돌기
색 바꾸기
앞으로 20픽셀 이동하기
오른쪽으로 30도 돌기</td><td>8번 반복하기
    색 바꾸기
    앞으로 20픽셀 이동하기
    오른쪽으로 30도 돌기</td></tr>
<tr><th>꽃 한송이(꽃잎 6장) 그리기</th></tr>
<tr><td>6번 반복하기
    8번 반복하기
        색 바꾸기
        앞으로 20픽셀 이동하기
        오른쪽으로 30도 돌기
    오른쪽으로 60도 돌기</td></tr>
</table>

❑ 파이썬 반복문 종류

파이썬 반복문으로는 while 문과 for 문이 제공된다. while 문은 조건 반복문이라 부르며, 주어진 조건이 만족되는 동안 while 문 블록안의 문장을 반복해서 수행한다. 횟수 반복문이라고도 부르는 for 문은 정해진 횟수만큼 블록 내의 문장을 반복 수행한다.

아래는 “트랙을 한바퀴 도세요!”를 1000 번 출력하는 프로그램을 순차구조와 while 문과 for 문을 사용한 반복구조로 표현한 것이다. 이 예에서는 1000 번 횟수만큼 문장 블록을 반복 실행하는 문제이므로, 횟수 반복문인 for 문을 이용하여 작성하는 것이 더 편리하다. while 문과 for 문에 대해서는 이후의 절에서 자세히 살펴보기로 한다.

<table>
<tr><th>순차 구조로 표현</th><th>조건 반복문</th></tr>
<tr><td rowspan="3">print(“트랙을 한바퀴 도세요!")
print("트랙을 한바퀴 도세요!")
print("트랙을 한바퀴 도세요!")
print("트랙을 한바퀴 도세요!")
print("트랙을 한바퀴 도세요!")
…</td><td>count = 0
while count 〈 1000 :
    print("트랙을 한바퀴 도세요!")
    count = count + 1</td></tr>
<tr><th>횟수 반복문</th></tr>
<tr><td>for i in range(1000):
    print("트랙을 한바퀴 도세요!")</td></tr>
</table>

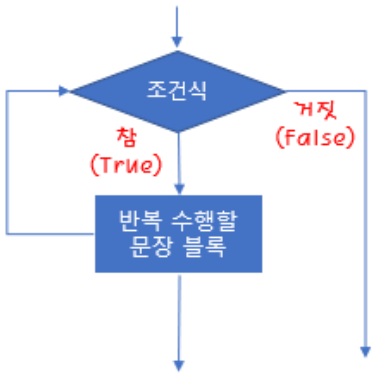
조건 반복(while문)

STEP 02

파이썬에서는 조건 반복문으로 while 문을 제공한다. while 문은 주어진 조건식이 참인 동안 while 문에 속해있는 블록안의 문장들을 반복해서 수행한다. .

❑ while 문 형식

while 문의 형식을 의사코드와 순서도로 표현하면 다음과 같다.

while문 형식	순서도
while 조건식 : 반복 수행할 문장1 … 반복 수행할 문장n 반복 수행할 문장 블록	

while 키워드 옆에 나오는 조건식의 값을 판별하여 참이면 아래에 있는 같은 블록내 문장들을 반복 수행하고, 조건식의 값이 거짓이 되는 순간 반복구조를 빠져나온다.

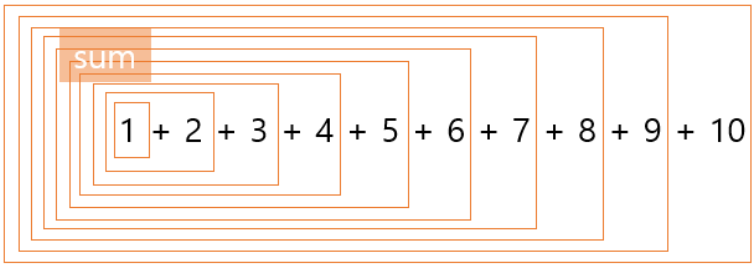
다음은 while 문을 이용하여 “무궁화 꽃이 피었습니다.”를 10 번 출력하는 예시이다.

프로그램	실행 결과

i = 0
while i < 10 :
  print("무궁화 꽃이 피었습니다.")
  i = i + 1
	무궁화 꽃이 피었습니다.
	무궁화 꽃이 피었습니다.
	무궁화 꽃이 피었습니다.
	무궁화 꽃이 피었습니다.
	무궁화 꽃이 피었습니다.
	무궁화 꽃이 피었습니다.
	무궁화 꽃이 피었습니다.
	무궁화 꽃이 피었습니다.
	무궁화 꽃이 피었습니다.
	무궁화 꽃이 피었습니다.

💻 실습 예제 1. 숫자 합 계산하기

1~10 까지의 합을 계산하여 출력하는 문제를 while 문으로 해결해보자.




```
sum = 0
sum = sum + 1
sum = sum + 2
...
sum = sum + 9
sum = sum + 10
```

컴퓨터는 사람과 다르게 위의 연산을 한꺼번에 할 수가 없으므로, 중간 결과를 저장할 변수 sum 을 이용하여 한번에 하나씩 덧셈 연산을 하여 값을 누적시켜 나가야 한다. 위의 그림에서 하나의 박스가 한번의 덧셈 연산에 해당된다. 합을 구하는 절차는 다음과 같다.

① sum 변수의 값을 0 으로 초기화 한다.
② sum 변수에 1 을 더해서 sum 에 저장한다.
③ sum 변수에 2 을 더해서 sum 에 저장한다.
④ sum 변수에 3 을 더해서 sum 에 저장한다.
⑤ sum 변수에 4 을 더해서 sum 에 저장한다.
⑥ sum 변수에 5 을 더해서 sum 에 저장한다.
⑦ sum 변수에 6 을 더해서 sum 에 저장한다.
⑧ sum 변수에 7 을 더해서 sum 에 저장한다.
⑨ sum 변수에 8 을 더해서 sum 에 저장한다.
⑩ sum 변수에 9 을 더해서 sum 에 저장한다.
⑪ sum 변수에 10 을 더해서 sum 에 저장한다.
⑫ sum 의 값을 출력한다.

위 알고리즘에서 ②~⑪ 단계는 공통된 패턴이 존재하므로, 이것을 반복문으로 묶어서 *sum = sum + (1~10 까지의 숫자)* 형태로 표현할 수 있다. *(1~10 까지의 숫자)*는 반복문에서 변수 *i* 를 사용하여 1 부터 시작하여(*i=1*) 10 까지(*i(=10*) 매번 반복할 때 마다 1 씩 증가하도록(*i=i+1*) 수식을 만들어주면 된다. 이때, 변수 *i* 가 반복을 제어하는 역할을 하므로 *i* 를 반복 제어변수라고 부른다.

프로그램	실행 결과

# 반복을 제어할 변수를 선언한다. 	
i = 1
sum = 0
 	
# i 값이 10보다 작으면 반복 	
while i <= 10 :
  sum = sum + i
  i = i + 1
 	
print("합계는", sum)
	합계는 55

💻 응용 문제 1. 51~100 까지의 숫자 합 계산하기

위의 실습 예제를 변형하여 51~100 까지의 숫자 합을 계산하여 출력하는 프로그램을 작성하여 보자.

💻 실습 예제 2. 홀수 합 구하기

1~100 까지의 정수 중 홀수합을 계산하여 출력하는 문제를 while 문으로 해결해보자. 이 문제는 [실습 예제 1]을 응용하여 반복 제어변수 *i*가 홀수일 때만 *sum* 에 더하는 방법으로 해결할 수 있다. 어떤 정수가 홀수인지 짝수인지를 알아내려면 나머지 연산자(%)를 이용하면 된다. 2 로 나눈 나머지가 0 이면 짝수, 1 이면 홀수로 판별할 수 있다. while 문이 100 번만 반복되도록 반복 제어변수 *i*를 이용하여 조건식을 만들어주고, 반복문 내에서 *sum* 에 *i* 값을 더할 때, 홀수인지를 판별하여 홀수인 경우만 *sum* 에 *i* 값을 더하도록 프로그램을 작성하면 된다. 짝수의 합을 구하려면, if 문의 조건식을 *i* % 2 == 0 으로 수정해주면 된다.

프로그램	실행 결과

# 반복을 제어할 변수를 선언한다. 	
i = 1
sum = 0
 	
# i 값이 100보다 작으면 반복 	
while i <= 100 :
  if i % 2 == 1 : # 홀수인지 판별
  sum = sum + i # 홀수의 합에 i 추가
  i = i + 1
 	
print("홀수 합 = ", sum)
	홀수 합 = 2500

💻 실습 예제 3. 짝수, 홀수 합 모두 구하기

이번에는 [실습 예제 2]를 조금 변형하여, 1~100 까지의 정수 중 짝수와 홀수의 값을 따로 구하여 출력하는 프로그램을 작성하여 보자.

프로그램	실행 결과

# 반복을 제어할 변수를 선언한다. 	
i = 1
esum = 0 # 짝수의 합을 저장할 변수를 0으로 초기화
osum = 0 # 홀수의 합을 저장할 변수를 0으로 초기화
 	
# i 값이 100보다 작으면 반복 	
while i <= 100 :
  if i % 2 == 0 : # 짝수인지 판별
  esum = esum + i # 짝수의 합에 i 추가
  else :
  osum = osum + i # 홀수의 합에 i 추가
  i = i + 1
 	
print("짝수 합 = ", esum)
print("홀수 합 = ", osum)
	짝수 합 = 2550
	홀수 합 = 2500

while 문에서 짝수와 홀수의 합을 한번에 구해야 하므로, 짝수합, 홀수합을 각각 저장할 변수를 분리해서 사용한다. 위의 예제에서 *esum* 은 짝수합 저장, *osum* 은 홀수합 저장 용도로 사용되었다. 반복문에서 if~else 문을 사용하여 짝수인 경우와 홀수인 경우로 2 가지로 나누어서 각 경우에 해당되는 *i* 값을 *esum* 또는 *osum* 에 누적하고, 반복문 종료후, 각각의 값을 출력한다.

횟수 반복(for문)

STEP 03

파이썬에서는 정해진 횟수만큼 문장들을 반복 수행하는데 사용되는 for 문을 제공한다. for 문은 횟수 반복문이라고도 부르며, 반복을 시작하기 전에 반복의 횟수를 미리 알 수 있는 경우에 주로 사용한다.

❑ for 문 형식

for 문의 형식을 의사코드와 순서도로 표현하면 다음과 같다.

for 문 형식	순서도
for *변수* in *시퀀스* : 반복 수행할 문장1 … 반복 수행할 문장n 반복 수행할 문장 블록	*시퀀스*에 다음 항목이 있는가? 시퀀스에 더 이상 항목이 없으면 시퀀스의 다음 항목 반복 수행할 문장을 실행

for 문은 *시퀀스* 안에 있는 값들을 차례로 *변수*에 대입하면서 블록 안에 있는 문장들을 시퀀스 원소 개수 만큼 반복 실행한다. *시퀀스*에는 문자열, 리스트, 튜플, 딕셔너리 등과 같이 순서를 가지고 연속적으로 나열된 자료형 상수 또는 변수들이 올 수 있다.

다음은 for 문을 이용하여 반복 메시지를 5 번 출력하는 예시이다.

프로그램	실행 결과
`for i in range(5) :` `  print("대~한민국 짝짝짝~짝짝")`	대~한민국 짝짝짝~짝짝 대~한민국 짝짝짝~짝짝 대~한민국 짝짝짝~짝짝 대~한민국 짝짝짝~짝짝 대~한민국 짝짝짝~짝짝

❑ range 함수를 이용한 반복

for 문은 횟수 반복을 위해 숫자 리스트를 자동으로 생성하는 range() 함수와 함께 많이 사용된다. range() 함수의 사용법은 아래와 같다. start 와 step 값은 생략 시, 파이썬에서 미리 설정되어 있는 디폴트값(default value)으로 설정된다.

range() 함수 사용법
[형식] range(start=0, end, step=1) **[설명]** start : 시작값, 생략 가능(디폴트값은 0) end : 종료값, end 값은 포함되지 않음 step : 한번에 증가되는 값, 생략 가능(디폴트값은 1) **[예제]** for i in range(10) : *# 0, 1, 2, 3, 4, 5, 6, 7, 8, 9* ... for i in range(1, 10) : *# 1, 2, 3, 4, 5, 6, 7, 8, 9* ... for i in range(1, 10, 2) : *# 1, 3, 5, 7, 9*

💻 실습 예제 4. 입력된 정수까지의 합 구하기

앞에서 살펴본 정수합 구하기 예제를 for 문을 이용하여 합을 구할 마지막 값을 입력받는 방식으로 프로그램을 변경해보자.

프로그램	실행 결과

`# 반복을 이용한 정수합 구하기`
`sum = 0`
 	
`limit = int(input("어디까지 계산할까요: "))`
`for i in range(1, limit+1) :`
`  sum += i`
`print("1부터 ", limit, "까지의 정수의 합= ", sum)`
	어디까지 계산할까요: 100
	1부터 100 까지의 정수의 합= 5050

합을 구할 마지막값을 정수로 입력받아 *limit* 변수에 저장한 후, range(1,*limit*+1) 함수를 이용해서 [1, 2, 3, ..., *limit-1*, *limit*] 까지의 숫자 리스트를 자동으로 생성한다. range() 함수에서 마지막값은 숫자 리스트에 포함되지 않으므로 원하는 숫자보다 +1 큰 값으로 지정하는 것을 잊지 말아야 한다. for 문에서 한번 반복할 때마다 생성된 숫자 리스트 원소가 순서대로 하나씩 변수 *i* 에 대입되어, *i* 값을 sum 에 누적하는 방식으로 범위 내의 정수 합을 구하게 된다.

💻 실습 예제 5. 입력된 정수까지의 팩토리얼 구하기

이번에는 정수합을 구하는 대신에 정수 팩토리얼(Factorial) 값을 구해보도록 하자. 팩토리얼은 다음과 같은 연산을 말한다 :

n! = 1 X 2 X 3 X .... X n-1 X n , n >= 1 인 정수

팩토리얼을 구하기 위해서는 1~n 까지의 정수합 구하기에서 sum 에 1~n 까지의 값을 누적하여 더하는 대신에, 값을 곱해주면 된다. 주의할 것은 정수합의 경우에는 sum=0 으로 초기화하면 되지만, 팩토리얼의 경우에는 fact=1 과 같이 1 로 초기화하는 것이 중요하다.

아래에 for 문과 range() 함수를 이용한 횟수 반복의 편리함을 비교하기 위해, 동일한 문제를 for 문과 while 문으로 구현한 코드를 함께 보여주고 있다. 10! 을 구하기 위해 10 을 입력하는 경우, n 에 10 이 저장된다. 왼쪽의 for 문 구현 예제에서 range()문에 n 값이 사용되어 range(1,11) 이 되고, [1,2,3,4,5,6,7,8,9,10] 숫자 리스트가 생성된다. for 문에서 반복이 진행되면서, *i* 에 리스트 내의 1~10 의 값이 순차적으로 대입되어 fact 변수에 값이 누적해서 곱해진다. 오른쪽의 while 문으로 구현된 코드와 비교해보면, for 문이 while 문에 비해 코드의 길이도 짧고 횟수 반복 구조가 한눈에 더 잘 들어오는 것을 확인할 수 있을 것이다.

for 문 구현 예제	while 문 구현 예제

fact = 1
n =int(input("정수를 입력하시오: "))
for i in range(1, n+1) :
  fact = fact * i
 	
print(n, "!은", fact, "이다.")
	fact = 1
	n =int(input("정수를 입력하시오: "))
	i = 1
	while i <= n :
	  fact = fact * i
	  i =i + 1
	
	print(n, "!은", fact, "이다.")

정수를 입력하시오: 10 10 !은 3628800 이다.	

❑ 문자열 시퀀스와 for 문

for 문은 아래 예와 같이 문자열을 구성하는 각 문자에 대해서도 적용할 수 있다. 첫 번째 for 문은 새로운 반복을 수행할때마다 문자열 "abcdef"에 들어있는 문자를 차례로 변수 *c* 에 하나씩 대입하면서 print()문을 실행한다. print()문 두 번째 인자인 end=" "는 c 값을 출력 한 후에

마지막에 출력되는 문자를 공백 문자(“ ”)로 지정한다는 의미이다. end 인자를 지정하지 않으면 디폴트로 줄바꿈 문자('\n')가 출력 내용 끝에 연결되어 나온다.

프로그램	실행 결과

for c in "abcdef" :
  print(c, end = " ")
 	
print()
 	
for c in "가나다라마바사" :
  print(c, end = " ")
	a b c d e f
	가 나 다 라 마 바 사

두 번째 for 문은 첫 번째와 유사하게 문자열 “가나다라마바사”에 들어있는 문자를 차례로 변수 *c* 에 하나씩 대입하면서 print()문을 실행한다.

💻 실습 예제 6. 문자열에서 특정 문자 개수 세기

영문 문자열을 입력받아 특정 문자가 몇 번이나 나오는지를 카운트하는 프로그램을 작성하여 보자. 문제 해결을 위한 절차는 다음과 같다.

① 영문 문자열을 입력받아 str 변수에 저장한다.
② 찾고자 하는 문자를 입력받아 ch 변수에 저장한다.
③ 문자의 개수를 관리하는 변수 count 를 0 으로 초기화 한다.
④ for 문으로 str 에 있는 문자 하나씩을 c 에 대입하면서
  ㉠ c 가 ch 와 같으면 count 를 하나 증가시킨다.
⑤ count 를 출력한다.

프로그램	실행 결과

str = input("영문 문자열 입력 > ")
ch = input("찾고 싶은 문자 입력 > ")
count = 0
for c in str :
  if c == ch :
  count += 1
 	
print(str, "에는 ", count, "개의 ", ch, "가 들어있습니다.")
	영문 문자열 입력 > This is an apple
	찾고 싶은 문자 입력 > a
	This is an apple 에는 2 개의 a 가 들어있습니다

중첩 반복문

STEP 04

중첩 if 문과 마찬가지로 반복문도 하나의 문장이므로 중첩되어 사용될 수 있다. 반복문 안에 또 다른 반복문이 오는 형태로 for 문 안에 for 문이 오거나 while 문이 올 수 있고, while 문안에도 for 문이나 while 문이 올 수 있다.

❑ 중첩 반복문 형태

다음은 중첩 반복문의 다양한 형태이다. 가장 많이 사용되는 형태는 중첩 for 문으로 for 문안에 for 문이 중첩된 구조이다. 이론상으로는 for 문을 무한정 중첩할 수 있으나, 보통 삼중 for 문 이하로 사용한다.

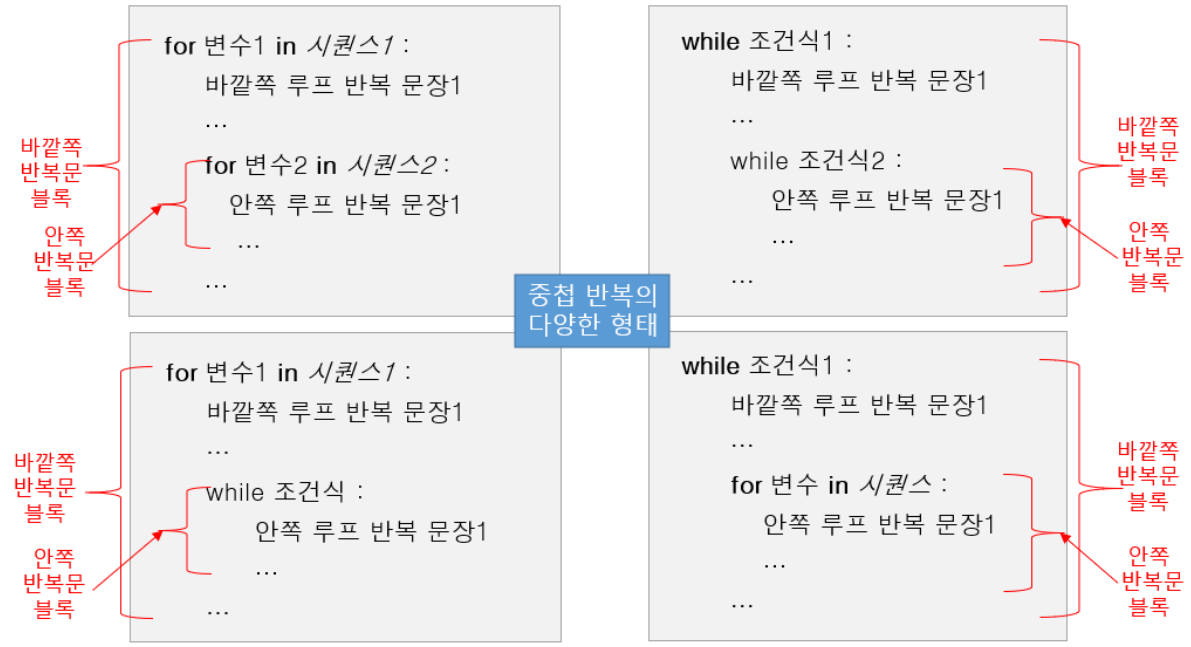
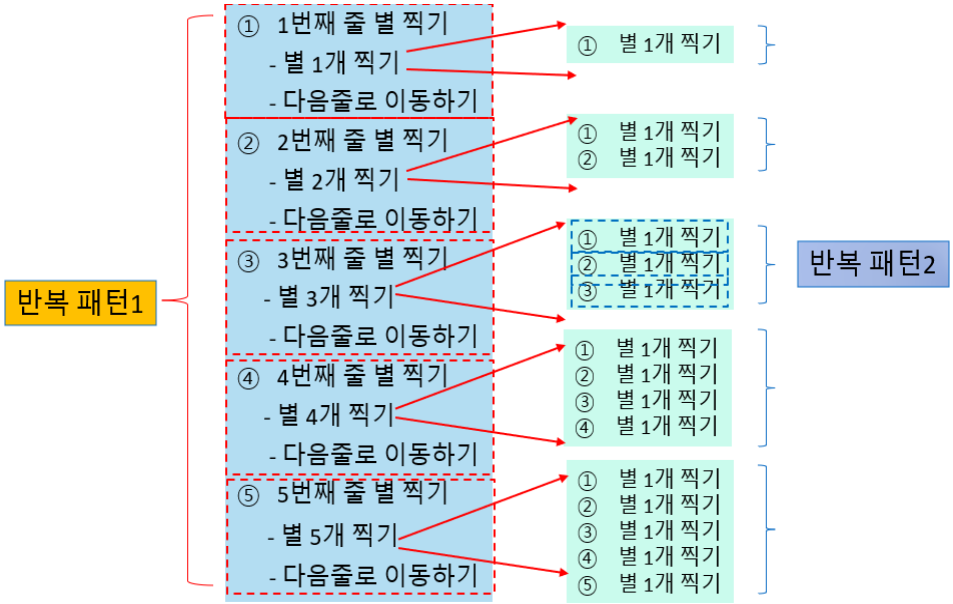


그림 26 중첩 반복문의 다양한 형태

중첩 반복문 사용 시, 문장들의 들여쓰기가 반복문의 범위(블록)을 표시하는 역할을 하므로, 들여쓰기에 주의하여 사용하여야 한다.

❑ 중첩 for 문

중첩 for 문을 이용하여 아래 왼쪽 그림과 같은 직각 삼각형 모양의 별 패턴을 출력해보자. 오른쪽 부분은 직각 삼각형 모양의 별 모양을 찍기 위한 절차를 정리한 것이다. 별 모양을 찍는 과정을 보면 먼저 매 줄마다 별을 1 개씩 늘려가면서 찍는 것을 5 번 반복하는 패턴(반복 패턴 1)이 발견된다. 각 줄의 별을 찍기 위해서는 다시 별 개수만큼 반복하여 별을 출력(반복 패턴 2)해야 한다.

<table>
<tr><th>별 모양</th><th>별 패턴 찍는 과정</th></tr>
<tr><td>*
* *
* * *
* * * *
* * * * *</td><td> 
</td></tr>
</table>

위의 절차를 반복구조로 정리하여 알고리즘을 작성하고, 아래와 같이 중첩 for 문을 이용하여 구현해볼 수 있다.

<table>
<tr><th>알고리즘</th><th>프로그램</th><th>실행 결과</th></tr>
<tr><td>줄번호 = 1
줄번호가 5이하인 동안
  열번호 = 1
  열번호가 줄번호 이하인 동안
  별 1개 출력
  다음줄로 이동하기</td><td>for i in range(1, 6) :
  for j in range(1, i+1):
  print(“*”, end=“”)
  print()</td><td>*
**
***
****
*****</td></tr>
</table>

💻 응용 문제 2. 숫자 패턴 출력하기

위의 실습 예제를 변형하여 다음과 같은 숫자 패턴을 출력하는 프로그램을 작성하여 보자.

(Hint) print 문 출력 부분에 별 대신에 해당 숫자를 저장하는 변수값을 출력한다.

패턴	프로그램
1 1 2 1 2 3 1 2 3 4 1 2 3 4 5	

💻 실습 예제 7. 구구단 만들기

중첩 for 문을 이용하여 오른쪽 그림과 같은 형식으로 2 단에서 9 단까지의 구구단을 출력해보자. 각 단 사이를 구분하기 위해 단이 끝날 때마다 "----------"를 출력한다. 프로그램 구현 전에 구구단 출력을 위한 알고리즘을 먼저 정리한 후, 중첩 for 문을 이용하여 프로그램을 작성해보자.

```
2 X 1 = 2
2 X 2 = 4
2 X 3 = 6
2 X 4 = 8
2 X 5 = 10
2 X 6 = 12
2 X 7 = 14
2 X 8 = 16
2 X 9 = 18
---------
중간 생략
---------
9 X 1 = 9
9 X 2 = 18
9 X 3 = 27
9 X 4 = 36
9 X 5 = 45
9 X 6 = 54
9 X 7 = 63
9 X 8 = 72
9 X 9 = 81
---------
```

알고리즘	프로그램

① 2~9 단에 대하여 다음을 반복 수행, 반복 시, 변수 *x*에 2~9 숫자를 차례로 저장
(ㄱ) *x* 단에 대하여, 다음을 반복 수행, 반복 시, 1~9 사이 숫자를 *y*에 저장
(ㄴ) *x* X *y* = *x* * *y* 형식으로 출력
② 구분선 출력
	for x in range(2, 10) :
	  for y in range(1, 10) :
	  print(x, "X", y, "=", x * y)
	  print("---------");

바깥쪽 for 문에서 range 함수는 마지막 숫자를 포함하지 않으므로, 2 단~9 단까지 반복을 수행하기 위해 range(2, 10)로 작성하였고, 안쪽의 for 문 역시 1~9 까지의 곱하는 숫자 범위를 포함하기 위해 range(1, 10)로 작성하였다. 중첩 for 문을 작성할 때 주의할 점은 각 for 문에서의 변수를 다르게 써야 한다는 것이다. 같은 변수를 사용할 경우, 값 대입에 문제가 생길 수 있으니 주의해야 한다. 예시된 것과 같이 출력 포맷팅을 하기 위해 print 문에서 ','를 이용하여 출력할 내용들을 연결하였다.

break와 continue

STEP 05

반복문 내에서 break 문을 사용하여 특정 조건이 만족되면 반복문을 빠져나오거나, continue 문을 사용하여 반복문 블록의 나머지 문장을 건너뛰소, 바로 다음 반복을 진행하도록 할 수 있다. break 문은 무한 루프문과 함께 사용되는 경우가 많다. 반복문의 실행 흐름을 변경시키는 방법에 대해 알아보자.

❑ 무한 루프(Loop)

파이썬에서는 while 문을 이용하여 반복이 무한정 계속되는 무한 루프를 구현할 수 있다. 다른 언어들과는 다르게 파이썬 for 문은 시퀀스에 있는 원소 개수만큼만 반복을 실행하므로 무한 루프를 만들기 어렵다. 무한 루프는 필요에 따라 명시적으로 만들 수도 있고, 조건식을 잘못 작성하여 우연히 무한 루프가 만들어 질 수도 있다. 이 경우는 디버깅을 통해 오류를 찾아 수정해야만 한다.

다음은 화면에 "Hello!"를 무한정 출력하는 무한 루프의 예이다. 첫 번째는 while 문 다음의 조건식을 True 로 만들어서 언제나 참이 되어 반복문이 무한정 수행되도록 명시적으로 무한 루프를 구현한 예이다. 두 번째는 루프 제어변수 i 를 증가시키는 코드가 누락되어서 오류로 무한 루프가 잘못 만들어진 예이다. while 문을 사용할 때는 이와같이 조건식을 잘못 써서 무한 루프가 발생하지 않도록 주의해야 한다.

프로그램	실행 결과
# 명시적인 무한 루프의 예 **while True :** **print**("Hello!")	Hello! Hello! Hello! Hello! Hello! Hello! ...
i = 0 # 변수 i를 증가시키는 부분이 없어서 무한 루프가 된다. **while i < 10 :** **print**("Hello!")	

❑ break 문

반복을 중간에 멈추는 역할을 하는 break 문은 break 문을 둘러싸고 있는 가장 가까운 반복문을 빠져나가게 된다. 특히 무한 루프를 빠져나가는데, break 문이 많이 사용된다.

무한 루프 + break문 사용법

[형식] while True :
  if *조건식* :
  break
[설명] 반복을 계속 실행하다가 주어진 *조건식*이 참이면 while문을 빠져나간다.

신호등 색깔이 초록색이 될 때까지 계속 기다리다가 초록색이 되면 반복을 멈추는 기능을 다음 예제와 같이 while 문 또는 무한 루프 + break 문으로 구현할 수 있다.

프로그램	실행 결과

light = input("신호등 색상을 입력하시오:")
while light != "green":
  print("기다리세요")
  light = input('신호등 색상을 입력하시오:')
 	
print("직진하세요")
	신호등 색상을 입력하시오:red
	기다리세요
	신호등 색상을 입력하시오:red
	기다리세요
	신호등 색상을 입력하시오:green
	직진하세요

while True :
  light = input('신호등 색상을 입력하시오:')
  if light =="green" :
  break
  print("기다리세요")
 	
print("직진하세요")	

두가지 모두 동일한 기능을 하는 프로그램으로 첫 번째 while 문에서는 조건식을 만들기 위해 입력을 먼저 받은 후, 조건을 체크해야 하고, 두 번째 while 문에서는 조건을 별도로 명시할 필요없이 무조건 반복을 실행하고, 중간에 탈출 조건을 체크해서 break 문으로 루프를 빠져나간다.

🖳 실습 예제 8. 반복해서 음료 주문하기

4 장의 [실습 예제 3]에서 다루었던 음료 주문하기 예제를 무한 루프와 break 문을 이용하여 반복해서 주문이 가능하도록 수정해보자. 0 번을 선택하는 경우 반복을 종료하도록 처리해보자.

프로그램	실행 결과

```
while True :
    print("******* 메뉴 *******")
    print("1. 아메리카노 : 3000원")
    print("2. 카페라떼 : 4000원")
    print("3. 카페모카 : 4000원")
    print("4. 흑당밀크티 : 5000원")
    print("0: (관리자메뉴) 종료")

    menuid = int(input("주문할 음료 번호를 말씀해주세요 > "))
    if menuid == 0 :
        break

    match menuid :
        case 1 :
            print("아메리카노를 주문하셨습니다.")
            print("3000원을 결제합니다");
        case 2 :
            print("카페라떼를 주문하셨습니다.")
            print("4000원을 결제합니다");
        case 3 :
            print("카페모카를 주문하셨습니다.")
            print("4000원을 결제합니다");
        case 4 :
            print("흑당밀크티를 주문하셨습니다.")
            print("5000원을 결제합니다");
        case _ :
            print("잘못된 주문입니다.")
            print("처음부터 다시 시작해주세요.")
```

```
******* 메뉴 *******
1. 아메리카노 : 3000원
2. 카페라떼 : 4000원
3. 카페모카 : 4000원
4. 흑당밀크티 : 5000원
0: (관리자메뉴) 종료
주문할 음료 번호를 말씀해주세요 > 1
아메리카노를 주문하셨습니다.
3000원을 결제합니다
******* 메뉴 *******
1. 아메리카노 : 3000원
2. 카페라떼 : 4000원
3. 카페모카 : 4000원
4. 흑당밀크티 : 5000원
0: (관리자메뉴) 종료
주문할 음료 번호를 말씀해주세요 > 5
잘못된 주문입니다.
처음부터 다시 시작해주세요.
******* 메뉴 *******
1. 아메리카노 : 3000원
2. 카페라떼 : 4000원
3. 카페모카 : 4000원
4. 흑당밀크티 : 5000원
0: (관리자메뉴) 종료
주문할 음료 번호를 말씀해주세요 > 0
```

반복해서 메뉴 주문이 가능하도록 무한 루프로 전체 주문 처리 부분을 감싸고, 관리자 메뉴인 종료에 해당되는 0 번이 입력되면 break 문을 이용하여 while 루프를 빠져나가도록 프로그램을 구현하였다.

❑ continue 문

반복문에서 continue 문을 만나면 이후에 나오는 반복 문장들을 건너뛰고 바로 다음 반복을 진행하게 된다. 다음 예제를 통해 continue 문의 동작을 이해해보자.

프로그램	실행 결과

```
sum3 = 0
for i in range(1, 51):
    if i % 3 == 0 :          # 3의 배수이면 아래 문장 건너뛰기
        continue
    sum3 += i                # 3의 배수가 아닌 숫자 합 구하기

    if i < 50 :
        print(i, end="+")    # 마지막 숫자가 아니면 더하기 기호 붙이기
    else :
        print(i, end="=")    # 마지막 숫자에 = 기호 붙이기

print(sum3)
```

```
1+2+4+5+7+8+10+11+13+14+16+17+19+20+22+23+25+26+28+29+31+32+34+35+37+38+40+41+43+44+46+47+49+50=867
```

위 예제는 1~50 까지의 숫자 중에서 3 의 배수가 아닌 숫자들의 합을 구하는 덧셈식을 표현하고, 마지막에 누적 합을 출력하는 프로그램이다. for 문 안에 첫 번째 if 문은 나머지 연산자를 이용하여 3 의 배수인지를 판별하여, 3 의 배수이면 아래의 덧셈 부분을 건너뛰고 바로 다음 숫자에 대한 반복을 진행하도록 continue 문을 사용하였다. 두 번째 if 문은 덧셈식 형태를 맞추기 위하여 숫자를 출력한 후, 마지막 숫자에는 '=' 기호를 붙이고, 나머지는 '+'기호로 연결하도록 print 문의 end 구분자를 다르게 처리하였다.

6장

함수

함수란?

STEP 01

함수(function)란 특정 작업을 수행하는 코드들의 모임으로 자주 사용되는 코드들을 하나로 묶어서 함수로 만들어 두면, 필요할 때 언제든지 호출하여 사용할 수 있다.

❑ 함수(function)란?

함수는 아래 그림과 같이 입력값을 받아서 어떤 일을 하고, 출력 결과를 반환값으로 내보내는 내부가 감추어진(black-box) 코드들의 모임이다. 함수에는 고유한 이름(함수명)이 있으며, 함수명은 통상적으로 기능을 연상할 수 있는 이름으로 명명한다. 함수는 일종의 소프트웨어 부품이라고 할 수 있으며, 함수의 이름과 기능, 사용법(입력값, 반환값)만 알면 함수의 내부 코드를 알 필요가 없이, 해당 기능이 필요할 때마다 함수를 호출하여 사용할 수 있다.

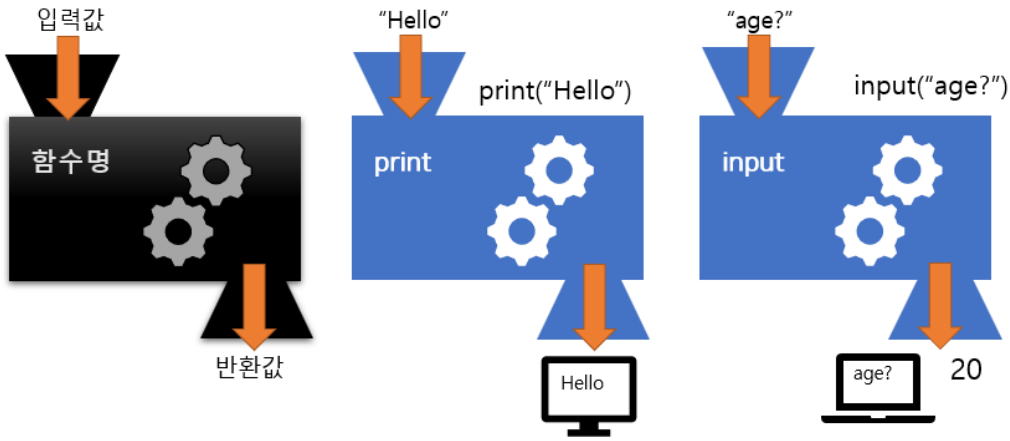


그림 27 함수의 개념

print("Hello")는 print() 함수 호출의 예시로 "Hello"를 입력값으로 주었을 때, 화면에 "Hello"가 출력된다. input("age?")는 input() 함수에 "age?"를 입력값으로 주었을 때, 화면에 age? 프롬프트가 출력되고 키보드로 입력된 값이 함수 호출의 결과값으로 반환된다.

❑ 함수 동작 원리

함수는 아래 그림과 같이 함수에서 주어진 기능을 수행하는 코드가 들어있는 **함수 정의** 부분과 함수를 사용하는 **함수 호출** 부분으로 구성된다.

함수 정의는 함수명 다음의 ( ) 안에 나타나는 매개변수에 입력값을 받아서 함수 내부의 코드들을 실행한 다음, 결과값을 return 문을 통해 밖으로 내보낸다. 왼쪽의 함수 호출 부분에서 함수명인 sum 에 입력값을 넣어서 sum 함수를 호출하면, 프로그램 실행 흐름이 함수 정의 부분으로 넘어가게 된다. 그런 다음, 함수 정의부에 있는 코드들을 실행하다가 return 문을 만나면 원래 함수 호출문으로 돌아와서, 함수가 반환한 결과값을 받아서 처리하고 다음 줄의 코드가 실행되게 된다. 함수 정의 및 호출에 대한 자세한 사항은 **STEP02 사용자 정의 함수 만들기** 를 참고하기 바란다.

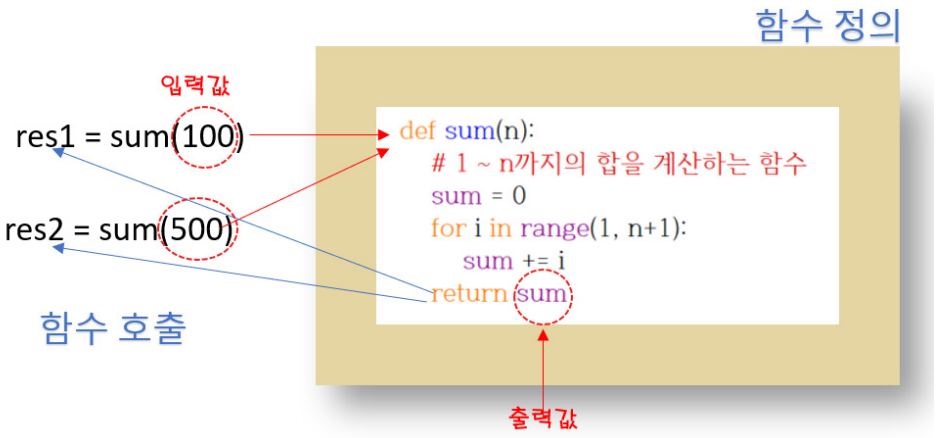


그림 28 함수 정의와 호출

❑ 파이썬 함수의 종류

파이썬은 사용자가 호출하여 편리하게 사용할 수 있는 많은 함수들을 제공한다. 파이썬에서 제공되는 함수는 크게 내장함수(built-in functions)와 표준 라이브러리 함수들(standard library functions)로 구분된다. 이들은 이미 함수의 코드가 정의되어 파이썬과 함께 설치되어 있으므로 필요할 때마다 호출하여 바로 사용할 수 있다. 반면에 사용자가 직접 필요한 함수를 만들어서 사용할 수도 있는데, 이를 사용자 정의 함수(user-defined functions)라고 한다.

파이썬을 설치하면, 모든 파이썬 프로그램에서 import 절차없이 바로 호출하여 사용할 수 있는 print(), input(), int(), range() 등의 기본적인 내장함수가 제공된다. 내장함수보다 더 방대한 기능을 제공하는 함수들을 라이브러리(library)를 통해 이용할 수 있다.

라이브러리는 프로그램에서 가져다 쓸 수 있는 다양한 프로그램 요소(함수,클래스 등)을 모아놓은 것으로 파이썬 설치 시 함께 설치되는 파이썬 표준 라이브러리와 파이썬 개발자들이 만들어 제공하는 외부 라이브러리로 구분할 수 있다. 표준 라이브러리는 별도의 설치과정 없이 sys, pickle, os, random, time 등 원하는 함수가 속해있는 모듈을 프로그램에 import 하여 사용할 수 있다.

표준 라이브러리외에도 개발하고자 하는 프로그램의 목적에 따라 바로 설치하여 소스 코드에 import 하여 사용할 수 있는 수십만 개의 외부 라이브러리들이 끊임없이 개발되고 있다. 예를 들어, 머신러닝 프로그램 개발을 위해 Keras, Tensorflow, Numpy, Pandas 등의 라이브러리를 해당 홈페이지에서 다운로드하여 설치하면, 라이브러리에서 제공하는 다양한 클래스, 함수들을 사용할 수 있다. 아래 그림에서 BeautifulSoup 는 웹크롤링을 위한 라이브러리이며, OpenCV 는 다양한 영상처리에 주로 사용되는 라이브러리이다.

그림 29 파이썬 외부 라이브러리 예

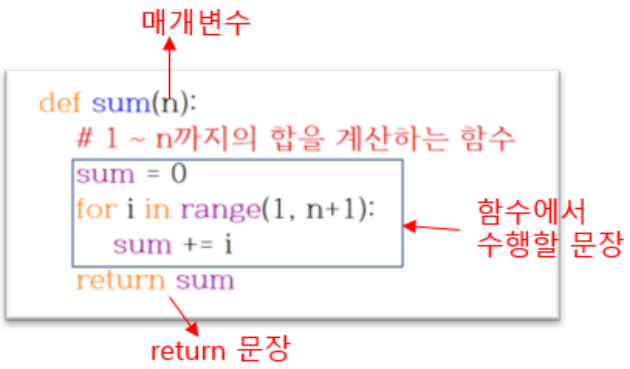
사용자정의 함수 만들기

STEP 02

여러 번 사용되는 처리 단계를 하나로 모아서 함수로 만들어 두면, 필요할 때 언제든지 호출하여 사용할 수 있어 편리할 뿐 아니라, 소스 코드의 중복성을 없애준다. 또한 복잡한 문제를 단순한 부분으로 분해하여 함수 단위로 나누어서 작성함으로써 모듈화가 가능하며, 한번 만들어진 함수는 라이브러리화하여 다른 프로그램에도 사용이 가능하다.

❑ 함수 정의하기

사용자정의 함수를 만드는 형식은 다음과 같다.

함수 정의 형식	함수 정의 예제
def *함수명(매개변수)* : 수행할 문장1 수행할 문장2 … return 문장	

① **def** 는 함수 정의 시작 부분에 나오는 파이썬 예약어로서, 함수 정의를 시작한다는 의미이다.

② *함수명*은 함수 호출에 사용되는 함수의 고유한 이름으로, 함수의 기능을 잘 나타낼 수 있는 이름을 붙이는 것이 좋으며, hello_world, display_menu, send_mail_to_user, get_input, delete_all_users 등과 같이 밑줄로 구분된 단어들을 사용하는 것이 권장된다.

③ *매개변수*는 함수에 입력으로 전달되는 값을 받기 위한 변수 리스트로, 입력값이 없는 경우에는 빈괄호로 남겨둔다.

④ 수행할 문장 1, ..., 수행할 문장 n 은 함수에서 주어진 기능을 수행하는데 필요한 코드들의 집합이다. 매개변수가 있는 경우, 매개변수값을 이용하여 연산을 수행하여 결과값을 만들어낸다.

⑤ return 문은 함수 실행을 끝내고 함수를 호출한 곳으로 돌아가는 문장이다. return 문 옆에 결과값을 함께 명시할 경우에는 결과값이 함수를 호출한 곳으로 반환된다. 결과값을 반환할 필요가 없는 경우(예 : 화면 출력만 하는 경우)에는 return 문 자체를 생략할 수 있다.

❑ 정의된 함수 사용하기

함수가 정의되었으면, 함수 호출을 통해 함수를 사용할 수 있다. 단, 함수 정의는 함수 호출문 전에 먼저 나와야 한다. 라이브러리 함수를 호출하는 경우에도 import 문을 통해 함수를 소스 프로그램으로 먼저 불러오는 과정이 필요하다.

아래 그림은 add() 함수의 정의 부분과 함수를 호출하였을 때, 값이 함수로 전달되고 결과값이 반환되는 과정을 보여주는 그림이다.

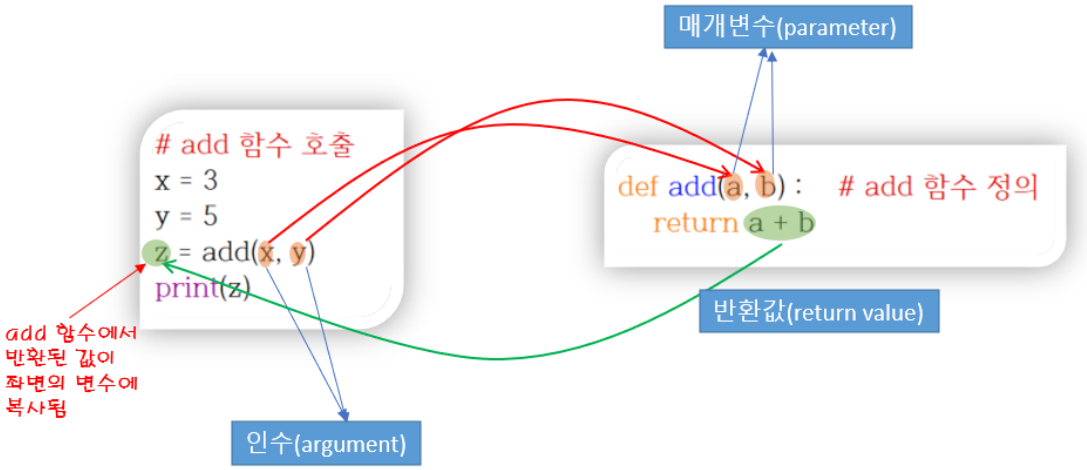


그림 30 함수 호출 시 값의 전달 및 반환

① 오른쪽 add() 함수 정의부에서는 두 개의 입력값 a, b 를 매개변수로 정의하였다. 함수에서는 매개변수 a, b 의 값에 대한 덧셈을 수행해서 나온 결과값을 return 문을 통해 반환하고 있다.

② 왼쪽의 add() 함수 호출부에서는 변수 x, y 에 3 과 5 를 각각 대입한 후, 합을 구하기 위해 x, y 값을 입력값으로 하여 함수 add()를 호출하였다. 함수 호출 시 매개변수에 전달되는 값을 함수의 인수(argument) 또는 인자라고 부른다. 함수 호출을 통해 반환된 값이 변수 z 에 저장되고, print 문을 통해 z 값을 화면에 출력한다.

❑ 함수 정의와 호출 순서

파이썬은 인터프리트 언어(Interprete language)이기 때문에 함수 정의와 호출 순서가 중요하다. 함수는 정의가 먼저 나온 다음에 호출문이 나중에 나오는 것이 일반적이다. 위 add() 예제 그림에서는 설명의 편의를 위해 add() 함수의 정의부가 오른쪽에 있지만, add() 함수 정의가 add() 함수 호출 코드보다 먼저 나와야 한다.

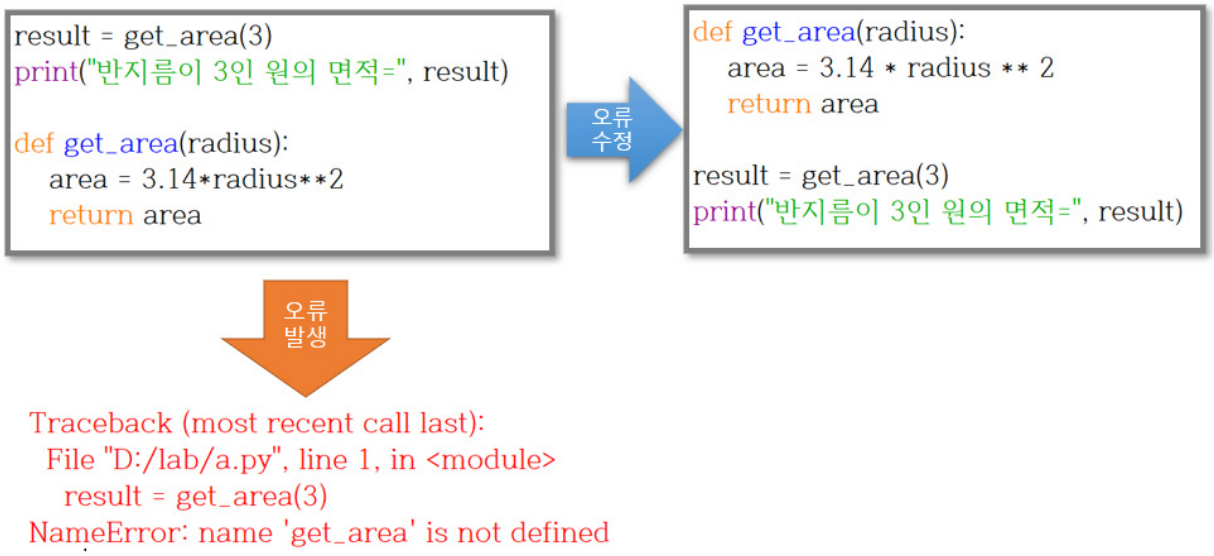


그림 31 함수 정의와 호출 순서가 잘못된 예

위 예제의 왼쪽 코드는 get_area() 함수 호출문이 함수 정의부보다 먼저 나와서 코드를 실행시켰을 때, 번역 과정에서 첫 번째 줄 함수 호출문 처리 시, get_area() 함수가 정의되어 있지 않아 오류가 발생한다. 이 오류를 수정하기 위해서는 오른쪽과 같이 함수 정의부를 함수 호출문 앞으로 위치를 옮기면 해결된다.

❑ 여러 개의 값을 반환하는 함수

파이썬 함수는 return 문을 통해 한 개 이상의 값을 반환할 수 있다. 하나의 값을 반환하는 예는 이미 앞에서 살펴보았고, 이번에는 여러 개의 값을 반환하는 예를 살펴보기로 하자. 아래 그림은 두 개의 값을 매개변수로 받아서 대소를 비교하여 최소값과 최대값을 차례로 반환하는 min_max() 함수 정의와 함수 호출 예를 보여주고 있다.

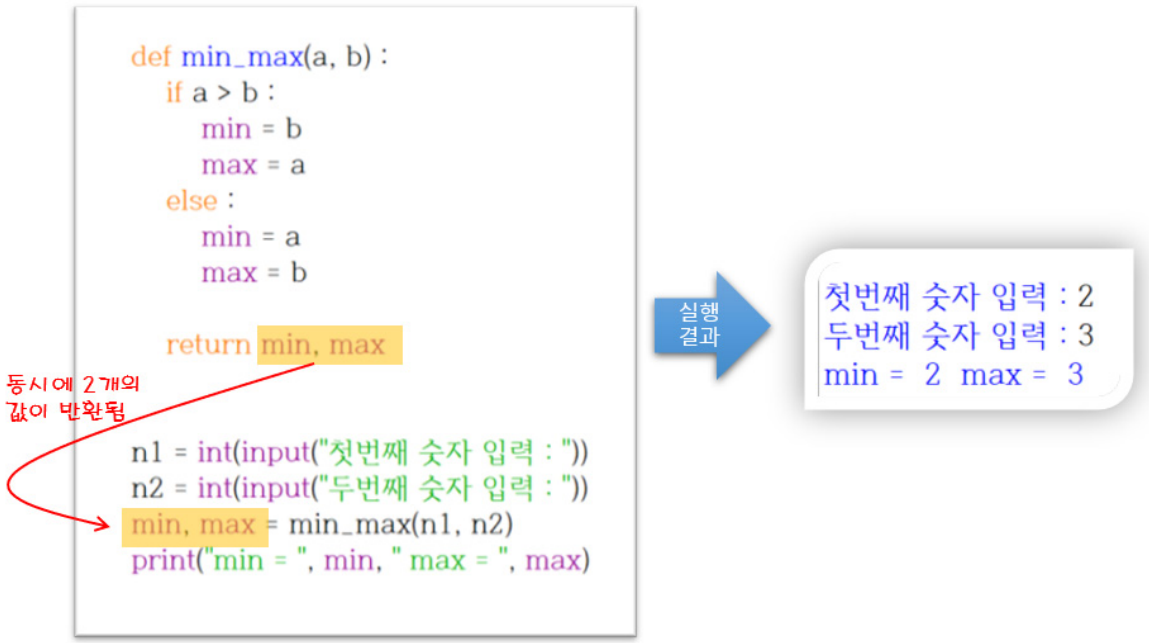


그림 32 2개의 값을 반환하는 함수

① 가장 먼저 나오는 min_max() 함수 정의부에서는 매개변수 a, b 중에 큰 수와 작은 수를 찾아서 각각 min, max 변수에 저장한다.

② return 문을 통해 min, max 값을 동시에 반환한다. 동시 반환은 내부적으로는 나중에 배우게 될 파이썬 시퀀스 타입 중의 하나인 튜플로 묶여서 처리된다.

③ 메인 코드부분에서는 2 개의 정수를 입력받아 n1, n2 에 저장한 후, min_max() 함수에 차례대로 전달하고, 최소값과 최대값을 동시에 반환받아서 왼쪽의 min, max 변수에 차례로 저장한 후, 결과를 출력한다. min_max() 함수 내에서의 변수 min, max 와 함수 바깥쪽에서의 변수 min, max 는 변수가 속한 블록 범위가 다르므로 서로 다른 변수로 취급된다.

❑ 반환값이 없는 함수

파이썬 함수는 다음과 같이 값 반환이 필요없는 경우에는 return 문을 아예 생략할 수 있다. 아래 예제는 햄버거 매장에서 고객의 주문을 받는 키오스크(Kiosk)를 간단하게 흉내 낸 프로그램이다.

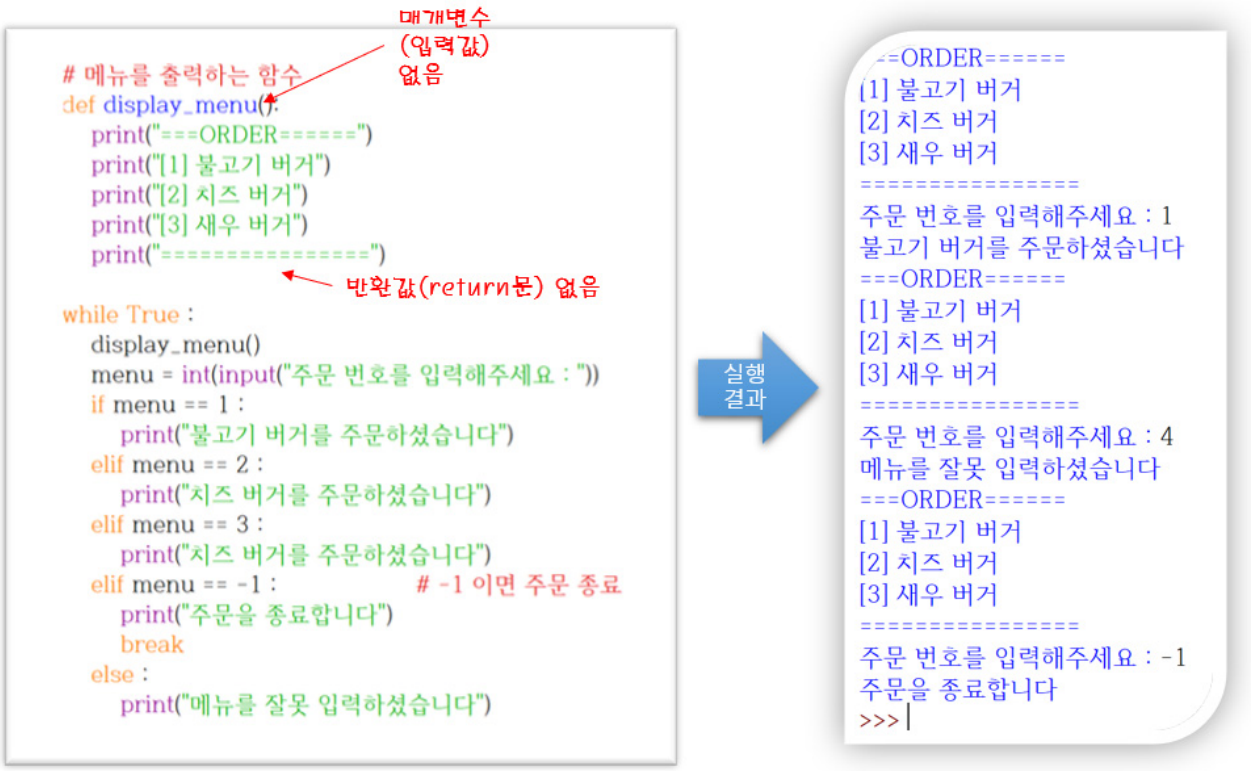


그림 33 반환값이 없는 함수 예

① display_menu() 함수는 주문 가능한 메뉴를 출력하는 함수로 화면에 메뉴만 출력하고 별도의 반환값을 생성하지 않으므로 return 문이 필요없다.

② 메인 프로그램에서 무한 반복을 하면서, display_menu() 함수를 호출하여 메뉴를 화면에 출력한다. input() 함수를 호출하여 주문 번호를 입력받아 menu 변수에 저장한 후, menu 값에 따라 해당 메뉴에 따른 메시지를 출력한다. menu 가 -1 인 경우에는 break 문에 의해 반복을 종료한다.

❑ 매개변수에 디폴트 값이 있는 함수

파이썬 함수는 함수 정의 시, 매개변수에 디폴트값을 지정할 수 있다. 매개변수에 디폴트값을 지정하면, 함수를 호출할 때 인수가 매칭되지 않으면, 해당 매개변수의 값으로 디폴트값이 사용된다.

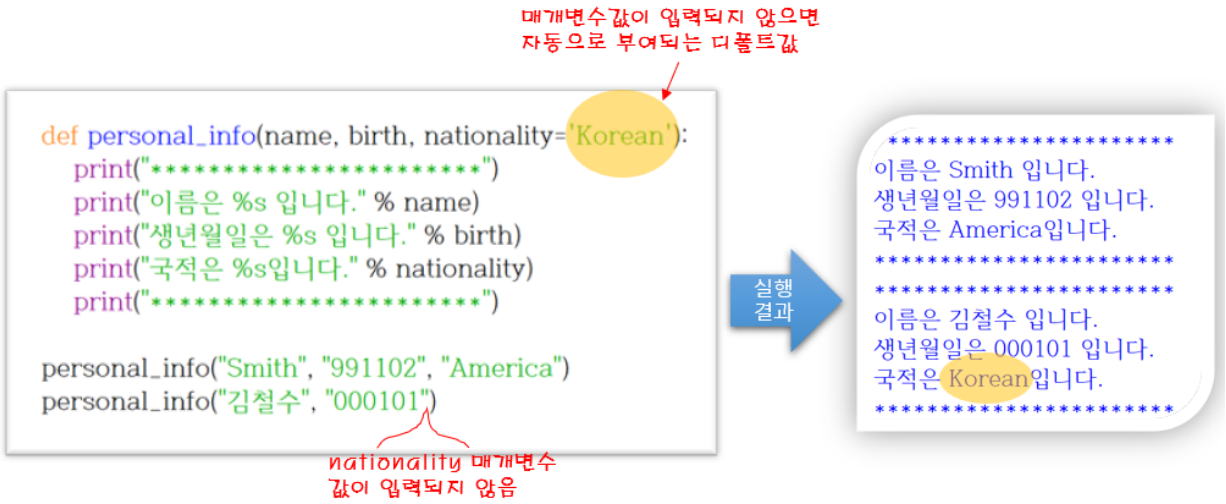


그림 34 매개변수의 디폴트값 사용 예

위 예에서 personal_info() 함수는 name, birth, nationality 의 3 개의 매개변수를 가지고 있고 마지막 nationality 매개변수에는 'Korean' 문자열이 디폴트값으로 지정되어 있다. 메인 프로그램 에 있는 첫번째 personal_info() 함수 호출문에서는 3 개의 인수를 모두 채워서 함수로 전달하고 있는 반면, 두 번째 personal_info() 함수에서는 2 개의 인수만 전달하고 있다. 이 경우, 마지막 1 개의 인수가 누락되어 있으므로 nationality 매개변수에 미리 지정되어 있는 디폴트값인 'Korean'이 저장되어 처리된다.

STEP 03

내장 함수

파이썬에서는 활용도가 높은 기능들을 내장함수(Built-in function)로 제공하여, 프로그램에서 별도의 import 문장 없이 바로 함수 호출이 가능하도록 하고 있다. 이 절에서는 파이썬에서 제공하는 주요 내장함수들에 대해 알아본다.

❑ 파이썬 내장함수 종류

파이썬 언어에서 제공하는 내장함수 리스트는 파이썬 홈페이지 Library Reference - Built-in Functions(https://docs.python.org/3/library/functions.html)에서 확인해 볼 수 있다.

Built-in Functions

A	**E**	**L**	**R**
abs()	enumerate()	len()	range()
aiter()	eval()	list()	repr()
all()	exec()	locals()	reversed()
anext()			round()
any()	**F**	**M**	
ascii()	filter()	map()	**S**
	float()	max()	set()
B	format()	memoryview()	setattr()
bin()	frozenset()	min()	slice()
bool()			sorted()
breakpoint()	**G**	**N**	staticmethod()
bytearray()	getattr()	next()	str()
bytes()	globals()		sum()
		O	super()
C	**H**	object()	
callable()	hasattr()	oct()	**T**
chr()	hash()	open()	tuple()
classmethod()	help()	ord()	type()
compile()	hex()		
complex()		**P**	**V**
	I	pow()	vars()
D	id()	print()	
delattr()	input()	property()	**Z**
dict()	int()		zip()
dir()	isinstance()		
divmod()	issubclass()		**_**
	iter()		__import__()

그림 35 파이썬 내장함수(3.11 버전 기준)

우리가 지금까지 많이 사용했던 print(), input(), int(), range() 등의 함수들도 모두 파이썬 내장함수에 속한다. 많이 사용되는 기본적인 함수들을 추가로 살펴보도록 하자.

❑ abs() 함수

abs() 함수는 인수로 주어진 숫자의 절대값을 반환하는 함수이다.

사용 예1	사용 예2

>>> x = -10
>>> abs(x)
10
	>>> x = -123.456
	>>> abs(x)
	123.456

❑ round() 함수

인수로 주어진 실수 값을 반올림한 결과를 반환하는 함수이다.

사용 방법	사용 예2

▪ 사용법 : round(*number[, ndigits]*)
▪ 소수점 다음 *ndigits* 자리까지 반올림
▪ *ndigits* : 생략 시, 입력값과 가장 가까운 정수 반환
	>>> x = 2.165
	>>> round(x)
	2
	>>> round(x, 1)
	2.2
	>>> round(x, 2)
	2.17

❑ list() 함수

주어진 인수의 내용을 바탕으로 리스트 객체를 생성하는 함수이다. 리스트는 여러 자료들의 순서있는 모임으로 자료들을 묶는데 [ ] 기호를 사용하고, 각 요소들을 콤마(,)로 구분한다. 리스트에 대해서는 다음 장에서 자세히 배울 것이다.

리스트 예	사용 예

days = ['Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat']
values = [1, 2, 3, 4, 5, 6, 7]
grades = [4.3, 4.1, 3.5, 2.8, 4.0, 3.9, 2.9, 3.5]
a = [ ]
b = [1, 2, [1, 2, 3]]
fruits = ['apple', 'pear', 'banana', 'cherry']
blist = [True, False]
	>>> str = "ABCDEF"
	>>> list(str)
	['A', 'B', 'C', 'D', 'E', 'F']
	>>> list(range(1, 5))
	[1, 2, 3, 4]

❑ len() 함수

len() 함수는 입력된 객체의 길이를 계산하여 반환하는 함수이다. 첫 번째 예는 문자열의 길이를 계산하는 예제이고, 두 번째 예는 리스트의 원소의 개수를 계산하여 반환하는 예이다.

사용 예1	사용 예2

>>> len("안녕하세요~")
6
>>> len("Hello World")
11
	>>> len([1,2,3,4,5])
	5

❑ min(), max() 함수

min() 함수는 인수로 주어진 시퀀스에서 가장 작은 값을 가지는 항목을, max() 함수는 가장 큰 값을 가지는 항목을 반환하는 함수이다.

사용 예1	사용 예2

>>> data = [1, 2, 3, 4, 5]
>>> min(data)
1
>>> max(data)
5
	>>> data = "hello"
	>>> min(data)
	'e'
	>>> max(data)
	'o'

첫 번째 예는 숫자 리스트가 저장된 data 에서 min() 함수로 가장 작은 값인 1 을, max() 함수로 가장 큰 값인 5 를 추출하는 예이다. 두 번째 예는 data 문자열 안의 문자들을 알파벳 순으로 정렬했을 때, min() 함수로 가장 앞에 나오는 문자 'e'를, max() 함수로 가장 나중에 나오는 문자 'o'를 추출하는 예이다.

❑ sum() 함수

sum() 함수는 인수로 전달된 반복 가능한 항목들을 가진 자료들의 합을 반환하는 함수로 아래 예에서와 같이 리스트에 있는 각 항목들에 합을 누적해서 구하는 sum 연산을 반복 수행해서 합을 구해서 반환한다.

사용 예1	사용 예2
>>> sum([1, 2, 3, 4, 5]) 15	>>> sum([-2, -1, 1, 2]) 0

❑ zip() 함수

zip() 함수는 2 개의 자료형을 하나로 묶어주는 함수이다. 아래 예는 zip 함수를 적용하여 숫자 리스트인 nlist 와 문자열 리스트인 slist 를 같은 인덱스를 가지는 요소끼리 튜플로 묶어준 후, 이것을 다시 list() 함수를 통하여 하나의 리스트로 만들어 result 에 저장하는 방법을 보여주고 있다. 튜플은 리스트와 유사한 자료형으로 자료들을 묶는데 ( )를 사용한다. 자료들의 순서있는 모임이라는 점에서는 리스트와 유사하나, 삽입/수정/삭제가 자유로운 리스트와 달리, 삽입/수정/삭제가 불가능하여 한번 생성된 후에는 읽기 전용으로만 사용할 수 있다는 점이 다르다. 리스트와 튜플에 대해서는 7 장에서 자세히 다루게 될 것이다.

사용 예

>>> nlist = [1, 2, 3, 4]
>>> slist = ["하나", "둘", "셋", "넷"]
>>> result = list(zip(nlist, slist))
>>> print(result)
[(1, '하나'), (2, '둘'), (3, '셋'), (4, '넷')]

❑ map() 함수

map() 함수는 리스트, 튜플 등과 같이 반복 가능한 객체의 각 항목에 첫 번째 인수로 주어진 함수를 적용한 후, 결과를 반환하는 함수이다.

사용 예	

>>> list1 = [-3.4, 1.456, 5.99, -45.0]
>>> list2 = list(map(round, list1))
>>> print(list2)
[-3, 1, 6, -45]
	>>> list3 = list(map(abs, list2))
	>>> print(list3)
	[3, 1, 6, 45]

왼쪽 두 번째 줄에서 map() 함수로 list1 의 각 항목에 반올림 round() 함수를 적용한 후, 다시 list() 함수를 적용하여 list2 를 생성한다. list2 출력 결과를 보면, list1 과 다르게 각 항목의 값이 반올림되어 출력된 것을 확인할 수 있다. 오른쪽은 다시 map() 함수를 이용하여 list2 의 각 항목에 abs() 함수를 적용한 후, list 로 만들어 list3 에 저장하고 있다. list3 를 출력한 결과, 음수 값이 양수로 변환되어 출력된 것을 확인할 수 있다.

❑ filter() 함수

filter() 함수는 특정 조건을 만족하는 원소들을 뽑아내는 함수이다.

사용 예

```
>>> def positive(x) :
        return x > 0

>>> result = filter(positive, (-5, 1, 30, -55, 0))
>>> print(list(result))
[1, 30]
```

위 예는 양수인지를 검사하여 True/False 값을 반환하는 positive() 함수를 먼저 정의한 후, (-5, 1, 30, -55, 0) 튜플에 positive() 함수를 반복 적용하여 True 로 반환되는 항목들만 뽑아서 리스트로 만들어 출력하고 있다.

라이브러리 활용하기

STEP 04

이 절에서는 미리 작성되어 있는 모듈(module)들의 집합인 라이브러리(library)를 활용하는 방법에 대하여 알아본다. 사용자는 원하는 기능의 라이브러리를 설치한 후, 모듈을 import 하면 모듈안에 들어있는 함수, 클래스 등을 자유롭게 사용할 수 있다.

❑ 파이썬 라이브러리 종류

파이썬 언어에서 사용할 수 있는 라이브러리는 파이썬과 함께 설치되는 표준 라이브러리(Standard Library)와 따로 설치해서 사용해야 하는 외부 라이브러리로 구분할 수 있다. 파이썬 언어는 sys, pickle, OS, glob, time, calendar, random, webbrowser 등 방대한 표준 라이브러리를 자체적으로 제공한다. 외부 라이브러리는 전 세계의 파이썬 개발자들이 개발해놓은 모듈들의 집합으로 다양한 기능의 오픈소스 라이브러리들을 설치하여 무료로 사용할 수 있다.

그림 36 파이썬 외부 라이브러리 예

많이 사용되는 대표적인 파이썬 오픈소스 라이브러리들의 예로서 데이터분석에 많이 사용되는 NumPy, Pandas, 머신러닝 대표 라이브러리인 scikit-learn, Tensorflow, Keras, 데이터 시각화 라이브러리인 matplotlib, seaborn, 영상처리 라이브러리인 OpenCV, 웹 개발에 사용되는 Requests, 웹 크롤링 라이브러리인 BeautifulSoup 등이 있다.

❑ 파이썬 모듈(module)

파이썬 언어에서 모듈이란 함수나 변수, 클래스 등 프로그램 요소들이 들어있는 소스 파일로 모듈 간에 서로 import 하여 사용할 수 있다. 이미 만들어진 다른 파이썬 모듈에 정의된 함수나 클래스 등을 사용하려면 먼저 해당 모듈을 import 해야만 한다.

❑ 모듈 불러오기 : import

다른 모듈을 불러올 때 사용하는 import 명령어는 다음과 같이 사용할 수 있다.

사용법	사용 예	실행 결과
import *모듈명*	import random dice=random.randint(1, 6) print(dice)	>>> ========== 2 >>> ========== 5

위 예는 주사위 던지기를 시뮬레이션하는 코드로 주사위 눈에 해당하는 1~6 사이의 무작위 수를 발생시키기 위해 파이썬 표준 라이브러리인 random 모듈에서 제공하는 randint() 함수를 호출하여 사용하고 있다. randint() 함수를 사용하려면 먼저 random 모듈을 import 한 후, random.randint() 형태로 사용한다. 모듈명.함수명 형식으로 사용하는 이유는 동일한 이름을 가지는 함수들이 여러 모듈에 존재할 경우, 이들을 구분하기 위해서이다.

모듈에서 제공하는 함수를 편리하게 사용하려면 다음과 같이 다양한 방식으로 import 할 수 있다.

	사용법	사용 예
1	from *모듈명* import *함수명*	from random import randint dice = randint(1, 6) print(dice)
2	from *모듈명* import *함수명1, 함수명2*	from random import seed, randint seed() dice = randint(1, 6) print(dice)
3	from *모듈명* import *	from random import * dice = randint(1, 100) rlist = sample(range(1,101), k=10) print(rlist) print(dice)

함수 호출 시, 모듈 이름을 생략하고 싶으면, 위와 같이 from 절 다음에 모듈명을 쓰고, import 다음에 함수명을 지정해주면 된다. from 절을 이용한 import 시, 여러개의 함수명을 콤마를 이용하여 나열할 수도 있고, 와일드카드 문자인 '*'를 이용하여 해당 모듈의 모든 함수, 클래스 등을 한꺼번에 가져오는 것도 가능하다.

❑ random 모듈

앞의 예제에서 나왔던 random 모듈은 파이썬 표준 라이브러리 모듈 중의 하나로 게임이나 시뮬레이션에 많이 쓰이는 무작위수 발생과 연관된 함수 등 프로그램 구성 요소들이 모여있다. random 모듈에서 많이 쓰이는 함수들을 알아보도록 한다.

함수명	사용 예	예제 설명
random.randint(a, b) ▪ a ≤ N ≤ b 를 만족하는 임의의 정수 N 을 반환한다.	from random import randint dice = randint(1, 6) print(dice)	1~6 사이의 임의의 정수를 반환 받아 출력
random.seed() ▪ 현재 시간을 사용하여 난수 생성기를 초기화 한다.	from random import seed, randint seed() dice = randint(1, 6) print(dice)	seed() 함수로 난수를 초기화하고, 1~6 사이의 임의의 정수를 반환 받아 출력
random.random() ▪ [0.0, 1.0) 구간에서 임의의 부동 소수점 숫자를 반환한다.	import random x = random.random() print(x) ========== 0.6045045589369999	random() 함수를 호출하여 0~1 사이의 실수를 임의로 생성하여 x에 저장 후 출력
random.shuffle(*x[, random]*) ▪ 시퀀스 *x* 를 제자리에서 섞는다.	from random import * deck = ['ace', 'two', 'three', 'four'] shuffle(deck) print(deck) ---------- ['three', 'ace', 'two', 'four']	deck 리스트에 있는 문자열을 섞은 결과를 출력
random.sample(*polulation,k,*, counts=None*) ▪ *polulation* 시퀀스나 집합에서 선택한 고유한 요소의 *k* 길이 리스트를 반환한다.	from random import * slist = sample([10,20,30,40,50,60], k=3) print(slist) ========== [40, 30, 50]	sample() 함수의 인자로 주어진 리스트에서 3개의 원소를 임의로 뽑아서 slist에 저장 후, 출력

❑ math 모듈

math 모듈에는 수학과 관련된 다양한 기능을 제공하는 함수, 상수 등이 정의되어 있다. 아래는 자주 쓰는 math 모듈 함수 및 상수들이다.

상수명 / 함수명	설명
math.pi	원주율 상수
math.e	자연 상수
math.trunc()	버림 계산 함수
math.factorial()	팩토리얼 계산 함수
math.degrees()	라디안을 입력받아 도를 계산하는 함수
math.radians()	도를 입력받아 라디안을 계산하는 함수
math.cos()	입력된 라디안에 대한 코사인 값을 계산하는 함수
math.sin()	입력된 라디안에 대한 사인 값을 계산하는 함수
math.tan()	입력된 라디안에 대한 탄젠트 값을 계산하는 함수
math.acos()	cos()의 역함수 값을 계산하는 함수
math.asin()	sin()의 역함수 값을 계산하는 함수
math.atan()	tan()의 역함수 값을 계산하는 함수
math.pow()	제곱 연산 결과값을 반환하는 함수
math.sqrt()	제곱근 연산 결과값을 반환하는 함수
math.log()	첫 번째 매개변수의 로그를 구하는 함수
math.log10()	밑수가 10인 로그를 계산하는 함수

사용 예

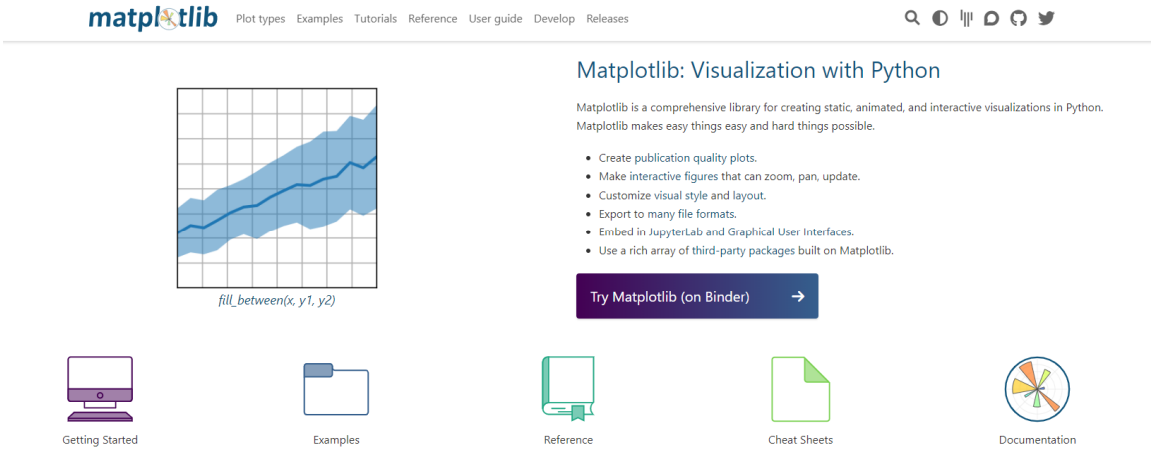
```
import math                 # math 모듈을 임포트
print(math.pi)              # math.pi 상수값을 출력
print(math.factorial(10))   # math.factorial() 함수를 이용하여 10! 값을 출력
print(math.sqrt(100))       # math.sqrt() 함수를 이용하여 100의 제곱근 값을 출력
print(math.pow(2, 5))       # math.pow() 함수를 이용하여 2의 5승 값을 출력
print(math.trunc(1.45))     # math.trunc() 함수를 이용하여 1.45의 버림값을 출력
print(math.gcd(6,8))        # math.gcd() 함수를 이용하여 6과 8의 최대 공약수 값을 출력

==============
3.141592653589793
3628800
10.0
32.0
1
2
```

❑ 외부 라이브러리 활용하기

파이썬에서는 Pandas, NumPy, matplotlib 나 Keras, OpenCV 등의 외부 라이브러리를 이용하여 데이터분석, 머신러닝, 영상처리 등 다양한 기능을 쉽게 구현할 수 있다. 외부 라이브러리를 사용하려면, 먼저 라이브러리 설치 작업이 필요하다.

외부 라이브러리 활용 방법을 데이터 시각화에 많이 사용되는 matplotlib 라이브러리를 활용하는 방법을 통해 알아보기로 하자.

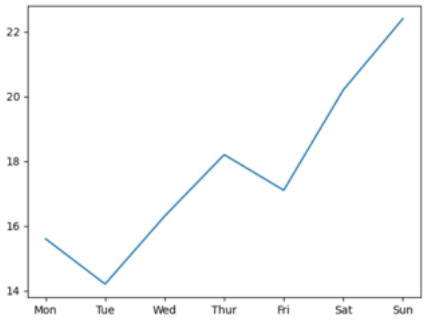
① matplotlib 홈페이지를 방문하여 User guide 페이지에서 라이브러리 설치 방법을 확인한다.



② matplotlib 홈페이지를 방문하여 User guide 페이지에서 라이브러리 설치 방법을 확인한다. 파이썬 IDLE 을 사용하는 경우에는 명령 프롬프트 창에서 아래와 같이 pip3 명령어를 이용하여 라이브러리를 설치하면 된다.

```
명령 프롬프트
C:₩Users₩miseo>pip3 install -U matplotlib
Collecting matplotlib
  Downloading matplotlib-3.4.2-cp39-cp39-win_amd64.whl (7.1 MB)
     |████████████████████████████████| 7.1 MB 2.2 MB/s
Collecting python-dateutil>=2.7
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
     |████████████████████████████████| 247 kB 6.8 MB/s
Collecting kiwisolver>=1.0.1
  Downloading kiwisolver-1.3.1-cp39-cp39-win_amd64.whl (51 kB)
     |████████████████████████████████| 51 kB ...
Collecting cycler>=0.10
  Downloading cycler-0.10.0-py2.py3-none-any.whl (6.5 kB)
Collecting numpy>=1.16
  Downloading numpy-1.21.1-cp39-cp39-win_amd64.whl (14.0 MB)
     |████████████████████████████████| 14.0 MB 6.8 MB/s
Collecting pyparsing>=2.2.1
  Downloading pyparsing-2.4.7-py2.py3-none-any.whl (67 kB)
     |████████████████████████████████| 67 kB ...
Requirement already satisfied: pillow>=6.2.0 in c:₩users₩miseo₩appdata₩local₩programs₩python₩python39₩lib₩site-packages
(from matplotlib) (8.3.1)
Collecting six
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: six, python-dateutil, pyparsing, numpy, kiwisolver, cycler, matplotlib
Successfully installed cycler-0.10.0 kiwisolver-1.3.1 matplotlib-3.4.2 numpy-1.21.1 pyparsing-2.4.7 python-dateutil-2.8.
2 six-1.16.0
WARNING: You are using pip version 21.1.3; however, version 21.2.3 is available
```

③ matplotlib 라이브러리 설치가 끝나면, 다음 예와 같이 파이썬 소스파일에 matplotlib 모듈을 import 하여 필요한 기능을 사용하면 된다.

사용 예	실행 결과
import matplotlib.pyplot as plt X = ["Mon", "Tue", "Wed", "Thur", "Fri", "Sat", "Sun"] Y = [15.6, 14.2, 16.3, 18.2, 17.1, 20.2, 22.4] plt.plot(X, Y) plt.show()	

첫 번째 줄의 matplotlib,pyplot 은 matplotlib 라이브러리의 pyplot 모듈을 import 하고, matplotlib,pyplot 모듈명 대신 plt 라는 별명을 사용하겠다는 의미이다. 요일명을 항목으로 가지고 있는 X 리스트를 정의하고, 요일별 온도 데이터를 Y 리스트에 저장한 후, X, Y 를 인수로 하여 matplotlib,pyplot 모듈에서 제공하는 plot() 함수를 호출하면, 오른쪽 그림과 같은 꺾은선 그래프가 그려진다.

7장

파이썬 자료구조

자료구조란?

STEP 01

자료구조(Data Structure)란 컴퓨터에서 사용되는 많은 자료를 사용 방법이나 성격에 따라 효율적으로 사용하기 위하여 체계적으로 정리하여 저장하는 방법이다. 이절에서는 컴퓨터에서 자료를 정리하는 기본적인 방법들에 대해 알아보자.

❑ 자료구조의 필요성

일상생활에서 보관할 물건이 많은 경우, 용도에 맞는 정리함을 활용하여 정리해두면, 나중에 물건을 찾는 속도가 빨라지고, 여유 공간을 한눈에 파악하여 빠른 정리를 할 수 있다. 이와 유사하게 컴퓨터에서도 자료를 빨리 찾아 쓰려면 자료를 잘 정리해두는 것이 매우 중요하다. 용도에 맞는 적절한 자료구조를 활용하여 정리를 해두면, 기억장치 공간도 효율적으로 잘 쓸 수 있고, 자료 처리 속도도 향상된다.

❑ 자료구조의 종류

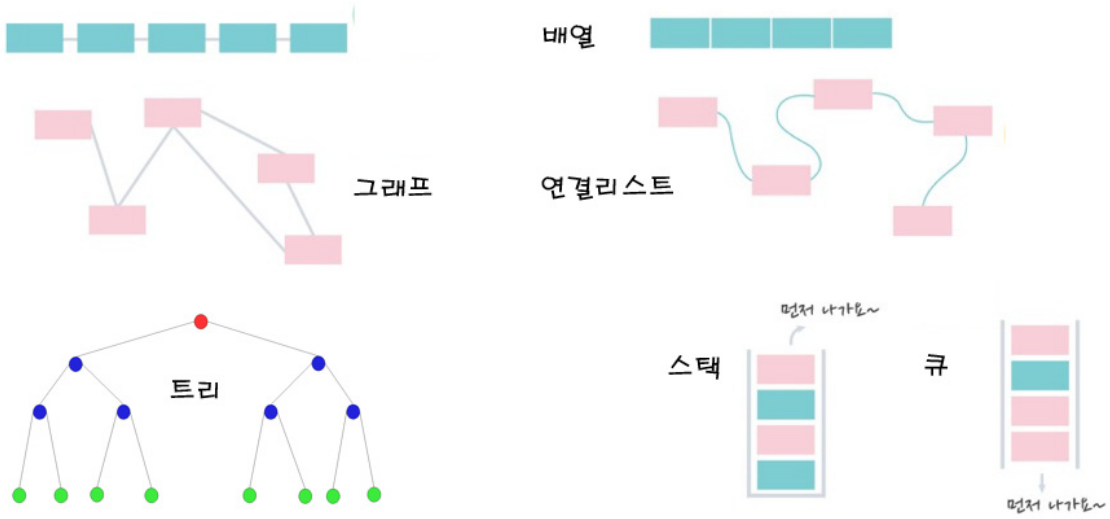


그림 37 자료구조의 종류(출처 : http://www.playsw.or.kr)

컴퓨터 자료구조는 표현하고자 하는 자료간의 관계에 따라, 혹은 기억장치에 구현되는 방법에 따라, 혹은 처리 방법에 따라 종류를 구분할 수 있다. 많이 사용되는 대표적인 자료구조로는 리스트(List), 스택(Stack), 큐(Queue), 트리(Tree), 그래프(Graph) 등이 있다.

- 리스트 : 순서있는 항목들의 모임을 저장하는데 적합한 자료구조로 배열이나 연결리스트 형태로 구현된다.
- 스택 : 아래가 막혀있고 위가 오픈된 선형 자료구조이다. 맨 윗부분에서 자료의 삽입과 삭제가 이루어지고, 마지막에 저장된 원소가 가장 먼저 빠져나오는 LIFO(Last-In-First-Out) 특징을 가진다. 대부분의 프로그램들에서 제공되는 가장 최근에 실행했던 명령을 취소하는 되돌리기(Undo) 기능을 구현하는데 유용하다.
- 큐 : 양 끝단이 뚫려있는 선형 자료구조로 맨 끝단(rear)에서 자료의 삽입이 이루어지고, 맨 앞(front)에서 자료의 삭제가 이루어진다. 맨 먼저 삽입된 원소가 맨 먼저 빠져나오는 FIFO(First-In-First-Out)의 특징을 가진다. 은행이나 식당, 상점 등에서의 서비스 대기열을 구현하는데 적합하다.
- 그래프 : 자료들의 관계를 나타내는데 사용되는 비선형 자료구조로 자료들 간에 연결 사이클(cycle)이 존재할 수 있다. 생태계 먹이사슬 관계, 도로망 등을 표현하는데 적합하다.
- 트리 : 조직도나 윈도우 탐색기처럼 자료들의 계층적인 관계를 표현하는데 사용되는 비선형 자료구조로 사이클이 없는 그래프의 일종이다.

프로그램에서 다루는 데이터 및 응용의 특성에 따라 적합한 자료구조를 선택해서 사용하면 프로그램의 성능을 높일 수 있다. 이 책에서는 여러 자료구조 중에서 가장 기초적이라 할 수 있는 리스트 자료구조를 다루는데 필요한 파이썬 리스트에 대해서 살펴보기로 한다.

리스트

STEP 02

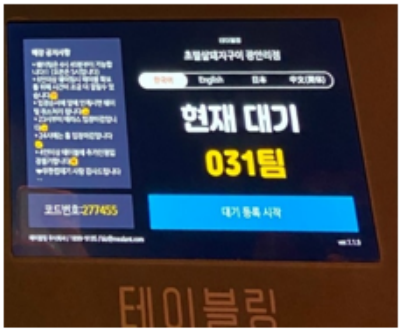
리스트(List)는 순서가 있는 항목들의 모임으로, 숫자, 문자열 등 다양한 자료형으로 리스트를 구성할 수 있다. 이 절에서는 기초적이고 가장 중요한 자료구조인 리스트의 개념과 사용 방법에 대해서 자세히 알아보기로 한다.

❑ 리스트란?

리스트는 관련있는 항목들을 순서대로 저장한 자료구조이다. 일상 생활에서도 리스트를 많이 사용한다. 인기 영화 리스트, 검색어 리스트, 버킷 리스트, 체크 리스트, 장바구니 목록 등이 모두 리스트의 예이다.

- 요일들 : [일요일, 월요일, ..., 토요일]
- 매 월 : [1 월, 2 월, 3 월, ..., 12 월]
- 도시명 : [서울, 부산, 대구, 광주, 인천, 대전]
- 영화리스트 : [오펜하이머, 타겟, 달짝지근해, 콘크리트 유토피아]

아래 그림과 같이 식당에서 대기자들의 전화번호를 저장하는 대기자 리스트도 리스트 자료구조를 이용하여 구현할 수 있다.



010-111-1111	010-111-2222	010-111-3333	010-111-4444	010-111-5555

그림 38 식당 대기 리스트 예

❑ 리스트 인덱싱(indexing)

리스트는 여러 항목들의 순서있는 모임이므로 각 항목마다 고유한 순서 번호가 매겨질 수 있는데 이를 인덱스(index)라고 한다. 리스트의 인덱스는 0 부터 시작하며, 인덱스를 사용하여 리스트의 각 항목에 유일한 번호를 할당하는 것을 리스트 인덱싱(indexing)이라고 한다.

다음은 2023 년 9 월 기준 박스오피스 순위 리스트 예제이다. 영화 '오펜하이머'는 1 번 위치에 있으므로 인덱스는 0 이며, 영화 '여름을 향한 터널, 이별의 출구'는 10 번째에 위차하므로 인덱스는 9 가 된다.

순위	영화 제목	인덱스
1	오펜하이머	0
2	타겟	1
3	달짝지근해 : 7510	2
4	콘크리트 유토피아	3
5	밀수	4
6	엘리멘탈	5
7	스파이 코드명 포춘	6
8	신체모음.zip	7
9	마야 3: 숲속 왕국의 위기	8
10	여름을 향한 터널, 이별의 출구	9

리스트 안의 항목은 보통 대괄호를 사용하여 나타내므로 전체 영화 리스트를 movie_list 라고 하면, 첫 번째 영화는 movie_list[0], 마지막 영화는 movie_list[9]로 표현할 수 있다. 리스트 내용은 삽입, 삭제, 수정이 가능하다. 위의 예에서 박스오피스 순위가 바뀔 때마다 해당하는 영화명을 최신 데이터로 수정할 수 있다.

파이썬 리스트

STEP 03

파이썬 언어에서도 시퀀스(Sequence) 자료형 중 하나로 관련 있는 자료들의 순서있는 모임인 리스트를 제공한다. 이 절에서는 파이썬 리스트를 사용하는 방법에 대하여 자세히 알아보자.

❑ 파이썬 리스트

파이썬 리스트는 같은 자료형 데이터 뿐만 아니라, 서로 다른 자료형의 데이터를 모아서 리스트를 구성할 수 있다.

파이썬 리스트 사용법

[형식] *리스트명* = [*요소1, 요소2, 요소3,…*]
[설명] 리스트 구성 요소들을 [ ] 안에 콤마로 구분하여 순서대로 나열하여 생성
리스트 요소로 모든 자료형이 올 수 있다.
생성된 리스트를 *리스트명* 에 해당하는 변수와 연결하여 사용
[예제] days=[‘월’, ‘화’, ‘수‘, ‘목’, ‘금‘, ‘토‘, ‘일‘] # 요일명을 저장하는 문자열 리스트 days 생성
students=[30,27,31,30,29] # 반별 학생수를 저장하는 숫자 리스트 students 생성

❑ 다양한 형태의 파이썬 리스트

파이썬 리스트는 요소가 하나도 없는 공백 리스트부터, 숫자로만 이루어진 숫자 리스트, 문자열들로만 이루어진 문자열 리스트 등 모든 자료형으로 다양한 리스트를 구성할 수 있다.

1) 공백 리스트(Empty List)

- 요소를 하나도 가지고 있지 않은 비어있는 리스트이다.
- 공백 리스트를 생성한 후, 리스트의 append() 메소드 등을 이용하여 리스트에 요소를 추가하여 사용한다. append() 메소드에 대해서는 뒤에서 설명할 것이다.

공백 리스트 사용 예	실행 결과
a =[] # 공백 리스트 a 생성 for i in range(10) : a.append(i) # 0~9 까지의 정수를 a 리스트에 추가 print(a)	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

2) 숫자 리스트

- 숫자(정수, 실수 등)로 구성된 리스트로 학생들의 성적을 저장하는 리스트, 월별 매출 리스트, 요일별 기온 리스트 등 다양한 숫자 값들의 집합을 리스트로 구성할 수 있다.
- 예) 성적 리스트 : scores = [90, 100, 89, 87, 70, 95]
- 예) 온도 리스트 : temperatures = [29.5, 30.2, 31.5, 30.8, 29.9, 28.5, 30.0]

3) 문자열 리스트

- 문자열들을 요소로 가지는 리스트들로 부서명 리스트, 팀 리스트, 과목 리스트, 장바구니 리스트 등 다양한 리스트를 구성할 수 있다.
- 예) 월명 리스트 예 : month = [“Jan”, “Feb”, “Mar”, “Apr”, “May”, “Jun”, July“]
- 예) 도시명 리스트 예 : cities = [“서울”, “부산”, “대구”, “인천”, “광주”, “대전”]

❑ 리스트 인덱싱(Indexing)

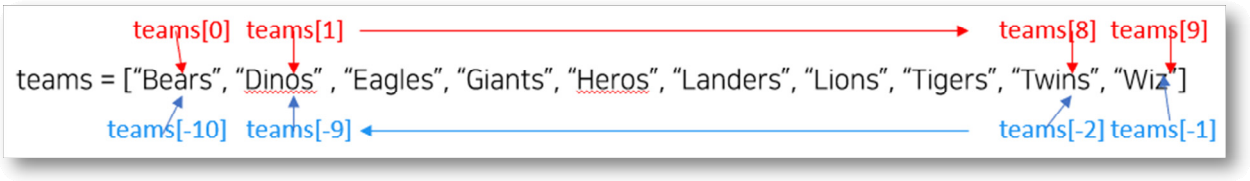
리스트의 요소 각각은 고유한 번호인 인덱스 값을 가진다. 인덱스를 사용하여 특정 요소를 추출하는 것을 인덱싱이라고 하며, 파이썬에서는 맨 앞 요소에서 0 부터 시작하여 1 씩 증가하며 번호가 매겨지는 순방향 인덱싱과 마지막 원소부터 -1 부터 시작하여 맨 앞 요소까지 -1 씩 증가하며 번호가 매겨지는 역방향 인덱싱이 모두 제공된다.

다음은 리스트 인덱싱 사용 예로서 프로야구 10 개 구단명을 저장하고 있는 teams 리스트를 생성한 후, 인덱싱을 사용하여 앞에서부터 3 번째 요소와, 뒤에서부터 10 번째 요소값을 추출하고 있다.

리스트 인덱싱 사용 예

>>> teams = ["Bears", "Dinos", "Eagles", "Giants", "Heros", "Landers", "Lions", "Tigers", "Twins", "Wiz"]
>>> teams[2]
'Eagles'
>>> teams[-10]
'Bears'

teams 리스트는 아래 그림과 같이 순방향 인덱싱으로 0~9 까지, 역방향 인덱싱으로 -1~-10 까지 각각의 요소에 인덱스가 부여되고, 인덱스를 이용하여 각 팀 이름을 추출할 수 있다.



왼쪽은 숫자 리스트를 생성한 후, 인덱싱을 이용하여 해당 위치의 값을 추출하는 예이다.

리스트 인덱싱 사용 예	리스트를 요소로 가지는 리스트 사용 예

>>> a = [1, 2, 3, 4] # 숫자 리스트 생성
>>> a
[1, 2, 3, 4]
>>> a[0] # a의 첫번째 요소 접근
1
>>> a[-1] # a의 마지막 요소 접근
4
	>>> b = [1, 2, 3, ['a', 'b', 'c']]
	>>> b[0]
	1
	>>> b[-1] ← 리스트의 마지막 요소인 ['a' , 'b' , 'c'] 리스트 반환
	['a', 'b', 'c']
	>>> b[-1][0]
	'a' ← 리스트 안에 있는 ['a' , 'b' , 'c'] 리스트의 요소에 접근
	>>> b[-1][-1]
	'c'

오른쪽 예는 리스트의 요소로 리스트가 오는 예제로 서로 다른 자료형들로 이루어진 혼합 리스트의 예이기도 하다. 리스트 b 의 마지막 요소는 ['a', 'b', 'c'] 리스트가 되어, b[-1]을 뽑아내면 ['a', 'b', 'c'] 가 추출되게 된다. b[-1]가 ['a', 'b', 'c'] 이기 때문에, ['a', 'b', 'c']의 각 요소는 인덱싱을 추가로 적용하여 추출할 수 있다. b[-1][0]은 ['a', 'b', 'c'] 중 첫 번째 요소인 'a'를 추출하고, b[-1][-1]은 마지막 요소인 'c'가 추출된다.

❑ 리스트 슬라이싱(Slicing)

리스트도 문자열과 같이 리스트의 요소들을 일부 잘라내는 슬라이싱 연산을 적용할 수 있다. 리스트 슬라이싱의 사용 방법은 다음과 같다.

리스트 슬라이싱 사용법
[형식] *list_name[start : end : step]* **[설명]** *start* : 슬라이싱을 시작하는 시작 위치, 생략시 디폴트 값은 0 *end* : 슬라이싱을 끝낼 마지막 위치로 *end* 는 슬라이싱 결과에 미포함 *step* : 몇 개씩 끊어서 가져올지와 방향을 정함. 생략 시 디폴트 값은 1

간단한 리스트 슬라이싱 예를 살펴보자. 왼쪽은 list1 의 각 요소의 순방향 인덱스와 역방향 인덱스를 표시한 것이다. 리스트 슬라이싱 첫 번째 예인 list1[0:2] 는 list1 의 0 번 요소에서 2 번 바로 앞 원소까지 원소들을 추출하여 리스트로 만드는 예이다. 그 결과, ['a', 'b'] 서브 리스트가 생성되었다.

리스트 인덱싱	리스트 슬라이싱 사용 예

list1 = ['a', 'b', 'c', 'd', 'e']
// Index References
| a | b | c | d | e |
| 0 | 1 | 2 | 3 | 4 |
|-5 |-4 |-3 |-2 |-1 |
	>>> list1 = ['a', 'b', 'c', 'd', 'e']
	>>> list1[0:2]
	['a', 'b']
	>>> list1[:2] # 처음부터 list1[1]까지
	['a', 'b']
	>>> list1[2:] # list1[2]부터 마지막 까지
	['c', 'd', 'e']
	>>> list1[-4:-2] # list1[-4]에서 list1[-2] 까지
	['b', 'c']
	>>> list1[3:0:-1] # 인덱스 1~3까지 요소를 거꾸로 가져오기
	['d', 'c', 'b']

순방향 리스트 슬라이싱 시, 앞의 인덱스가 생략되면 맨 앞에서부터 슬라이싱을 진행하고, 마지막 인덱스가 생략되면, 마지막 원소까지 포함하여 슬라이싱을 진행한다.

list1[-4:-2] 는 역방향 인덱싱 번호 -4 번에 해당하는 'b'에서부터 -2 에 해당하는 'd' 까지 순방향으로 슬라이싱을 진행한다. 반면에 list1[3:0:-1]은 3 번 위치에 해당하는 'd'에서 0 번 위치에 있는 'a'까지 역방향으로 -1 간격으로 슬라이싱을 진행하여, ['d', 'c', 'b'] 가 추출된다.

❑ 리스트 내용 변경하기

리스트의 내용을 변경하려면 아래 예제와 같이 인덱싱과 대입문을 이용하여 원하는 요소의 값을 직접 변경하면 된다.

예제	실행 결과

fruits = ["apple", "banana", "orange"]
print(fruits) #리스트 변경 전
fruits[1] = "grape"
print(fruits) #리스트 변경 후
	['apple', 'banana', 'orange']
	['apple', 'grape', 'orange']

❑ 리스트 전용 메소드(Method)

리스트의 경우에는 print(x), len(x) 등과 같은 내장함수 외에도 append(), insert(), sort() 등 리스트를 위한 다양한 전용 함수들이 제공된다. 이들은 리스트 자료구조에 속해있는 함수로 리스트의 메소드라고 부른다. 내장함수에서는 함수명만을 이용해서 함수를 호출하는 것과 달리 리스트 메소드는 *list 명.메소드명* 형식으로 리스트 변수 이름을 먼저 쓰고 도트(.)을 찍은 후, 함수명을 적는다. 리스트에서 많이 사용되는 메소드들은 다음과 같다.

메소드명	설명
append()	리스트의 끝에 요소를 추가한다.
clear()	리스트의 모든 요소들을 삭제한다.
copy()	리스트의 복사본을 반환한다.
count()	특정 값을 갖는 요소들의 수를 반환한다.
extend()	리스트의 요소들을 현재 리스트의 끝에 추가한다.
index()	특정 값을 갖는 첫 번째 요소의 인덱스를 반환한다.
insert()	특정 위치에 하나의 요소를 삽입한다.
pop()	특정 위치에 있는 요소를 삭제한다.
remove()	특정 값을 갖는 요소를 삭제한다.
reverse()	리스트의 순서를 반대로 바꾼다.
sort()	리스트를 정렬한다.

❑ 리스트의 마지막에 요소 추가하기

리스트의 마지막에 요소를 삽입하려면, 다음 예와 같이 append() 메소드를 사용하면 된다. 3 행의 fruits.append("orange")를 실행한 결과, 마지막에 "orange"가 하나 더 추가된 것을 실행 결과에서 확인할 수 있다.

예제	실행 결과

fruits = ["apple", "banana", "orange"]
print(fruits) #리스트에 원소 추가 전
fruits.append("orange")
print(fruits) #리스트에 원소 추가 후
	['apple', 'banana', 'orange']
	['apple', 'banana', 'orange', 'orange']

이번에는 공백 리스트에 사용자로부터 값을 입력받아서 요소를 추가하는 예제를 살펴보자.

예제	실행 결과

scores = [ ]
 	
for i in range(10):
  data = int(input("성적을 입력하시오 : "))
  scores.append(data)
 	
print(scores)
	성적을 입력하시오 : 89
	성적을 입력하시오 : 90
	성적을 입력하시오 : 78
	성적을 입력하시오 : 100
	성적을 입력하시오 : 78
	성적을 입력하시오 : 81
	성적을 입력하시오 : 94
	성적을 입력하시오 : 90
	성적을 입력하시오 : 80
	성적을 입력하시오 : 70
	[89, 90, 78, 100, 78, 81, 94, 90, 80, 70]

내용이 비어있는 scores 리스트를 생성한 후에, 10 회 반복하면서 성적을 입력받아서 scores.append(data)를 통해서 scores 리스트에 입력된 값을 요소로 추가한다. 반복문이 끝난 후에 scores 리스트를 출력해보면, 입력된 값들이 모두 추가되어 있는 것을 확인할 수 있다.

❑ 리스트의 특정 위치에 요소 삽입하기

리스트의 특정 위치에 요소를 삽입하려면, insert() 메소드를 이용하여 원하는 위치 바로 앞에 값을 삽입하면 된다.

예제	실행 결과

fruits = ["apple", "banana", "orange"]
print(fruits) #리스트 삽입 전
fruits.insert(0, "tomato")
print(fruits) # 토마토 삽입 후
fruits.insert(2, "cherry")
print(fruits) # 체리 삽입 후
	['apple', 'banana', 'orange']
	['tomato', 'apple', 'banana', 'orange']
	['tomato', 'apple', 'cherry', 'banana', 'orange']

위 예에서는 과일명들의 리스트인 fruits 를 생성한 후, fruits.insert(0, “tomato”)를 호출하여, 0 번 위치 앞, 즉 맨 앞에 “tomato” 요소를 삽입하고 있다. 삽입 후 바뀐 리스트를 출력해 보면 맨 앞에 “tomato”가 추가된 것을 확인할 수 있다.

❑ 리스트 요소 삭제하기

리스트의 요소를 삭제하는 방법은 remove() 메소드를 이용하여 특정값을 갖는 첫 번째 요소를 삭제하는 방법과 pop() 메소드를 이용하여 특정 인덱스 위치에 있는 항목을 삭제하는 방법이 있다.

remove() 사용 예제	실행 결과
fruits = ["apple", "banana", "cherry"] print(fruits) fruits.remove("banana") print(fruits)	['apple', 'banana', 'cherry'] ['apple', 'cherry']

위 예에서는 fruits.remove("banana")를 호출하여 fruits 리스트에서 "banana" 값을 갖는 첫 번째 요소가 삭제된 것을 확인할 수 있다.

이번에는 pop() 메소드를 사용하여 특정 위치의 요소를 삭제하는 방법을 살펴보자.

pop() 사용 예제	실행 결과

fruits = ["apple", "banana", "cherry", "tomato"]
print(fruits)
fruits.pop(0)
print(fruits)
fruits.pop(2)
print(fruits)
fruits.pop() # 인덱스를 안주면 마지막 항목 삭제
print(fruits)
	['apple', 'banana', 'cherry', 'tomato']
	['banana', 'cherry', 'tomato']
	['banana', 'cherry']
	['banana']

위 예에서는 첫 번째 pop() 메소드 호출에서 0 번 인덱스 값을 삭제하여 "apple" 값을 갖는 요소가 리스트에서 제거되었다. 두 번째 fruits.pop(2) 에서는 2 번 인덱스에 해당하는 "tomato" 가 삭제되었다. 마지막으로 인덱스없이 호출된 fruits.pop()에서는 마지막 항목이 삭제되어 최종적으로 "banana"만 리스트에 남게된다. 이와 같이 pop()에서 인덱스가 생략되는 경우에 디폴트 값인 마지막 항목이 삭제된다.

❑ 리스트 관련 내장함수

리스트 전용 메소드 외에도 내장함수들을 사용하여 리스트에 대해 다양한 작업을 수행할 수 있다. 다음은 리스트에 적용 가능한 주요 내장함수들의 예이다.

내장 함수명	설명
round()	주어진 자리 수까지 반올림한 값을 반환한다.
reduce()	특정 함수를 리스트 안의 모든 요소에 적용하여 결과값을 저장하고 최종 합계 값만을 반환한다.
sum()	리스트 안의 숫자들을 모두 더한다.
ord()	유니코드 문자의 코드값을 반환한다.
cmp()	첫번째 리스트가 두 번째 리스트보다 크면 1을 반환한다.
max()	리스트 요소들의 최대값을 반환한다.
min()	리스트 요소들의 최소값을 반환한다.
all()	리스트의 모든 요소의 값이 True이면 True를 반환한다.
any()	리스트안의 한 요소라도 True 이면 True를 반환한다.
len()	리스트의 길이를 반환한다.
enumerate()	리스트의 각 요소들에 순차 번호를 붙여서 튜플로 만든 리스트를 생성한다.
accumulate()	누적된 합 또는 누적된 함수 적용 결과를 반환하는 이터레이터(iterator)를 만든다.
filter()	리스트의 각 요소가 참인지 아닌지를 검사한다.
map()	특정한 함수를 리스트의 각 요소에 적용하고 결과를 담은 리스트를 반환한다.

❑ 리스트 합 구하기, 최대/최소값 찾기

sum(), len(), min(), max() 내장함수를 이용하여 리스트 요소들의 합과 평균, 최대/최소값을 구하는 간단한 예를 살펴보자.

예제	실행 결과
scores=[90, 89, 76, 88, 93, 67, 70, 80] print("평균 =", sum(scores) / len(scores)) print("최고점 =", max(scores)) print("최저점 =", min(scores))	평균 = 81.625 최고점 = 93 최저점 = 67

평균을 구하기 위해, sum() 함수를 이용하여 scores 리스트 요소들의 합을 구한 후, len() 함수로 리스트 요소의 개수를 구하여 나누면 평균값을 구할 수 있다.

❑ 리스트 요소 하나씩 방문하기

for 문과 in 연산자를 사용하여 리스트의 요소를 하나씩 방문하는 방법을 알아보자.

방법1	방법2	실행 결과

x = [1, 2, 3, 4, 5, 6]
for element in x :
  print(element, end = ',')
	x = [1, 2, 3, 4, 5, 6]
	for i in range(len(x)) :
	  print(x[i], end = ',')
		1,2,3,4,5,6,

방법 1 은 반복을 수행하면서 리스트안에 있는 원소를 하나씩 element 변수에 받아서 출력을 하는 예이다. 방법 2 는 range() 함수로 인덱스 리스트 [0, 1, 2, 3, 4, 5]를 생성한 후에, for 문을 돌면서 i 에 인덱스를 하나씩 받아서 x[i]로 값을 추출하여 출력을 하고 있다. 방법 2 에서 사용된 len(x)는 리스트 x 의 길이를 반환한다. 두가지 모두, 결과로 리스트의 원소를 하나씩 출력하고 있다.

❑ 리스트 안 요소 찾기

간단한 예를 통해 if 문과 in 연산자를 사용하여 리스트안에 특정 원소가 있는지를 알아내는 방법을 알아보자.

예제	실행 결과
students = ["김철수", "이영희", "오선우", "김진태", "신미나"] if "이영희" in students : print("이영희는 우리반 학생입니다")	이영희는 우리반 학생입니다

위 예에서 학생 이름 리스트인 students 안에 “이영희”가 있는지를 알아내기 위해 if 와 in 문장을 이용하여, in 다음에 오는 리스트에 if 문 다음의 요소와 일치하는 값이 있으면 True 가 되어, print 문을 실행하게 된다.

❑ 2 차원 리스트

지금까지 살펴본 리스트는 요소들이 순서대로 나열되어 있는 형태의 1 차원 리스트이다. 파이썬에서는 1 차원 리스트들을 요소로 하는 2 차원 리스트를 만들 수 있다. 2 차원 리스트는 행과 열로 구성된 표 형태의 자료를 관리하는데 적합하다. 2 차원 리스트를 확장하여 3 차원, 4 차원 리스트도 만들 수 있으나, 이 책의 범위를 벗어나므로 가장 많이 활용되는 2 차원 리스트에 대해서만 간단하게 알아보기로 한다.

아래 그림 왼쪽은 행과 열로 구성된 표 형태의 자료구조를 도식화한 것이다. 이와 같은 표 형태의 자료는 파이썬에서 오른쪽과 같은 리스트의 리스트 형태로 구현된다.

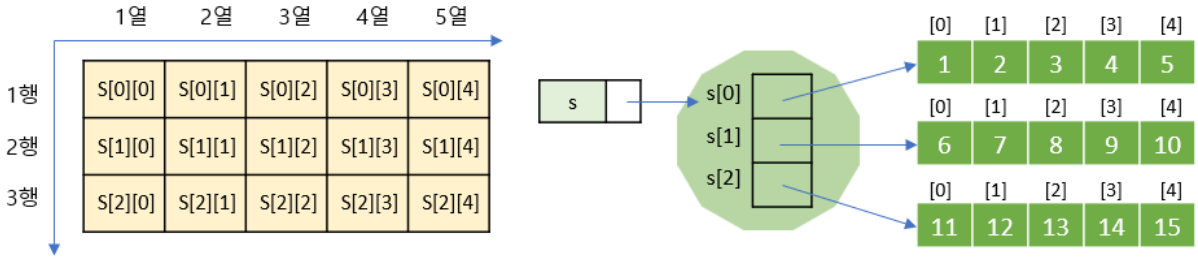


그림 39 2차원 리스트 개념 및 구현 방식

즉, 각 행이 1 차원 리스트가 되고, 1 차원 리스트의 이름은 s[0], s[1], s[2]가 된다. 2 차원 리스트 s 는 다시 s[0], s[1], s[2]를 요소로 가지게 되어 오른쪽 그림과 같은 리스트의 구성이 완성된다. 아래 예제는 위 그림의 리스트를 파이썬 코드로 구현한 것이다.

리스트 생성 예제

```
# 2차원 리스트를 생성한다.
s= [ [1, 2, 3, 4, 5],
     [6, 7, 8, 9, 10],
     [11, 12, 13, 14, 15] ]

print(s)
```

실행 결과

```
[[1, 2, 3, 4, 5], [6, 7, 8, 9, 10], [11, 12, 13, 14, 15]]
```

원소 출력 예제

```
s = [ [ 1, 2, 3, 4, 5 ],
      [ 6, 7, 8, 9, 10 ],
      [11, 12, 13, 14, 15 ] ]

rows = len(s)    # 행의 개수
cols = len(s[0]) # 열의 개수

for r in range(rows):
    for c in range(cols):
        print(s[r][c], end=",")
    print()
```

실행 결과

```
1,2,3,4,5,
6,7,8,9,10,
11,12,13,14,15,
```

가장 왼쪽의 코드는 2 차원 리스트를 생성하고 바로 출력하는 예로서, 리스트 안에 리스트들이 요소로 들어있는 구조를 확인할 수 있다. 두 번째 예제는 동일한 리스트에 대해 이중 for 문을 돌면서 하나씩 원소를 출력하는 예제이다. 행의 개수는 len(s)를 통해 s 의 길이를 구하면 알 수 있는데, s 의 요소는 s[0], s[1], s[2] 이므로 rows 에 3 이 저장된다. 열의 개수는 3 개의 행이 동일한 크기를 가지므로, len(s[0])를 통해 첫 번째 행인 s[0]의 요소 개수를 구하여 cols 에 저장하면 된다. 이중 for 문에서는 각 행마다 열의 개수만큼 반복하면서, 변수 r, c 값과 리스트 인덱싱을 통해 s[r][c]에 해당하는 개별 요소값을 출력한다.

STEP 04

튜플

파이썬 튜플(Tuple)은 자료들의 순서있는 모임으로 리스트와 유사하나, 리스트와 다르게 생성 후에 요소의 삽입/삭제/수정이 불가능하다. 이 절에서는 튜플의 개념과 기본적인 사용법을 간단한 예제를 통하여 알아보자.

❑ 튜플이란?

파이썬 튜플은 내용을 변경할 수 없는 시퀀스 자료형이다. 리스트와 유사하지만, 튜플의 요소들을 묶을 때, [ ] 대신 ( ) 기호를 사용하며, 한번 생성되면 값을 추가하거나 수정, 삭제할 수 없다.

파이썬 튜플 사용법

[형식] *튜플명* = (*요소1, 요소2, 요소3,…*)

[설명] 튜플 구성 요소들을 ( ) 안에 콤마로 구분하여 순서대로 나열하여 생성한다.
괄호 ( )를 생략하는 것도 가능하다. (예) t1 =1, 2, 3
튜플 요소로 모든 자료형이 올 수 있다.
생성된 튜플을 *튜플명* 에 해당하는 변수와 연결하여 사용한다.

[예제] days=('월', '화', '수', '목', '금', '토', '일') # 요일명을 저장하는 문자열 튜플 days 생성

❑ 튜플 생성하기

다음은 튜플을 생성하는 다양한 예를 살펴보기로 하자.

튜플 생성 예

```
>>> t1 = ()
>>> t2 = (1,)  # 튜플의 원소가 한개일때는 ','를 붙여야 함
>>> t3 = (1, 2, 3)
>>> t4 = 1, 2, 3
>>> t5 = ('a', 'b', (1, 2))
```

첫 번째는 공백 튜플 t1 을 생성하는 코드이다. 리스트와 달리 튜플은 요소 추가가 불가능하므로 t1 은 공백 리스트로 그대로 사용되어야만 한다. 두 번째 예에서와 같이 튜플의 요소가 1 개인 경우에, 일반 수식에서 사용되는 괄호와 구분하기 위해 요소 다음에 반드시 콤마(,)를 붙인 후, 괄호를 닫아야 한다. 콤마 없이 t2 = (1)과 같이 쓰는 경우, t2 는 튜플이 아닌 1 이라는 정수 값을 가지는 변수가 된다. t3 = (1, 2, 3) 과 t4 = 1, 2, 3 은 괄호의 유무에 차이가 있지만, t3 와 t4 는 동일한 내용의 튜플이 생성된다. 마지막의 t5 생성 예제는 튜플의 요소로 또 다른 튜플이 올 수 있음을 보여주는 예이다.

❑ 튜플 인덱싱(Indexing)

리스트와 마찬가지로 튜플의 각 요소에도 왼쪽에서 오른쪽 방향으로 0 부터 시작하는 고유한 번호가 부여된다. 오른쪽부터 왼쪽 방향으로는 -1 부터 시작해서 -1 씩 증가되는 음의 인덱스가 부여된다.

튜플 인덱싱 사용 예

```
>>> days = ('월', '화', '수', '목', '금', '토', '일')
>>> days[0]
'월'
>>> days[6]
'일'
```

```
>>> t1 = (1, 2, 3)
>>> t1[0] = 10
Traceback (most recent call last):
  File "<pyshell#19>", line 1, in <module>
    t1[0] = 10
TypeError: 'tuple' object does not support item assignment
```

왼쪽의 예는 요일명을 가지고 있는 days 튜플을 생성하고, days[0]을 통해서 첫 번째 요소값과 days[6]을 통해 마지막 요소값을 추출하는 코드이다. 오른쪽 예는 튜플 t1 을 생성한 후, t1[0] = 10 을 통해서 0 번째 요소값을 변경하려고 시도했을 때, 오류가 발생하는 상황을 보여준다. 오류 설명으로 튜플 객체는 항목 대입이 지원되지 않는다는 내용이 출력되고 있다.

❑ 튜플 슬라이싱(Slicing)

리스트와 마찬가지로 튜플도 특정 범위에 속하는 요소들을 잘라낼 수 있다.

튜플 슬라이싱 사용 예

```
>>> days = ('월', '화', '수', '목', '금', '토', '일')
>>> weekdays = days[:5]
>>> print(weekdays)
('월', '화', '수', '목', '금')
```

```
>>> weekends = days[5:]
>>> print(weekends)
('토', '일')
```

요일명을 저장하는 days 튜플을 생성한 후에, days[:5]를 통해서 맨 앞의 '월'에서부터 4 번 인덱스에 해당하는 '금'까지 요소들을 추출하여 weekdays 튜플을 생성하고 있다. 튜플 슬라이싱을 할 때 맨 앞의 요소 인덱스가 생략되면 0 번째 요소부터 슬라이싱이 진행된다. 오른쪽 예에서는 days[5:] 로 5 번 인덱스에 해당하는 '토'부터 끝 인덱스가 생략되었으므로 디폴트값이 맨 마지막 요소인 '일'까지 슬라이싱이 진행되어 weekends 튜플이 생성되었다.

❑ 튜플 길이 구하기

리스트와 마찬가지로 튜플도 내장함수 len()을 이용하여 튜플의 길이를 구할 수 있다.

튜플 길이 구하기 예

```
>>> t1 = (1, 2, ('a', 'b'), ('가', '나'))
>>> t1
(1, 2, ('a', 'b'), ('가', '나'))
```

```
>>> len(t1)
4
>>> len(t1[2])
2
```

먼저 튜플 안에 또 다른 튜플을 요소로 가지는 t1 을 생성한 후, 오른쪽 맨 위 코드와 같이 len() 함수를 이용하여 t1 의 길이를 구하면 튜플안에 포함된 각 튜플을 1 개의 요소로 계산하여 4 가 나온다. 다시 t[2]의 길이를 구하면, t[2]는 t1 의 요소인 ('a', 'b')에 해당하므로 2 개의 요소를 가지므로 길이는 2 가 된다.

❑ 튜플의 덧셈과 곱셈

튜플에 대한 덧셈과 곱셈 연산은 어떤 의미를 가지는지를 예제를 통해 알아보자. 튜플에 대한 덧셈 연산은 피연산자로 사용된 두 개의 튜플이 연결되는 결과가 나오고, 튜플에 대한 곱셈은 첫 번째 피연산자로 사용된 튜플의 요소가 두 번째 피연산자로 오는 숫자만큼 반복되는 결과가 나온다.

튜플의 덧셈과 곱셈 예

```
>>> t1 = (1, 2)
>>> t2 = (3, 4)
>>> t3 = t1 + t2
>>> t3
(1, 2, 3, 4)
```

```
>>> t1 = ('a', 'b')
>>> t2 = t1 * 3
>>> t2
('a', 'b', 'a', 'b', 'a', 'b')
```

왼쪽 예에서 t1 과 t2 에 대한 덧셈 결과로 t1, t2 튜플 요소들이 연결된 새로운 튜플이 생성됨을 확인할 수 있다. 오른쪽 예에서는 t1 * 3 연산의 결과로 t1 의 각 요소들이 3 번 반복된 새로운 튜플이 생성되었다.

딕셔너리

STEP 05

파이썬 딕셔너리(Dictionary)는 Key 와 Value 쌍을 요소로 갖는 시퀀스 자료형으로 삽입/삭제/변경이 가능하다. 이 절에서는 딕셔너리 자료형의 특징과 기본적인 사용법에 대하여 알아보자.

❑ 딕셔너리란?

딕셔너리는 Key 와 Value 를 한쌍으로 갖는 자료형으로 아래 그림과 같은 대응 관계를 나타내는데 적합하며, Key 값을 통해 Value 를 바로 얻을 수 있다.

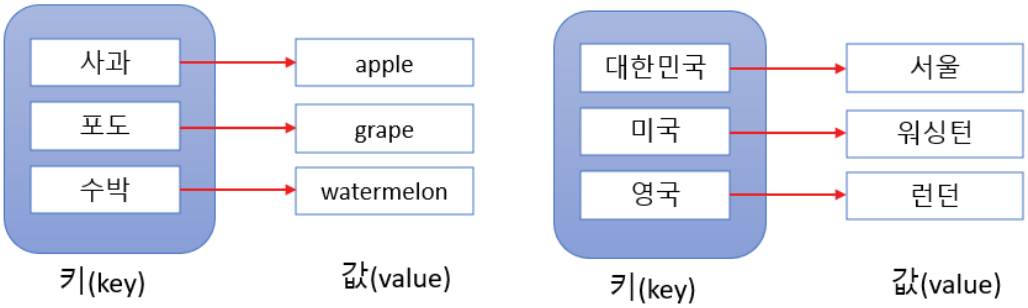


그림 40 딕셔너리로 표현하기에 적합한 대응 관계 예제

위 그림에서 왼쪽은 한글 과일명-영어 과일명 사이의 대응 관계를, 오른쪽은 국가명-수도명의 대응 관계를 보여주고 있다. 이와 같은 대응 관계는 딕셔너리 자료형을 이용하면 쉽게 관리할 수 있다. 파이썬 딕셔너리는 아래와 같이 중괄호({ })안에 key:value 쌍을 순서대로 나열하여 생성할 수 있다.

파이썬 딕셔너리 사용법

[형식] *딕셔너리명* = {*key1:value1, key2:value2, key3:value3,…*}
[설명] 딕셔너리 구성 요소(key:value 쌍)들을 { } 안에 콤마로 구분하여 순서대로 나열하여 생성한다.
생성된 딕셔너리를 *딕셔너리명* 에 해당하는 변수와 연결하여 사용한다.
[예제] fruits = {"사과":"apple", "포도":"grape", "수박":"watermelon"} # 과일명 딕셔너리 생성
capitals = {"대한민국":"서울", "미국":"워싱턴", "영국":"런던"} # 각 나라의 수도 저장 딕셔너리

마지막에 제시된 예제는 위 그림에 해당하는 fruits 와 capitals 딕셔너리를 생성하는 코드이다.

딕셔너리에서 Key 는 고유한 값이므로 중복되는 Key 값을 설정해 놓으면 하나를 제외한 나머지 것들이 모두 무시된다는 점에 주의해야 한다. 다음 예에서 볼 수 있듯이 동일한 Key 가 2 개 존재할 경우, 앞에 오는 1: 'A' 와 2:'B' 쌍이 무시되고, 맨 나중에 정의한 key:value 쌍만 남는다.

Key 중복 사용 예

```
a ={1:'A', 1:'a', 2:'B', 2:'b'}
a
{1: 'a', 2: 'b'}
```

❑ 딕셔너리에서 Key 사용하여 Value 얻기

딕셔너리는 Key 를 이용하여 Key 에 해당하는 값을 얻어올 수 있는데, 인덱싱을 사용하거나 딕셔너리 자료형에서 제공되는 get() 메소드를 사용하는 방법이 있다.

1) 인덱싱 사용하기

Key 값을 이용하여 인덱싱을 하면, 대응되는 Value 값을 얻을 수 있다. 아래 예에서 (학생 이름 : 점수) 쌍으로 이루어진 grade 딕셔너리를 생성한 후에, Key 값인 학생 이름(성)을 이용하여 grade 에 대한 인덱싱을 수행하면, 해당 Key 값에 대한 Value, 즉 점수가 추출된다.

인덱싱 사용 예

```
>>> grade = {'kim' : 95, 'lee' : 90, 'song' : 86, 'park' : 89}
>>> grade['park']  # key 가 'park'인 딕셔너리의 value를 반환
89
>>> grade['kim']  # key 가 'kim'인 딕셔너리의 value를 반환
95
>>> grade['seol']  # key 가 'seol'인 딕셔너리의 value를 반환
Traceback (most recent call last):
  File "<pyshell#15>", line 1, in <module>
    grade['seol']  # key 가 'seol'인 딕셔너리의 value를 반환
KeyError: 'seol'
```

마지막 grade['seol'] 문장 실행 결과는 grade 딕셔너리에 'seol'에 해당되는 Key 값이 없으므로 KeyError 가 발생하여 오류 메시지가 출력된다.

2) get() 메소드 사용하기

딕셔너리 자료형에서 제공되는 get() 메소드를 사용하면, 인수로 주어진 Key 값에 대응되는 Value 가 반환된다. 만약 딕셔너리에 Key 값에 해당되는 요소가 없으면 None 이 반환된다.

get() 메소드 사용 예

```
>>> student = {'id':'2021-01', 'name':'김철수', 'phone':'010-2222-3333'}
>>> student.get('name')
'김철수'
>>> student.get('phone')
'010-2222-3333'
```

위 예제는 학생 정보를 저장하기 위한 딕셔너리 student 를 생성한 후, Key ‘name’과 ‘phone’에 해당되는 Value 값을 get 메소드를 이용하여 추출하는 과정을 보여준다.

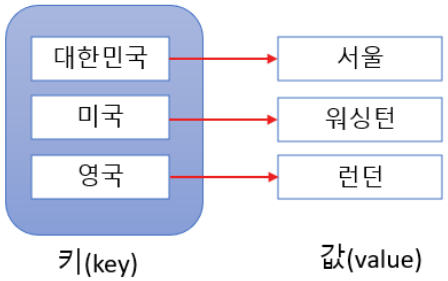
❑ 딕셔너리 항목 추가하기

딕셔너리에 항목을 추가하려면, Key 값을 이용한 인덱싱과 대입문을 이용하여 새로운 항목을 추가할 수 있다. 아래 예에서는 먼저 공백 상태인 capitals 딕셔너리를 생성한 후, capitals[“대한민국”] = “서울” 과 같은 형태로 키값을 이용한 인덱싱과 대입문으로 대응되는 값을 넣어주어 (Key:Value) 쌍을 추가하고 있다. 오른쪽 그림은 코드 실행 후 딕셔너리의 상태이다.

딕셔너리 항목 추가 코드 예

```
>>> capitals = {}
>>> capitals["대한민국"] = "서울"
>>> capitals["미국"] = "워싱턴"
>>> capitals["영국"] = "런던"
>>> print(capitals)
{'대한민국': '서울', '미국': '워싱턴', '영국': '런던'}
```

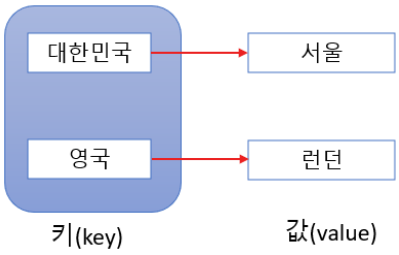
코드 실행 후, 딕셔너리 상태



❑ 딕셔너리 항목 삭제하기

딕셔너리에서 항목을 삭제하려면 아래 예와 같이딕셔너리 자료형에서 제공되는 pop 메소드를 사용하면 된다. city = capitals.pop(“미국”)을 실행하면, 해당 항목이 딕셔너리에서 삭제되고, Key 값이 “미국”인 항목의 Value 값이 반환되어 city 에 저장된다. 오른쪽 그림은 코드 실행 후 딕셔너리의 상태이다.

딕셔너리 항목 추가 코드 예	코드 실행 후, 딕셔너리 상태

>>> city = capitals.pop("미국")
>>> print(city)
워싱턴
>>> print(capitals)
{'대한민국': '서울', '영국': '런던'}
	

❑ 딕셔너리 요소 방문하기

for 반복문과 인덱싱을 사용하여 딕셔너리를 순차적으로 방문하여 요소값을 알아낼 수 있다.

딕셔너리 요소 방문 예	실행 결과
capitals = {"대한민국":"서울", "미국":"워싱턴", "영국":"런던", "프랑스":"파리"} for key in capitals: print(key, ":", capitals[key])	대한민국 : 서울 미국 : 워싱턴 영국 : 런던 프랑스 : 파리

위 예에서는 (나라명, 수도명) 으로 이루어진 딕셔너리 capitals 를 생성한 후, for 문을 실행하면서 매 반복 때마다 현재 방문하고 있는 요소의 Key 값을 key 변수에 저장한 후, key 변수로 인덱싱을 수행하여 대응되는 Value 값을 얻어내어 출력하고 있다.

❑ 딕셔너리 key, value 목록 얻기

딕셔너리 자료형의 keys() 메소드를 이용하면 해당 딕셔너리의 Key 목록을, values() 메소드를 이용하면 Value 목록을 알아낼 수 있다.

딕셔너리 Key, Value 목록 출력 예

>>> capitals = {"대한민국":"서울", "미국":"워싱턴", "영국":"런던", "프랑스":"파리"}
>>> print(capitals.keys())
dict_keys(['대한민국', '미국', '영국', '프랑스'])
>>> print(capitals.values())
dict_values(['서울', '워싱턴', '런던', '파리'])

위 예제에서는 capitals 딕셔너리를 생성한 후, capitals.keys() 메소드를 호출하여 capitals 의 Key 값만을 모아 dict_keys 객체를 반환하여 출력한다. 마찬가지로 capitals.values() 메소드를 호출하여 capitals 의 Value 값들만 모아놓은 dict_values 객체를 반환받아 출력하고 있다.

❑ 딕셔너리의 모든 요소 삭제하기

딕셔너리 자료형의 clear() 메소드를 사용하면 딕셔너리의 모든 요소들을 삭제할 수 있다. 빈 딕셔너리는 { }로 표현된다.

딕셔너리 요소 삭제 예

```
>>> capitals.clear()
>>> capitals
{}
```

8장

알고리즘과 프로그래밍

알고리즘

STEP 01

알고리즘(algorithm)은 문제해결을 위한 방법과 단계적인 절차로서, 알고리즘을 프로그램 언어로 구현하는 과정인 프로그래밍을 통해 컴퓨터에서 실행 가능한 프로그램이 만들어진다. 이 절에서는 알고리즘의 개념과 표현 방법에 대해서 자세히 살펴보기로 한다.

❑ 알고리즘이란?

알고리즘은 문제해결을 위한 방법과 단계적인 절차이다. 알고리즘은 컴퓨터를 통한 문제해결 뿐만 아니라 일상 생활 속의 문제를 해결하는데도 많이 활용된다. 예를 들어 우리가 요리를 할 때 참고하는 요리법들도 모두 알고리즘에 해당된다. 다음은 쿠키를 만드는 알고리즘 예시이다.

쿠기 만드는 알고리즘 예
① 실온에 두었던 무염 버터를 보울(bowl)에 부드럽게 풀어준다. ② 설탕을 두 번 나누어 버터와 잘 섞어준다. ③ 실온에 두었던 달걀을 풀어 달걀물을 만든 후, ②번의 반죽에 넣고 하나로 섞어준다. ④ 박력분, 베이킹파우더, 소금을 체에 곱게 내려 준비한다. ⑤ ④번의 가루를 ③번의 반죽에 넣고 잘 섞어준다. ⑥ 반죽을 한덩어리로 뭉쳐 위생 백에 담아 냉장고에서 20~30분 휴지시킨다. ⑦ 냉장고에서 꺼낸 반죽을 5~6mm 두께로 균일하게 밀어준다. ⑧ 원하는 모양의 쿠키틀을 이용하여 모양을 찍어준다. ⑨ 170도로 예열된 오븐에서 13분정도 구워준다.

위 예시 이외에도 다양한 방법으로 쿠키를 만들 수 있듯이, 어떤 문제를 해결하는 알고리즘도 여러 가지 방법이 있을 수 있다. 알고리즘에서 가장 중요한 것은 정확성과 효율성이므로 같은 문제를 정확하게 해결할 수 있는 여러 알고리즘 중에서 알고리즘 평가를 통해 가장 효율적인 알고리즘을 선택하는 것이 중요하다.

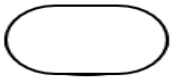
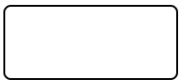
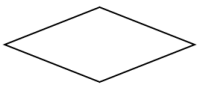
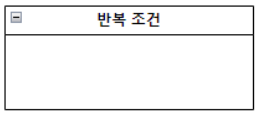
❑ 알고리즘 기술 방법

알고리즘 기술 방법으로는 자연어, 순서도(Flowchart), 의사코드(Pseudo code)가 있다.

- 자연어 : 쿠키 만드는 알고리즘처럼 사람이 사용하는 언어와 비슷하게 일의 순서로 표현하는 방법으로 누구나 알고리즘을 쉽게 작성할 수 있으나 절차의 구조를 체계적으로 파악하기 어렵다.
- 순서도 : 도형과 화살표를 이용하여 알고리즘을 표현하는 방법으로 플로차트, 흐름도라고도 한다. 절차의 구조를 파악하기에는 쉬우나 알고리즘이 복잡해지면 표현하기가 힘들다는 단점이 있다.
- 의사코드 : 간단한 한글이나 영어로 일의 순서를 한줄씩 적어 표현하는 방법으로 자연어보다는 더 간결하게 알고리즘의 핵심 부분을 구조적으로 표현할 수 있고, 프로그래밍 언어보다는 작성하기가 더 쉽기 때문에 주로 많이 사용되는 방법이다.

❑ 순서도

순서도는 알고리즘에서의 작업 순서를 처리 유형별로 여러 가지 약속된 모양의 도형으로 그리고 순서대로 화살표로 연결하여 표현한다. 순서도에서 주로 사용되는 도형과 기호는 아래 표와 같다.

기호	명칭	설명
	시작 / 종료	시작과 종료를 나타낸다.
	흐름선	제어의 흐름과 실행 순서를 나타낸다.
	처리	각종 연산 및 처리를 나타낸다.
	입출력	데이터의 입력 및 출력을 나타낸다.
	문서 출력	문서를 통한 입출력을 나타낸다.
	준비	초기화 작업을 나타낸다.
	판단	조건 판단을 나타낸다.
	순환 반복	조건을 만족하면 반복한다.

순차구조 알고리즘

STEP 02

이 절에서는 3 장 1 절에서 살펴본 3 가지 기본 제어구조 중에서 순차구조 알고리즘을 지금까지 배운 파이썬 문법들을 이용하여 프로그램으로 구현하는 방법을 연습해보자.

❑ 3 가지 기본 제어구조

프로그램의 실행 순서를 제어하기 위한 제어구조는 다음과 같이 순차구조, 선택구조, 반복구조로 나눌 수 있다.

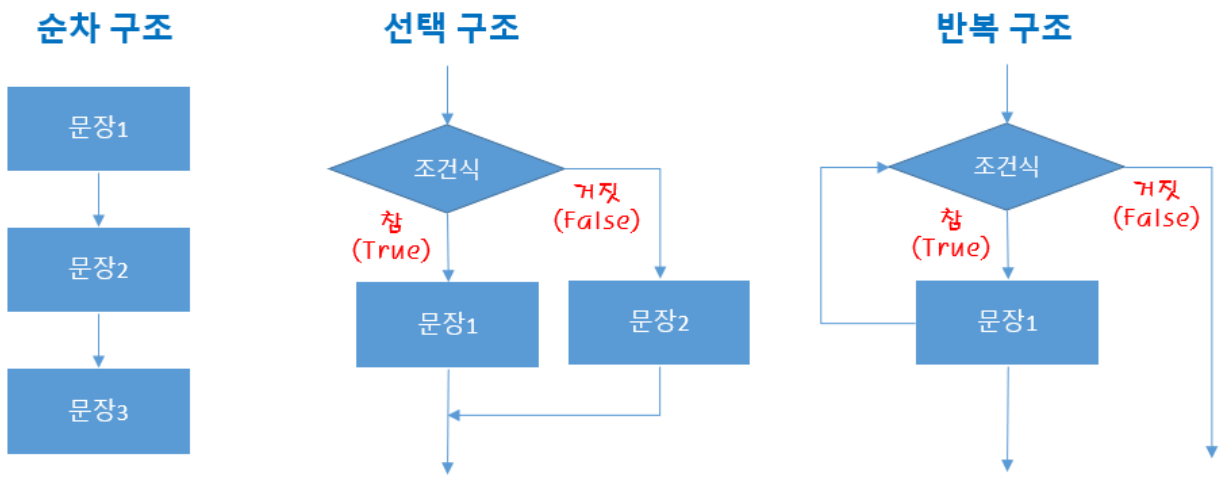
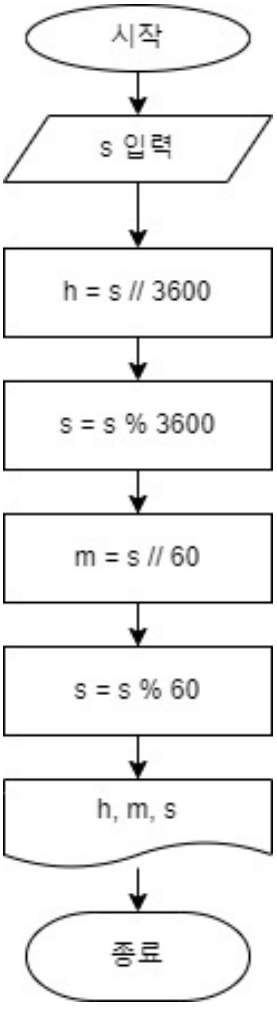


그림 41 3가지 기본 제어 구조

순차구조는 가장 기본적인 제어구조로 프로그램에 작성된 순서대로 문장들을 위에서 아래 방향으로 코드를 실행한다. 선택구조는 조건식의 값을 판별하여 참(True)과 거짓(False)인 경우로 나누어 실행될 문장을 선택하는 구조이다. 반복구조는 조건식의 값이 참이면 문장을 반복해서 실행하고, 거짓이면 반복을 종료하는 구조이다.

❑ 예제 1 : 초 단위 시간을 시간, 분, 초로 변환하기

초 단위 시간을 입력받아서 시간, 분, 초 값으로 변환하여 출력하는 알고리즘을 순서도로 표현하고, 프로그램으로 구현해보자.

<table>
<tr><th>순서도</th><th>의사 코드</th></tr>
<tr><td rowspan="4"> 
</td><td>① 사용자로부터 초단위 시간을 입력받아 s에 저장한다.
② s에서 시간을 추출하기 위해 s를 3600으로 나눈 몫을 h에 저장한다.
③ 시간을 구하고 남은 나머지 초단위 값을 구하기 위해 s를 3600으로 나눈 나머지를 다시 s에 저장한다.
④ s를 다시 60으로 나눈 몫을 분단위 값에 해당하는 변수 m에 저장한다.
⑤ 분단위 값을 구하고 남은 초단위 값을 구하기 위해 s를 다시 60으로 나눈 나머지 값을 구해 s에 저장한다.
⑥ 시간단위, 분단위, 초단위 값이 저장되어 있는 h,m,s를 출력한다.</td></tr>
<tr><th>파이썬 코드</th></tr>
<tr><td><pre>s = int(input("초단위 시간 입력: "))
h = s // 3600
s = s % 3600
m = s // 60
s = s % 60
print(f'{h}시간 {m}분 {s}초')</pre></td></tr>
<tr><th>실행 결과</th></tr>
<tr><td><pre>초단위 시간 입력: 10000
2시간 46분 40초</pre></td></tr>
</table>

선택구조 알고리즘

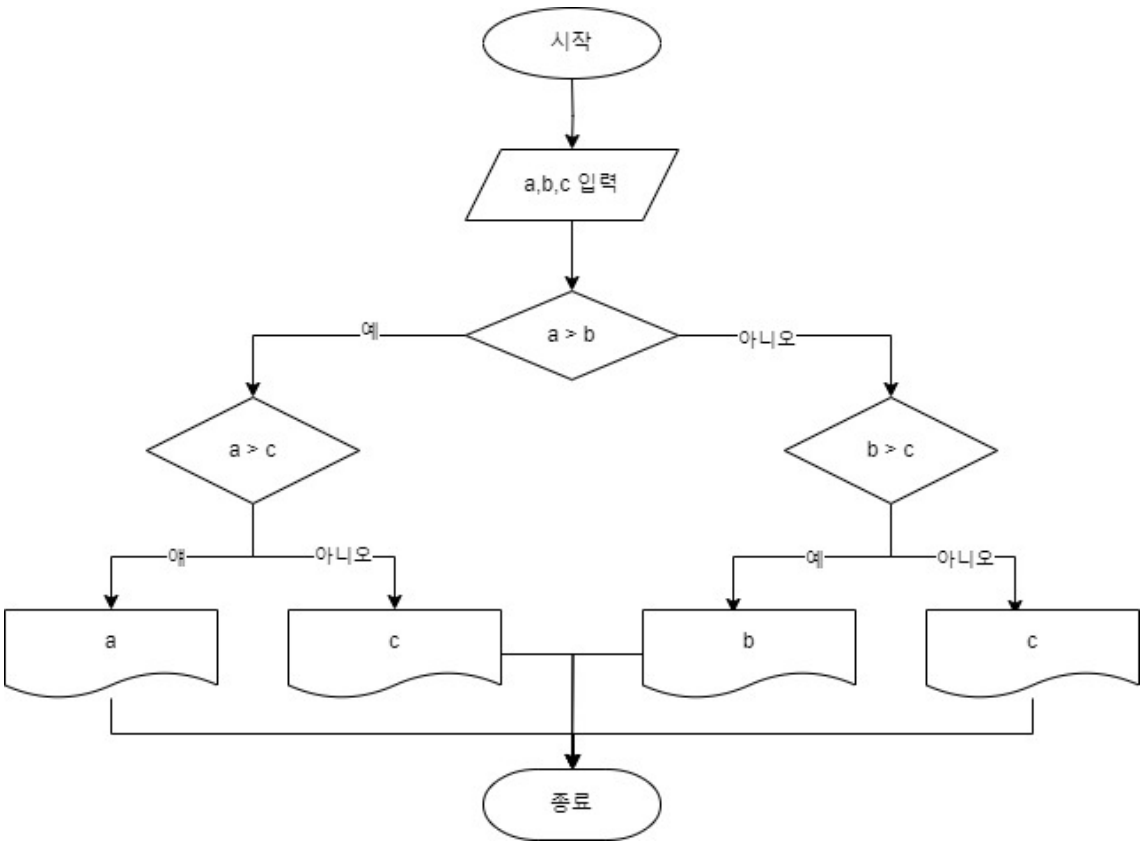
STEP 03

이 절에서는 3 가지 기본 제어구조 중에서 선택구조 알고리즘을 지금까지 배운 파이썬 문법들을 이용하여 프로그램으로 구현하는 방법을 연습해보자.

❑ 예제 1 : 3 개의 수 중 가장 큰 수 찾기

3 개의 정수 a, b, c 를 입력받아 가장 큰 수를 찾아 출력하는 알고리즘을 순서도로 표현하고, 이를 파이썬 프로그램으로 구현해보자.

순서도



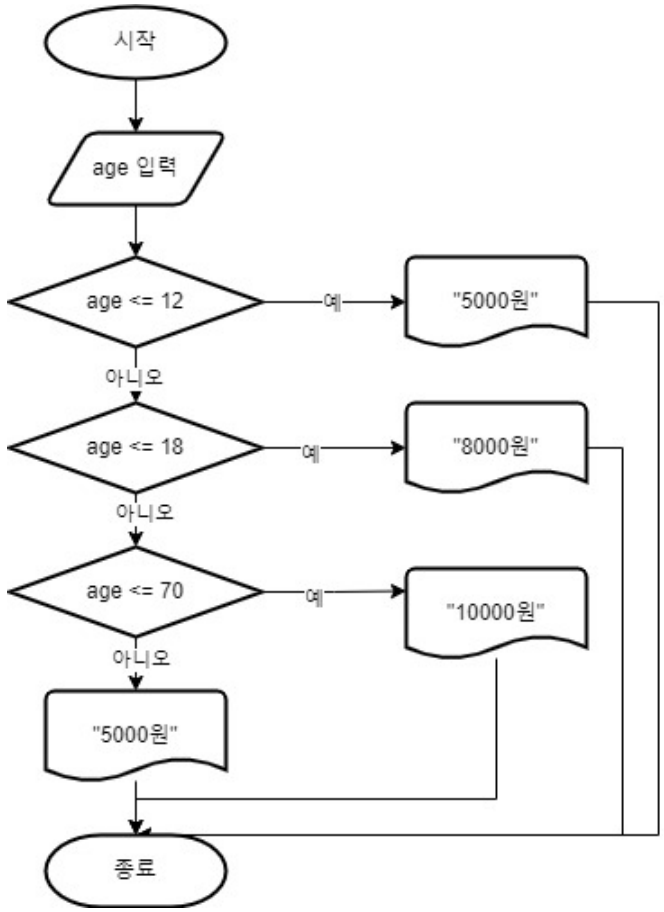
<table>
<tr><th>의사 코드</th><th>파이썬 코드</th></tr>
<tr><td rowspan="2">(1) 사용자로부터 3개의 정수를 입력받아 a,b,c에 저장한다.
(2) a, b의 대소를 비교한다.
(2-1) a가 큰 경우 a와 c의 대소를 비교한다.
(2-1-1) a가 큰 경우 a를 출력한다.
(2-1-2) b가 큰 경우 b를 출력한다.
(2-2) b가 큰 경우 b와 c의 대소를 비교한다.
(2-2-1) b가 큰 경우 b를 출력한다.
(2-2-2) c가 큰 경우 c를 출력한다.</td><td><pre>a = int(input("a 입력 :"))
b = int(input("b 입력 :"))
c = int(input("c 입력 :"))
if a > b :
    if a > c :
        print("최대값 :", a)
    else :
        print("최대값 :", c)
else :
    if b > c :
        print("최대값 :", b)
    else :
        print("최대값 :", c)</pre></td></tr>
<tr><td>실행 결과
<pre>a 입력 :6
b 입력 :3
c 입력 :5
최대값 : 6</pre></td></tr>
</table>

❑ 예제 2 : 놀이 공원 입장료 계산하기

나이에 따른 공원 입장료를 계산하는 문제를 해결하기 위한 알고리즘을 순서도로 표현하고, 이를 파이썬 프로그램으로 구현해보자. 나이에 따른 입장료 요금 체계는 아래 표와 같다.

나이	입장료
어린이(12세 이하)	5,000
청소년(18세 이하)	8,000
대인(18세~70세)	10,000
경로 우대(70세 초과)	5,000

나이에 따른 공원 입장료를 계산하는 문제는 다중 선택구조에 해당하며, 아래와 같이 연속적인 여러개의 선택구조로 이루어진 순서도로 표현할 수 있다. 이와 같은 경우, 파이썬 코드로 구현할 때, if/else 문을 중첩하여 쓰는 것 보다, if/elif/else 문을 사용하는 것이 편리하다.

순서도	의사 코드
	① 사용자로부터 나이를 입력받아 age에 저장한다. ② age가 12 이하이면, “5000원” 출력하고 종료 ③ age가 12 초과이고 18 이하이면, “8000원” 출력하고 종료 ④ age가 18 초과이고 70 이하이면, “10000원” 출력하고 종료 ⑤ age가 70 초과이며, “5000원” 출력하고 종료
파이썬 코드	**실행 결과**

`age = int(input("나이 입력 : "))`
`if age <= 12 :`
`  print("입장료는 5000원입니다.")`
`elif age <= 18 :`
`  print("입장료는 8000원입니다.")`
`elif age <= 70 :`
`  print("입장료는 10000원입니다.")`
`else :`
`  print("입장료는 5000원입니다.")`
	나이 입력 : 25
	입장료는 10000원입니다.

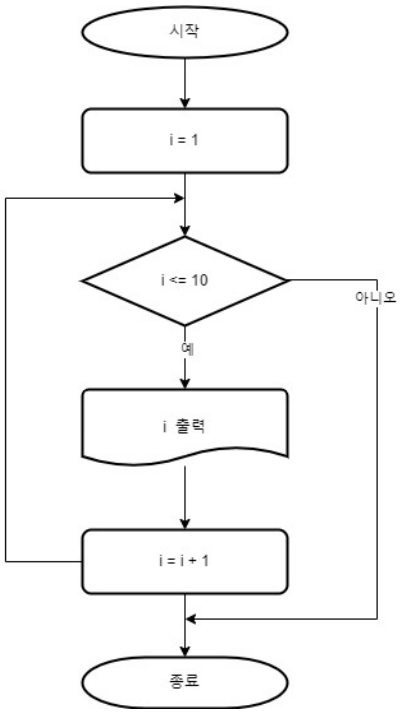
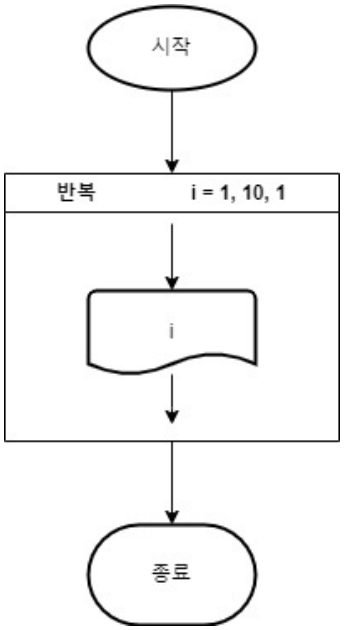
반복구조 알고리즘

STEP 04

이 절에서는 3 가지 기본 제어구조 중에서 반복구조 알고리즘을 지금까지 배운 파이썬 문법들을 이용하여 프로그램으로 구현하는 방법을 연습해보자.

❑ 예제 1 : 1 에서 10 까지 출력하기

1 에서 10 까지 정수를 출력하는 알고리즘을 순서도로 표현하고, 이를 파이썬 프로그램으로 구현해보자. 아래 그림은 문제해결을 위한 알고리즘을 2 가지 방법으로 순서도로 표현한 것이다.

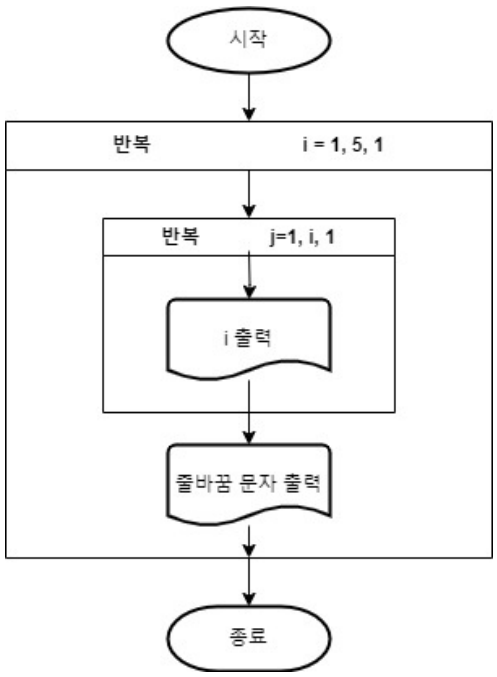
순서도1	순서도2
	

알고리즘1 설명	의사 코드
(1) i 값을 1로 초기화 (2) i 값이 10 이하인가? (2-1) Yes 인 경우 : (2-1-1) i 값 출력 (2-1-2) i 값을 1 증가 시킴 (2-1-3) (2)번으로 이동 (2-2) No 인 경우 : 종료	(1) i를 1에서 10까지 1씩 증가시키면서 다음을 반복 수행 (1-1) i 값 출력
파이썬 코드	**파이썬 코드 / 실행 결과**

`i =1`
`while i <= 10 :`
`  print(i, end=" ")`
`  i =i + 1`
[실행 결과]
1 2 3 4 5 6 7 8 9 10
	`for i in range(1, 11) :`
	`  print(i, end=" ")`
	[실행 결과]
	1 2 3 4 5 6 7 8 9 10

❑ 예제 2 : 직각 삼각형 모양으로 수 출력하기

1~5 까지의 숫자를 직각 삼각형 모양으로 출력하기 위한 알고리즘을 순서도로 표현하고, 파이썬 코드로 구현해보자.

순서도	의사 코드
	(1) i 값을 1에서 5까지 1씩 증가시키면서 반복 (1-1) i 값을 1에서 i까지 1씩 증가시키면서 반복 (1-1-1) i 값 출력 (1-2) 줄바꿈 문자 출력
	파이썬 코드

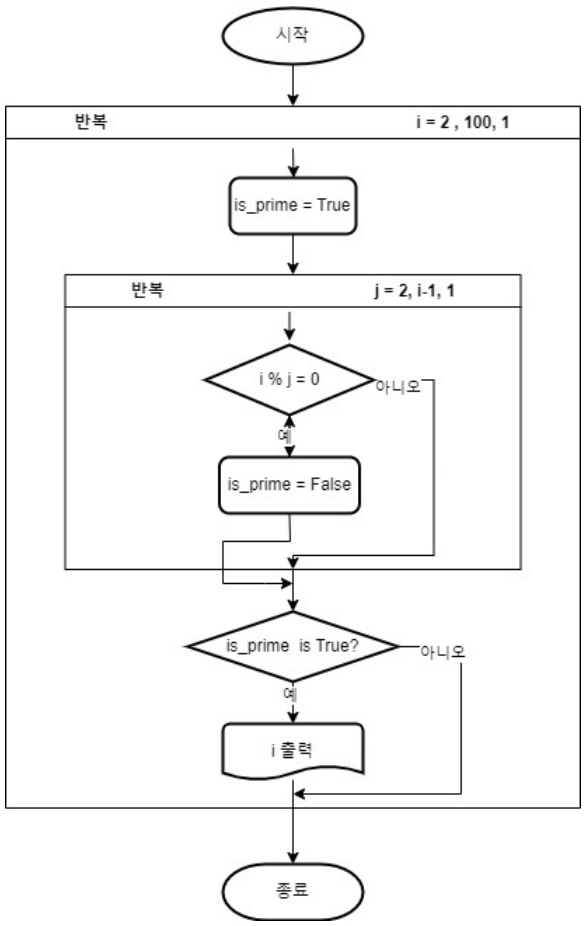
	`for i in range(1, 6) :`
	`  for j in range(1, i+1) :`
	`  print(j, end=" ")`
	`  print()`

	실행 결과
	1 1 2 1 2 3 1 2 3 4 1 2 3 4 5

❑ 예제 3 : 2 에서 100 까지의 소수 구하기

2~100 까지의 숫자 중 소수(prime number)를 구하여 출력하기 위한 알고리즘을 순서도로 표현하고, 파이썬 코드로도 구현해보자.

순서도



의사 코드

```
i 값을 2에서 100까지 1씩 증가시키면서 반복
  (1) j 값을 2에서 I-1까지 1씩 증가시키면서 반복
      (1-1) i는 소수라 가정(is_prime=True)
      (1-2) i가 j로 나누어 떨어지는가?
          (Yes) i는 소수가 아님
                is_prime=False
                j 반복 루프를 빠져나감
  (2) is_prime은 True 인가?
      (Yes) i는 소수이므로 i 값 출력
```

파이썬 코드

```
for i in range(2, 101) :
    is_prime = True
    for j in range(2, i) :
        if i % j == 0 :
            is_prime = False
            break
    if is_prime :
        print(i, end=" ")
```

실행 결과

```
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47
53 59 61 67 71 73 79 83 89 97
```

먼저 2~100 까지의 숫자 중 소수를 구하기 위해, for 문을 이용하여 2~100 정수에 대해서 매 반복 시마다 변수 i 에 1 씩 증가하는 정수를 대입하면서 i 가 소수인지를 검사하는 코드를 수행한다. i 가 소수인지를 알려면, i 보다 작은 정수들에 대해서 반복적으로 i 가 해당 정수로 나누어 떨어지는지를 검사하면 된다. 이를 위해서 다시 2~i-1 범위 내의 정수 j 에 대해서 i % j = 0 인지를 검사하는 반복구조를 만들면 된다. 검사의 효율성을 위해 먼저 i 가 소수라고 가정을 해놓고(is_prime = True), j 값을 1 씩 증가시키면서 반복을 돌면서 i 가 j 로 나누어 떨어지면 j 가 i 의 약수가 되므로, is_prime = False 로 바꾸고 j 루프를 빠져나온다. j 루프 실행이 완료되거나 중간에 빠져나오면, 그때의 is_prime 값을 검사해서 여전히 True 로 남아있으면, i 는 소수이므로, i 값을 출력하면 된다.