



# 2023 A.C.T. 주관 Hackathon Python 기초 교육

2023 A.C.T. 1학년 일동



# 기초 교육 진행 목차

## 1. 설치하기

- Python / VS code / Github 설치

## 2. 자료형

- 

## 3. 제어문

- 

## 4. 입출력

- 

## 5. 날개달기

- 

✓ Do Hyeon

1. Python 3.11 설치
2. Visual Studio Code 및. 한국어. Pylance. Code Runner 확장 설치
3. Github 개념 및 Git 설치(Windows 기준)
4. Python 기본 및 자료형(변수. 숫자형. 문자형. Bool)
5. Python 자료구조(List. Tuple. Dictionary. Set)
6. Python 제어문(if. while. for)
7. Python 함수
8. Python 파일 접근 및 사용자 입출력
9. Python Class
10. Python 모듈. 라이브러리. 패키지. 내장 함수
11. Python 예외 처리

이 문장은 제작 시 참고용. 발표시 지을 예정



# 1장

## - 설치하기 -

천혜성

# [python 설치하기]



**PYTHON**



1. 검색창에 python 검색하기
2. welcome to python.org 클릭

Google 1

python

X | ■ | 🔍 | ⚡ | 1

Tutorial | 이미지 | 연산자 | 옵션 | 차이 | 강좌 | Range | 동영상 | 1

검색결과 약 1,060,000,000개 (0.30초)

2 python.org  
https://www.python.org

Welcome to Python.org

The official home of the Python Programming Language.

python.org 검색

Downloads

Python 3.11.4 - Python 3.10.12 - Python 3.11.1 - Python 3.11.3

파이썬 자습서

1. 입맛 돋우기 - 5. 자료 구조 - 9. 클래스 - 6. 모듈 - ...

Documentation

Beginner's Guide - Python Version - Python Periodicals - ...

Python 3.11.4

Python 3.11.4 is the newest major release of the Python ...

### 3. Downloads → windows 클릭

The screenshot shows the Python.org Downloads page. The 'Downloads' tab is highlighted with a red circle. On the left, there's a sidebar with links like 'Download Python', 'Looking for Python', 'Linux/UNIX, macOS', 'Want to help test dev?', and 'Docker images'. Below that is a list of platforms: 'All releases', 'Source code', 'Windows' (which is also circled in red), 'macOS', 'Other Platforms', 'License', and 'Alternative Implementations'. To the right, there's a section titled 'Download for Windows' featuring a large image of a person with a yellow and white striped parachute. A button for 'Python 3.12.0' is visible. Below it, a note says 'Note that Python 3.9+ cannot be used on Windows 7 or earlier.' and 'Not the OS you are looking for? Python can be used on many operating systems and environments. View the full list of downloads.'

Active Python Releases

For more information visit the Python Developer's Guide.

Python version	Maintenance status	First released	End of support	Release schedule
3.13	prerelease	2024-10-01 (planned)	2029-10	PEP 719
3.12	bugfix	2023-10-02	2028-10	PEP 693
3.11	bugfix	2022-10-24	2027-10	PEP 664
3.10	security	2021-10-04	2026-10	PEP 619
3.9	security	2020-10-05	2025-10	PEP 596
3.8	security	2019-10-14	2024-10	PEP 569

<https://www.python.org/downloads/windows/>

## 4. 3.11.6 → 'Download Windows installer (64bit)' 클릭

Python » Downloads » Windows

100% ← + 초기화

### Python Releases for Windows

- Latest Python 3 Release - Python 3.12.0

#### Stable Releases

- Python 3.12.0 - Oct. 2, 2023

Note that Python 3.12.0 cannot be used on Windows 7 or earlier.

- Download Windows embeddable package (32-bit)
- Download Windows embeddable package (64-bit)
- Download Windows embeddable package (ARM64)
- Download Windows installer (32-bit)
- Download Windows installer (64-bit)
- Download Windows installer (ARM64)

- Python 3.11.6 - Oct. 2, 2023

Note that Python 3.11.6 cannot be used on Windows 7 or earlier.

- Download Windows embeddable package (32-bit)
- Download Windows embeddable package (64-bit)
- Download Windows embeddable package (ARM64)
- Download Windows installer (32-bit)
- Download Windows installer (64-bit)**
- Download Windows installer (ARM64)

#### Pre-releases

- Python 3.13.0a1 - Oct. 13, 2023

- Download Windows embeddable package (32-bit)
- Download Windows embeddable package (64-bit)
- Download Windows embeddable package (ARM64)
- Download Windows installer (32-bit)
- Download Windows installer (64-bit)
- Download Windows installer (ARM64)

- Python 3.12.0rc3 - Sept. 19, 2023

- Download Windows embeddable package (32-bit)
- Download Windows embeddable package (64-bit)
- Download Windows embeddable package (ARM64)
- Download Windows installer (32-bit)
- Download Windows installer (64-bit)
- Download Windows installer (ARM64)

- Python 3.12.0rc2 - Sept. 6, 2023

- Download Windows embeddable package (32-bit)
- Download Windows embeddable package (64-bit)

## 5. 우측 상단 클릭

The screenshot shows a web browser displaying the Python Releases for Windows page at [python.org/downloads/windows/](https://python.org/downloads/windows/). The page lists stable releases (Python 3.12.0, Python 3.11.6, Python 3.11.5) and pre-releases (Python 3.13.0a1, Python 3.12.0rc3, Python 3.12.0rc2). Each release section provides links for Windows embeddable packages and installers in 32-bit, 64-bit, and ARM64 architectures. A 'Recent Downloads' window is overlaid on the page, showing a single download entry: 'python-3.11.6-amd64.exe 24.6MB • 완료'. The browser's top right corner features a download icon, which is circled in red. A larger oval also circles the 'Recent Downloads' window itself.

python.org/downloads/windows/

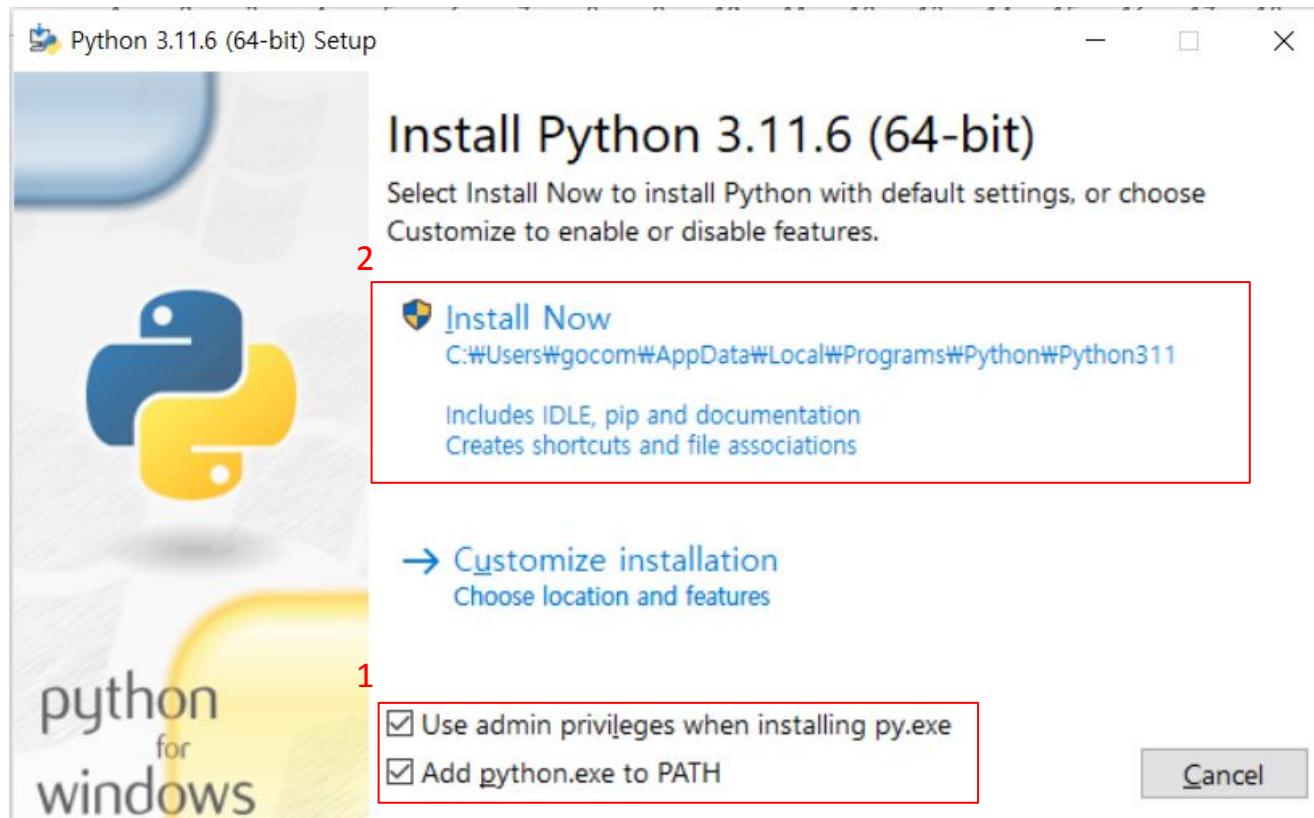
### Stable Releases

- Latest Python 3 Release - Python 3.12.0
  - Python 3.12.0 - Oct. 2, 2023
    - Note that Python 3.12.0 cannot be used on Windows 7 or earlier.
    - Download Windows embeddable package (32-bit)
    - Download Windows embeddable package (64-bit)
    - Download Windows embeddable package (ARM64)
    - Download Windows installer (32-bit)
    - Download Windows installer (64-bit)
    - Download Windows installer (ARM64)
  - Python 3.11.6 - Oct. 2, 2023
    - Note that Python 3.11.6 cannot be used on Windows 7 or earlier.
    - Download Windows embeddable package (32-bit)
    - Download Windows embeddable package (64-bit)
    - Download Windows embeddable package (ARM64)
    - Download Windows installer (32-bit)
    - Download Windows installer (64-bit)
    - Download Windows installer (ARM64)
  - Python 3.11.5 - Aug. 24, 2023
    - Note that Python 3.11.5 cannot be used on Windows 7 or earlier.
    - Download Windows embeddable package (32-bit)

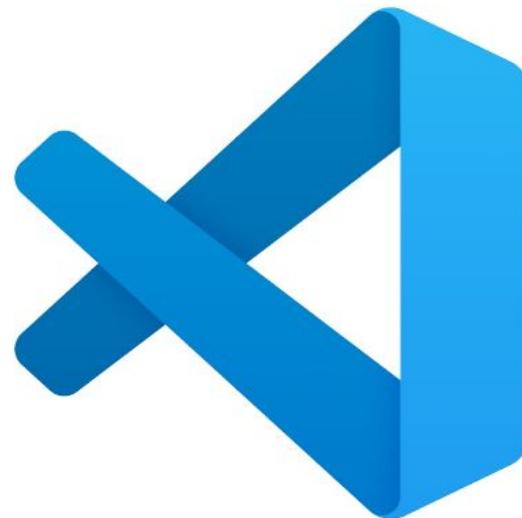
### Pre-releases

- Python 3.13.0a1 - Oct. 13, 2023
  - Download Windows embeddable package (32-bit)
  - Download Windows embeddable package (64-bit)
  - Download Windows embeddable package (ARM64)
  - Download Windows installer (32-bit)
  - Download Windows installer (64-bit)
  - Download Windows installer (ARM64)
- Python 3.12.0rc3 - Sept. 19, 2023
  - Download Windows embeddable package (32-bit)
  - Download Windows embeddable package (64-bit)
  - Download Windows embeddable package (ARM64)
  - Download Windows installer (32-bit)
  - Download Windows installer (64-bit)
  - Download Windows installer (ARM64)
- Python 3.12.0rc2 - Sept. 6, 2023
  - Download Windows embeddable package (32-bit)
  - Download Windows embeddable package (64-bit)
  - Download Windows embeddable package (ARM64)
  - Download Windows installer (32-bit)
  - Download Windows installer (64-bit)

## 6. 밑에 체크박스 선택 후 Install Now 클릭



# [Visual Studio Code 설치하기]



Visual Studio Code

# 구글에서 “vs code” 검색 후 맨 상단 클릭

google.com/search?q=vs+code&oq=vs+code&gs\_lcrp=EgZjaHJvbWUqDAGAEUYOixAxAjABDIMCAAQRg7GLEDGIAEMgcjARAAGIAEMgcjAhAAGIAEMgcjAxAAGIAEMgcjBBAAGI... ☆ 🔍

Google vs code

전체 이미지 동영상 도서 뉴스 더보기 도구 세이프,

검색결과 약 10,730,000,000개 (0.32초)

**Visual Studio Code**  
https://code.visualstudio.com ·

**Visual Studio Code - Code Editing. Redefined**

Visual Studio Code is a code editor redefined and optimized for building and debugging modern web and cloud applications. Visual Studio Code is free and ...

**Download**  
Visual Studio Code is free and available on your favorite ...

**Docs**  
Using GCC with MinGW - Getting Started with Java - Overview - C++

**VS Code for the Web**  
Visual Studio Code for the Web provides a free, zero-install ...

**Updates**  
vscode.dev is now cross origin isolated. Cross origin isolation ...

[visualstudio.com](https://visualstudio.com) 검색결과 더보기 »

**나무위키**  
https://namu.wiki · Visual Studio Code ·

**Visual Studio Code**

비주얼 스튜디오 코드 <

비주얼 스튜디오 코드 또는 코드는 마이크로소프트가 마이크로소프트 윈도우, macOS, 리눅스용으로 개발한 소스 코드 편집기이다. 디버깅 지원과 Git 제어, 구문 강조 기능, SSH 접속 등이 포함되어 있으며, 사용자가 편집기의 테마와 단축키, 설정 등을 수정할 수 있다.  
위키백과

프로그래밍 언어: 자바스크립트, C, C#, CSS, 타입스크립트

개발: 마이크로소프트

플랫폼: IA-32, x86-64, AArch64

최초 출시일: 2015년 4월 29일

# 우측 상단의 'Download' 클릭

The screenshot shows the official Visual Studio Code website. At the top, there's a navigation bar with links for Visual Studio Code, Docs, Updates, Blog, API, Extensions, FAQ, and Learn. To the right of the navigation is a search bar labeled "Search Docs" and a prominent blue "Download" button, which is highlighted with a red border. Below the navigation, a message states "Version 1.83 is now available! Read about the new features and fixes from September." The main content area features a large heading "Code editing. Redefined." and a subtext "Free. Built on open source. Runs everywhere." Below this are two download buttons: "Download for Windows Stable Build" and "Web, Insiders edition, or other platforms". A note below the buttons says "By using VS Code, you agree to its license and privacy statement." On the right side of the page, there's a screenshot of the Visual Studio Code interface showing the code editor with some JavaScript code, the Extensions Marketplace, and the terminal. The code editor window has tabs for "serviceWorker.js", "index.js", and "serviceWorker.js". The terminal at the bottom shows the command "create-react-app" being run.

# windows 사용자는 'windows' 클릭 / Mac 사용자는 'Mac' 클릭

The screenshot shows the official Visual Studio Code website. At the top, there's a navigation bar with links for Visual Studio Code, Docs, Updates, Blog, API, Extensions, FAQ, and Learn. To the right of the navigation is a search bar labeled "Search Docs" and a blue "Download" button. A banner at the top of the main content area says "Version 1.83 is now available! Read about the new features and fixes from September." Below the banner, the title "Download Visual Studio Code" is centered, followed by the subtitle "Free and built on open source. Integrated Git, debugging and extensions." Three download options are shown: a Windows icon with a red box around the "Windows" link, a Linux Tux icon with ".deb" and ".rpm" links, and an Apple icon with a red box around the "Mac" link.

## Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.



↓ Windows

Windows 10, 11



↓ .deb

Debian, Ubuntu

↓ .rpm

Red Hat, Fedora, SUSE



↓ Mac

macOS 10.15+

User Installer [x64](#) [x86](#) [Arm64](#)

System Installer [x64](#) [x86](#) [Arm64](#)

.zip [x64](#) [x86](#) [Arm64](#)

CLI [x64](#) [x86](#) [Arm64](#)

.deb [x64](#) [Arm32](#) [Arm64](#)

.rpm [x64](#) [Arm32](#) [Arm64](#)

.tar.gz [x64](#) [Arm32](#) [Arm64](#)

Snap [Snap Store](#)

CLI [x64](#) [Arm32](#) [Arm64](#)

.zip [Intel chip](#) [Apple silicon](#) [Universal](#)

CLI [Intel chip](#) [Apple silicon](#)

# 우측 상단 클릭

The screenshot shows a browser window displaying the Visual Studio Code documentation at [code.visualstudio.com/docs/?dv=win64user](https://code.visualstudio.com/docs/?dv=win64user). The page features a dark header with the Visual Studio Code logo and navigation links for Docs, Updates, Blog, API, Extensions, FAQ, and Learn. A banner at the top announces "Version 1.84 is now available! Read about the new features and fixes from October." On the left, a sidebar lists various documentation categories such as Overview, Setup, Get Started, User Guide, Source Control, Terminal, Languages, Node.js / JavaScript, TypeScript, Python, Java, C++, C#, Docker, Data Science, Azure, Remote, and Dev Containers. The main content area is titled "Getting Started" and describes VS Code as a lightweight source code editor. Below this, a section titled "Visual Studio Code in Action" shows a code editor with some JavaScript code and a callout highlighting code completion suggestions for "get". A floating sidebar on the right is titled "최근 다운로드 기록" (Recent Download History) and lists "VSCodeUserSetup-x64-1.84.0.exe" (90.2MB · 완료). The "전체 다운로드 기록" (Full Download History) and "GETTING STARTED" sections are also visible.

Documentation for Visual Studio Code

code.visualstudio.com/docs/?dv=win64user

Visual Studio Code Docs Updates Blog API Extensions FAQ Learn

Version 1.84 is now available! Read about the new features and fixes from October.

OVERVIEW

SETUP

GET STARTED

USER GUIDE

SOURCE CONTROL

TERMINAL

LANGUAGES

NODEJS /  
JAVASCRIPT

TYPESCRIPT

PYTHON

JAVA

C++

C#

DOCKER

DATA SCIENCE

AZURE

REMOTE

DEV CONTAINERS

Thanks for downloading VS Code for Windows!

Download not starting? Try this [direct download link](#).

Please take a few seconds and help us improve ... [click to take survey](#).

## Getting Started

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages and runtimes (such as C++, C#, Java, Python, PHP, Go, .NET). Begin your journey with VS Code with these [introductory videos](#).

## Visual Studio Code in Action

```
4 var server = express();
5 server.use(bodyParser.json());
6
7 server.g
8   ↗ get (property) Application.get: ((name: string)... ⓘ
9   ↗ getMaxListeners
10  ↗ arguments
11  ↗ engine
```

최근 다운로드 기록

VSCodeUserSetup-x64-1.84.0.exe  
90.2MB · 완료

전체 다운로드 기록

GETTING STARTED

VS Code in Action

Top Extensions

First Steps

Keyboard Shortcuts

Downloads

Privacy

Subscribe

Ask questions

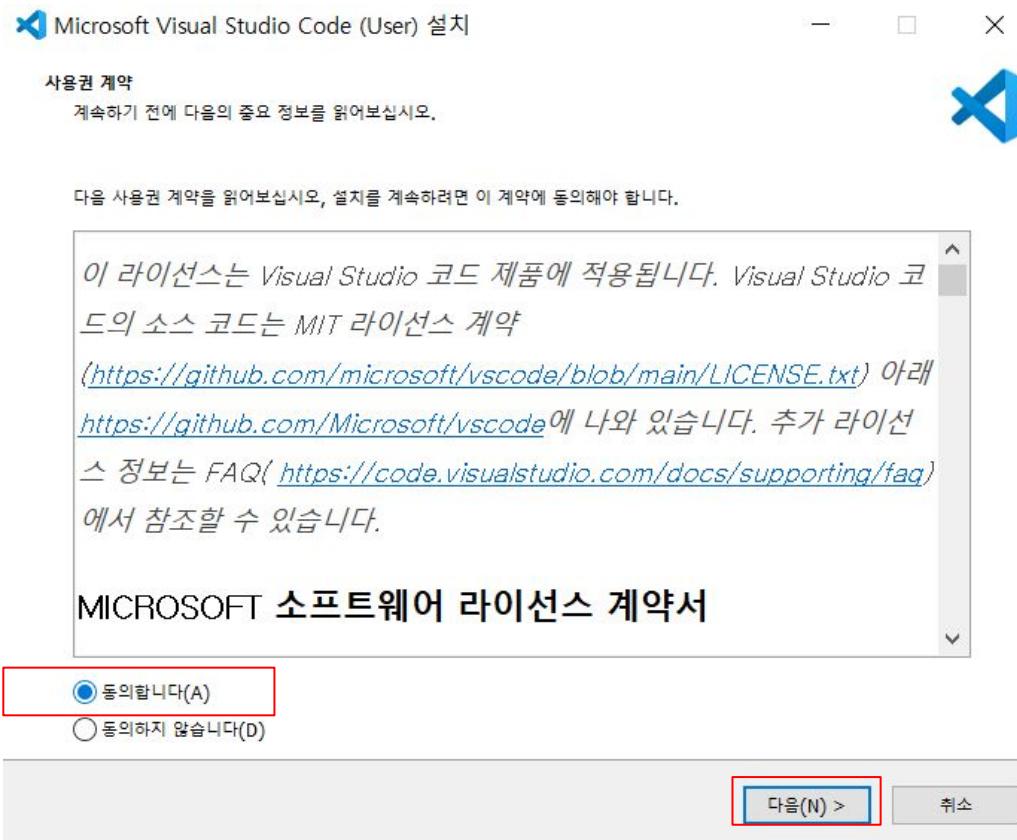
Follow @code

Request features

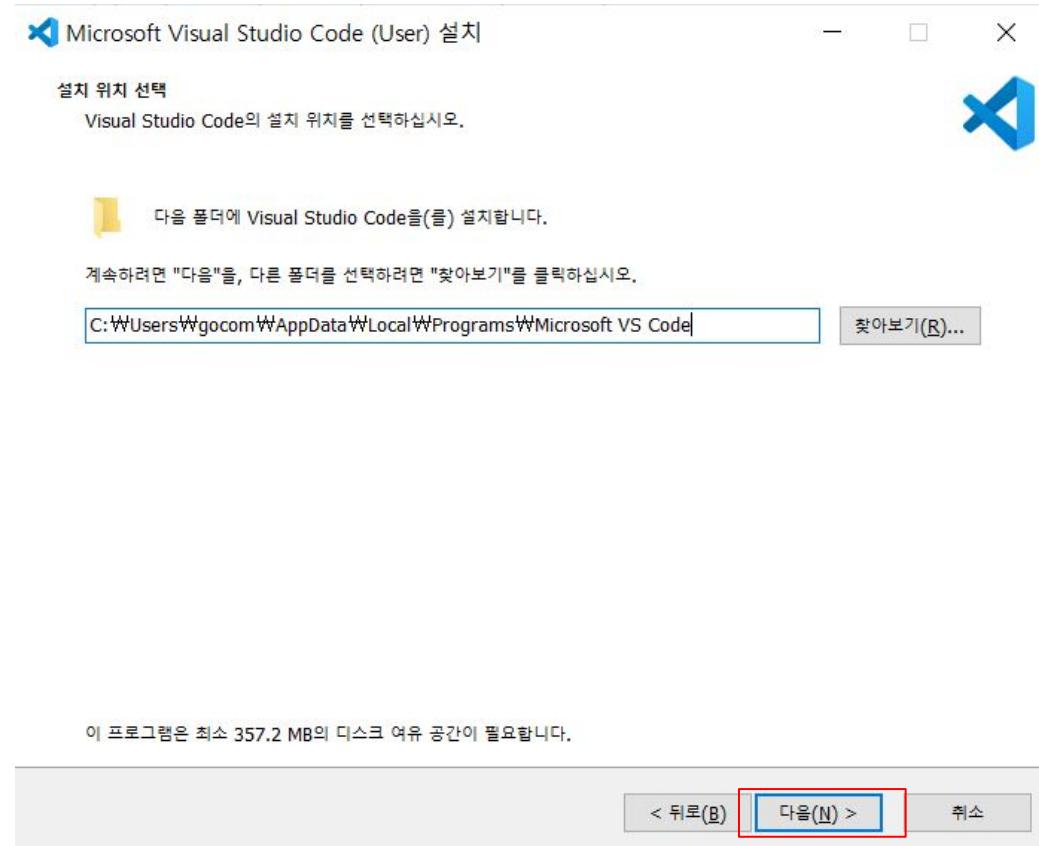
Report issues

Watch videos

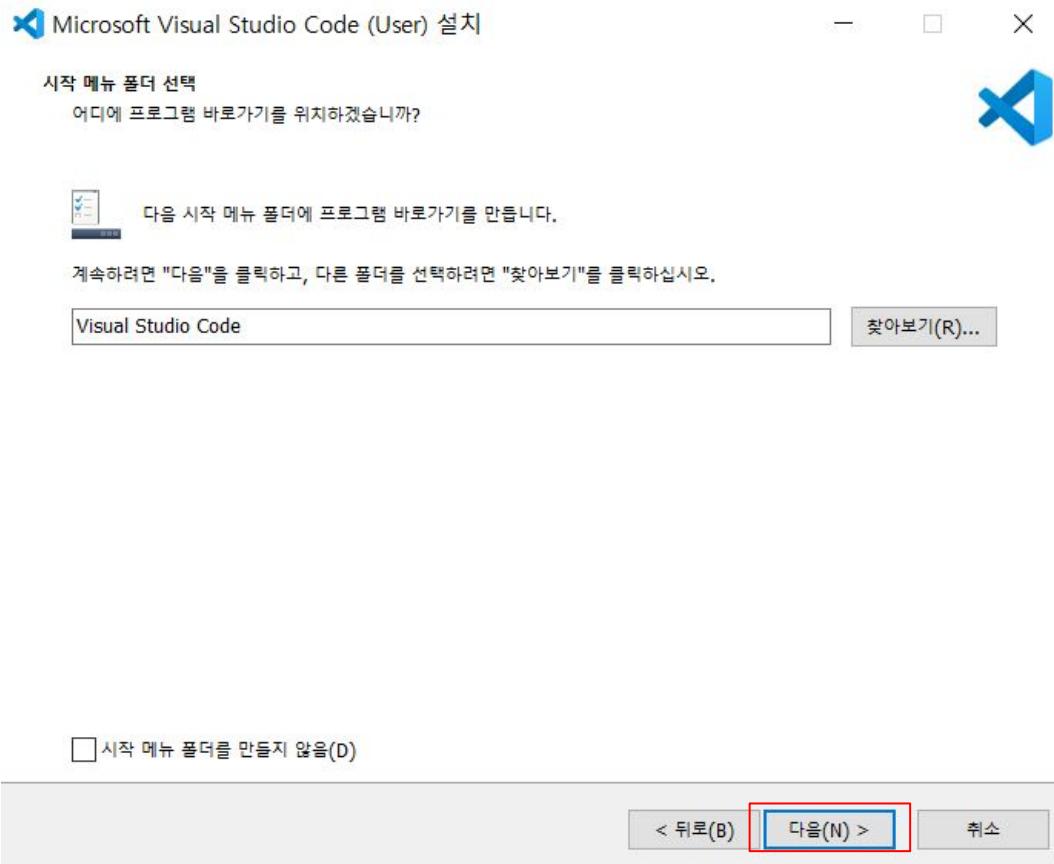
## '동의합니다' 선택 → '다음'



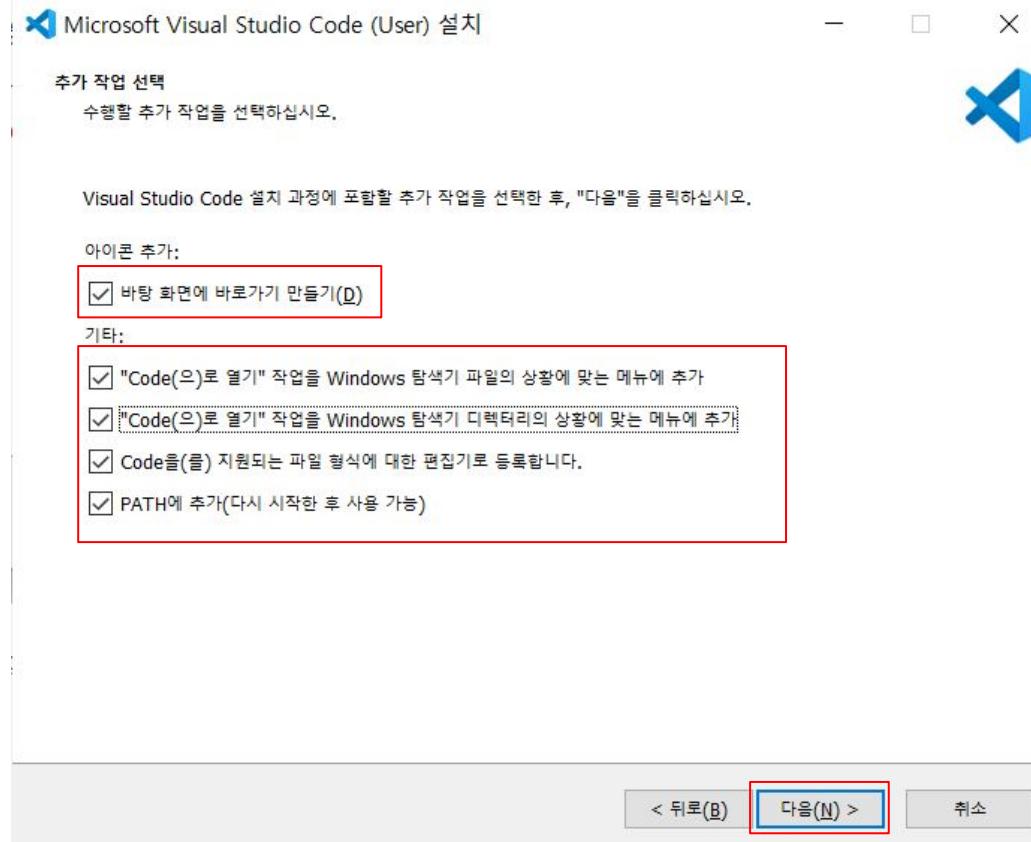
# '다음' 클릭



# '다음' 클릭



# 체크박스 체크 후 '다음' 클릭



# 설치 클릭



Microsoft Visual Studio Code (User) 설치



## 설치 준비 완료

귀하의 컴퓨터에 Visual Studio Code(를) 설치할 준비가 되었습니다.



설치를 계속하려면 "설치"를, 설정을 변경하거나 검토하려면 "뒤로"를 클릭하십시오.

### 설치 위치:

C:\Users\gocom\AppData\Local\Programs\Microsoft VS Code

### 시작 메뉴 폴더:

Visual Studio Code

### 추가 작업:

#### 아이콘 추가:

바탕 화면에 바로가기 만들기(D)

#### 기타:

"Code(으)로 열기" 작업을 Windows 탐색기 파일의 상황에 맞는 메뉴에 추가

"Code(으)로 열기" 작업을 Windows 탐색기 디렉터리의 상황에 맞는 메뉴에 추가

Code(를) 지원되는 파일 형식에 대한 편집기로 등록합니다.

PATH에 추가(다시 시작한 후 사용 가능)

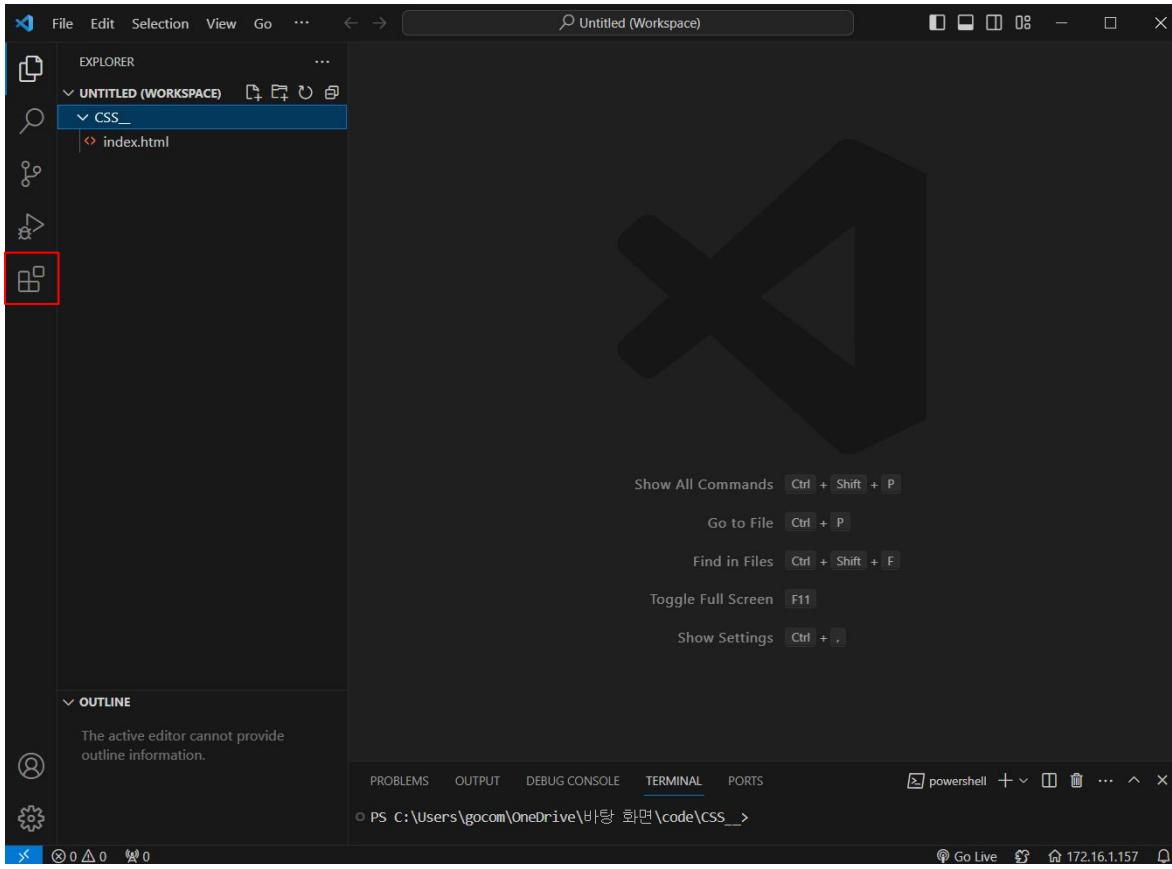
< 뒤로(B)

설치(I)

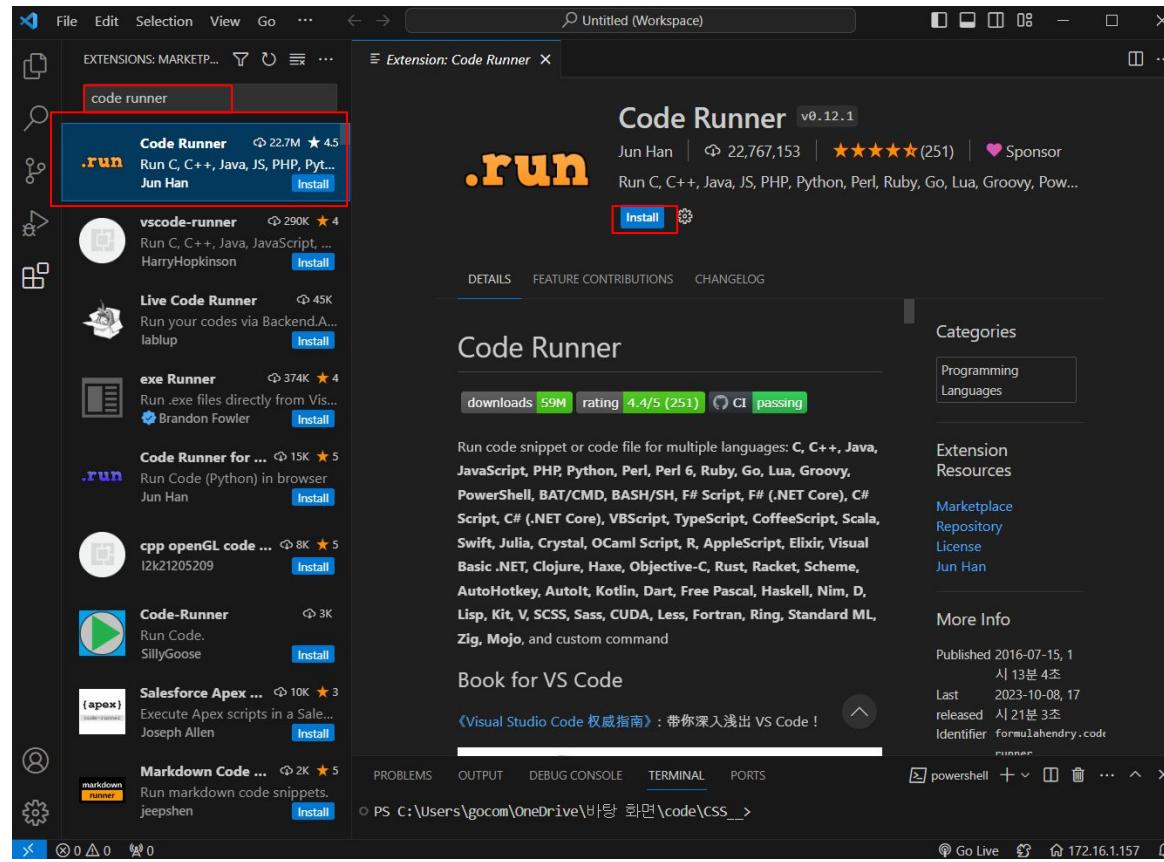
취소

# 환경 설정하기

# Visual Studio Code 열고 나서 잠자 클릭



# 검색창에 'code runner' 검색 후 최상단 클릭 -> install 클릭



# Github와 git 알아보기



git



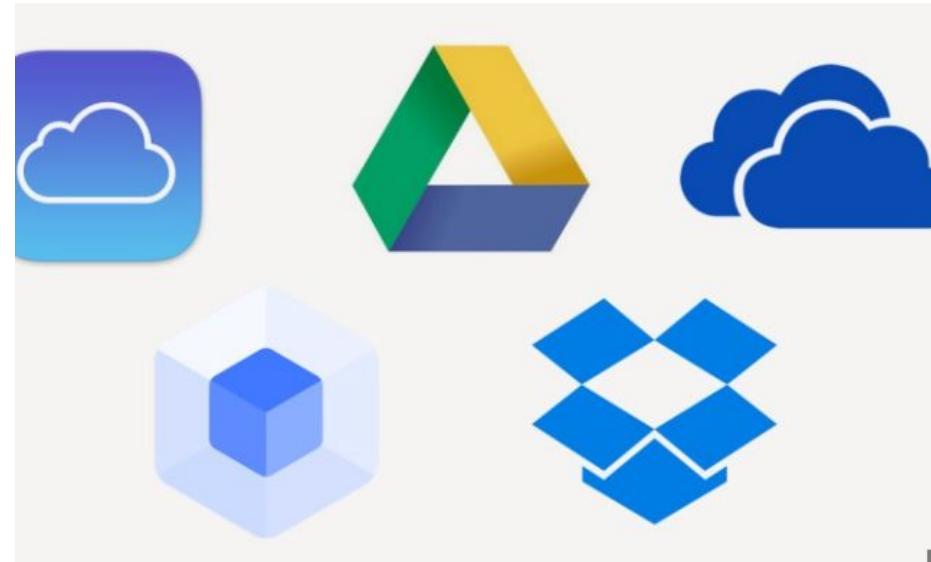
# 버전 관리 도구 git



# git

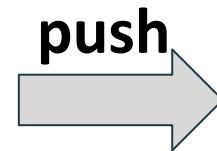
- 포스터 시안
- 포스터 시안
- 최종완료1 - 복사본 (2)
- 최종완료
- 진짜최종완료
- 진짜진짜최종완료
- 진짜진짜진짜최종완료
- 중간저장3
- 중간저장2
- 중간저장
- 완료
- 마지막
- OLOLOLO
- 끝
- asdasdasda
- 123123123444444444444444444444
- 11111111111111111111111111111111111111...
- 123123123123

# 코드저장소 **github**





git





github



전체

뉴스

이미지

동영상

지도

더보기

도구

세이프서치

검색결과 약 708,000,000개 (0.25초)



GitHub

https://github.com



## GitHub: Let's build from here · GitHub

GitHub is where over 100 million developers shape the future of software, together. Contribute to the open source community, manage your Git repositories, ...

[github.com 검색결과](#)

### Login

GitHub is where people build software. More than 100 million ...

### Explore GitHub

Explore is your guide to finding your next project, catching up ...

### GitHub Desktop

Open your favorite editor or shell from the app, or jump back to ...

### Join GitHub

GitHub is where people build software. More than 100 million ...



나무위키

https://namu.wiki › GitHub



## GitHub - 깃허브

2023. 10. 26. — The complete developer platform to build, scale, and deliver secure software. 보

## 깃허브 (GitHub)

소프트웨어 개발자



깃허브는 루비 온 레일스로 작성된 분산 버전 관리 툴인 깃 저장소 호스팅을 지원하는 웹 서비스이다. 깃허브는 영리적인 서비스와 오픈소스를 위한 무상 서비스를 모두 제공한다. 2009년의 깃 사용자 조사에 따르면 깃허브는 가장 인기있는 깃 저장소 호스팅 서비스이다.  
위키백과

**창립자:** 톰 프레스턴 워너, 크리스 월스트래스, 스캇 샤콘, P. J. 하이엇

**창립:** 2008년, 미국 캘리포니아 샌프란시스코

**본사:** 미국 캘리포니아 샌프란시스코

**자회사:** npm, Inc.

**CEO:** 토마스 드케 (2021년 11월 15일–)

면책조항

프로필



Product ▾ Solutions ▾ Open Source ▾ Pricing

Search or jump to... /

Sign in [Sign up](#)



GitHub Universe: Dive in to AI, security, and DevEx >  
Get your tickets how to join us on Nov. 8-9.

# Let's build from here

The AI-powered developer platform to build, scale, and deliver  
secure software



Welcome to GitHub!

Let's begin the adventure

Enter your email\*

✓ somimif833@mainmile.com

Create a password\*

✓ ••••••••

Enter a username\*

✓ asdf1234asdfcnjwno

Would you like to receive product updates and announcements via  
email?

Type "y" for yes or "n" for no

✓ n

Verify your account



Create account



# Welcome to GitHub

We are glad you're here.

This will help us guide you to the tools that are best suited for your projects.

How many team members will be working with you?

<input type="radio"/> Just me	<input type="radio"/> 2-5	<input type="radio"/> 5-10
<input type="radio"/> 10-20	<input type="radio"/> 20-50	<input type="radio"/> 50+

Are you a student or teacher?

<input type="radio"/> N/A	<input type="radio"/> Student	<input type="radio"/> Teacher
---------------------------	-------------------------------	-------------------------------

[Continue](#)

[Skip personalization](#)



# Welcome to GitHub

We are glad you're here.

This will help us guide you to the tools that are best suited for your projects.

How many team members will be working with you?

Just me

2-5

5-10

10-20

20-50

50+

Are you a student or teacher?

N/A

Student

Teacher

**Continue**

# Real-world tools, engaged students.

GitHub gives teachers free access to industry-standard tools for training developers.

## Free

- Unlimited public/private repositories
- 2,000 CI/CD minutes/month  
Free for public repositories
- 500MB of Packages storage  
Free for public repositories
- 120 core-hours of Codespaces compute
- 15GB of Codespaces storage
- Community support

## Get additional teacher benefits

### GitHub Team

- Protect your branches  
Ensure that collaborators on your repository cannot make irreversible changes to branches.
- Draft pull requests
- Required reviewers
- 3,000 CI/CD minutes/month  
Free for public repositories
- 2GB of Packages storage  
Free for public repositories
- Web-based support

### GitHub Teacher Toolbox

- Free access to the industry's best developer tools  
Includes GitHub, Bitbucket, Jenkins, CircleCI, Docker, GitLab, GitHub Actions, Microsoft Azure, Heroku, MongoDB, Datadog, Travis, and Drift.

### GitHub Classroom

- Automate your course  
Track and manage assignments in your classroom, grade work automatically, and help students when they get stuck.

### GitHub Campus Advisors

- Join a community for teachers  
GitHub Campus Advisors mentor GitHub and GitHub, and champion the use of real-world tools at their school.

Continue for free

Apply for your GitHub teacher benefits

Dashboard

Create your first project  
Ready to start building? Create a repository for a new idea or bring over an existing repository to keep contributing to it.

[Create repository](#) [Import repository](#)

Recent activity  
When you take actions across GitHub, we'll provide links to that activity here.

Home

Updates to your homepage feed

We've combined the power of the Following feed with the For You feed so there's one place to discover content on GitHub. There's improved filtering so you can customize your feed exactly how you like it, and a shiny new visual design. 🎉

[Learn more](#)

Start writing code

Start a new repository

A repository contains all of your project's files, revision history, and collaborator discussion.

asdf1234asdFcnjwno /

Public Anyone on the internet can see this repository

Private You choose who can see and commit to this repository

[Create a new repository](#)

Introduce yourself with a profile README

Share information about yourself by creating a profile README, which appears at the top of your profile page.

asdf1234asdFcnjwno / README.md [Create](#)

1 - 🌟 Hi, I'm @asdf1234asdFcnjwno  
2 - 🌐 I'm interested in ...  
3 - 🚀 I'm currently learning ...  
4 - 🤝 I'm looking to collaborate on ...  
5 - 📧 How to reach me ...  
6 -

Use tools of the trade

Simplify your development workflow with a GUI

Install GitHub Desktop to visualize, commit, and push changes without ever touching the command line.

Get AI-based coding suggestions

Try GitHub Copilot free for 30 days, which suggests entire functions in real time, right from your editor.

Follow this exercise to try the GitHub flow

GitHub's "Hello World" tutorial teaches you essentials, where you create your own repository and learn GitHub's pull request workflow for creating and reviewing code.

[Try the GitHub flow](#)

Latest changes

4 hours ago Secret scanning expands detection to include non-provider patterns (beta)

11 hours ago Subversion brownouts

3 days ago Multi-account support on GitHub.com

4 days ago [Public Beta] Administrative enhancements to Enterprise Managed Users SCIM

[View changelog →](#)

Explore repositories

supabase / supabase

The open source Firebase alternative.

58.9k · TypeScript

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?  
[Import a repository.](#)

Required fields are marked with an asterisk (\*).

Owner \*



asdf1234asdfcnjwno ▾

Repository name \*

asdfasdf1234

asdfasdf1234 is available.

Great repository names are short and memorable. Need inspiration? How about [expert-computing-machine](#) ?

Description (optional)

 Public

Anyone on the internet can see this repository. You choose who can commit.

 Private

You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file

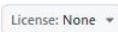
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore



Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license



A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

 You are creating a public repository in your personal account.

**Create repository**



#### Set up GitHub Copilot

Use GitHub's AI pair programmer to autocomplete suggestions as you code.

[Get started with GitHub Copilot](#)



#### Add collaborators to this repository

Search for people using their GitHub username or email address.

[Invite collaborators](#)

### Quick setup — if you've done this kind of thing before

 Set up in Desktop

or

 HTTPS  SSH

<https://github.com/asdf1234asdfcnjwno/asdfasdf1234.git>

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

### ...or create a new repository on the command line

```
echo "# asdfasdf1234" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/asdf1234asdfcnjwno/asdfasdf1234.git
git push -u origin main
```

### ...or push an existing repository from the command line

```
git remote add origin https://github.com/asdf1234asdfcnjwno/asdfasdf1234.git
git branch -M main
git push -u origin main
```

### ...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)



Drag additional files here to add them to your repository

Or choose your files

Elon\_musk.html x

Favoite.html x

John.F.Kennedy.html x

Steve\_jobs.html x



### Commit changes

Add files via upload

Add an optional extended description...

**Commit changes**

Cancel

asdf1234asdfcnjwno / asdfasdf1234

Type  to search | [+ +](#) | [\( \)](#) | [\( \)](#) | [\( \)](#) | [\( \)](#)

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

## asdfasdf1234 Public

[Pin](#) [Unwatch 0](#) [Fork 0](#) [Star 0](#)

[main](#) branch [1 branch](#) [0 tags](#)

[Go to file](#) [Add file](#) [Code](#)

### asdf1234asdfcnjwno Add files via upload

b32d116 now [1 commit](#)

<a href="#"></a> Elon_musk.html	Add files via upload	now
<a href="#"></a> Favoite.html	Add files via upload	now
<a href="#"></a> John.F.Kennedy.html	Add files via upload	now
<a href="#"></a> Steve_jobs.html	Add files via upload	now

### About

No description, website, or topics provided.

[Activity](#)

[0 stars](#)

[0 watching](#)

[0 forks](#)

### Releases

No releases published

[Create a new release](#)

### Packages

No packages published

[Publish your first package](#)



© 2023 GitHub, Inc.

[Terms](#)

[Privacy](#)

[Security](#)

[Status](#)

[Docs](#)

[Contact GitHub](#)

[Pricing](#)

[API](#)

[Training](#)

[Blog](#)

[About](#)

# Git 설치하기



git



git



전체

이미지

뉴스

도서

더보기

도구

세이프서치

검색결과 약 1,510,000,000개 (0.23초)



Git SCM

<https://git-scm.com> :

Git

Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with ...

## Downloads

Windows - macOS - Linux/Unix - GUI Clients - ...

## Download for Windows

Click here to download the latest (2.42.0) 32-bit version of Git for ...

## Pro Git 한국어 번역 - Book

Git 기초 - Git 설치 - 되돌리기 - Rebase 하기 - ...

## Download for macOS

There are several options for installing Git on macOS. Note ...

[git-scm.com](https://git-scm.com) 검색결과 더보기 »



GitHub

<https://github.com> :

## GitHub: Let's build from here · GitHub

GitHub is where over 100 million developers shape the future of software, together. Contribute to



git  
[More]

[이미지 더보기](#)

깃

소프트웨어



깃은 컴퓨터 파일의 변경사항을 추적하고 여러 명의 사용자들 간에 해당 파일들의 작업을 조율하기 위한 스냅샷 스트림 기반의 분산 버전 관리 시스템이다. 또는 이러한 명령어를 가리킨다. 위키백과

프로그래밍 언어: 파이썬, C, C++, 셀 스크립트, 펄, Tcl

개발자: 리누스 토르발스, Junio C Hamano

개발자: 주니오 하마노(Junio Hamano), 리누스 토르발스 등

라이선스: GNU 일반 공중 사용 허가서 v2

작성자 버전: 2421 / 2023년 11월 2일



# git

--distributed-is-the-new-centralized

Search entire site...

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**.



## About

The advantages of Git compared to other source control systems.



## Documentation

Command reference pages, Pro Git book content, videos and other material.



## Downloads

GUI clients and binary releases for all major platforms.



## Community

Get involved! Bug reporting, mailing list, chat, development and more.



**Pro Git** by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on [Amazon.com](#).



[Windows GUIs](#)



[Tarballs](#)



## About

## Documentation

## Downloads

GUI Clients  
Logos

## Community

The entire [Pro Git book](#)  
written by Scott Chacon and  
Ben Straub is available to [read online for free](#). Dead tree  
versions are available on  
[Amazon.com](#).

# Downloads



Older releases are available and the [Git source repository](#) is on GitHub.



## GUI Clients

Git comes with built-in GUI tools (`git-gui`, `gitk`), but there are several third-party tools for users looking for a platform-specific experience.

[View GUI Clients →](#)

## Logos

Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

[View Logos →](#)

## Git via Git

If you already have Git installed, you can get the latest development version via Git itself:

```
git clone https://github.com/git/git
```

You can also always browse the current contents of the git repository using the [web interface](#).



## About

## Documentation

## Downloads

GUI Clients

Logos

## Community

The entire [Pro Git book](#) written by Scott Chacon and Ben Straub is available to read [online for free](#). Dead tree versions are available on [Amazon.com](#).

# Download for Windows

[Click here to download](#) the latest (2.42.0) 64-bit version of **Git for Windows**. This is the most recent [maintained build](#). It was released **2 months ago**, on 2023-08-30.

## Other Git for Windows downloads

[Standalone Installer](#)

[32-bit Git for Windows Setup.](#)

[64-bit Git for Windows Setup.](#)

[Portable \("thumbdrive edition"\)](#)

[32-bit Git for Windows Portable.](#)

[64-bit Git for Windows Portable.](#)

## Using winget tool

Install [winget tool](#) if you don't already have it, then type this command in command prompt or Powershell.

```
winget install --id Git.Git -e --source winget
```

The current source code release is version 2.42.1. If you want the newer version, you can build it from [the source code](#).

## Now What?

Now that you have downloaded Git, it's time to start using it.



git --fast-version-control

## About

## Documentation

## Downloads

GUI Clients

Logos

## Community

The entire [Pro Git book](#) written by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

# Download for Windows

[Click here to download](#) the latest (2.42.0) 64-bit version of **Git for Windows**. This recent [maintained build](#). It was released **2 months ago**, on 2023-08-30.

## Other Git for Windows downloads

[Standalone Installer](#)

[32-bit Git for Windows Setup.](#)

[64-bit Git for Windows Setup.](#)

[Portable \("thumbdrive edition"\)](#)

[32-bit Git for Windows Portable.](#)

[64-bit Git for Windows Portable.](#)

## Using winget tool

Install [winget tool](#) if you don't already have it, then type this command in command prompt or Powershell.

```
winget install --id Git.Git -e --source winget
```

The current source code release is version 2.42.1. If you want the newer version, you can build it from the [source code](#).

## Now What?

Now that you have downloaded Git, it's time to start using it.

## 최근 다운로드 기록

 Git-2.42.0.2-64-bit.exe

58.4MB • 완료

 2023 A.C.T. 주관 Hackathon Python

기초 교육 (3).pptx

14.7MB • 2시간 전

 2023 A.C.T. 주관 Hackathon Python

기초 교육 (3).pptx

14.7MB • 3시간 전

 2023 A.C.T. 주관 Hackathon Python

기초 교육 (2).pptx

12.0MB • 3시간 전

 2023 A.C.T. 주관 Hackathon Python

기초 교육 (1).pptx

10.0MB • 12시간 전

 2023 A.C.T. 주관 Hackathon Python

기초 교육.pptx

10.0MB • 14시간 전

 HSC2015\_안티키테라\_연구결과서.pdf

전체 다운로드 기록



## Information

Please read the following important information before continuing.



When you are ready to continue with Setup, click Next.

## GNU General Public License

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.  
59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies  
of this license document, but changing it is not allowed.

### Preamble

The licenses for most software are designed to take away your  
freedom to share and change it. By contrast, the GNU General Public  
License is intended to guarantee your freedom to share and change

<https://gitforwindows.org/>

Only show new options

Install

Cancel



## Completing the Git Setup Wizard

Setup has finished installing Git on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.



Launch Git Bash

View Release Notes

Only show new options

Finish



# 2장

## -자료형-

팀장: 정창영 팀원: 차연우, 정민서



# 자료형

1. 숫자형
2. 문자열 자료형
3. 리스트 자료형
4. 튜플 자료형
5. 딕셔너리 자료형
6. 집합 자료형
7. 불 자료형
8. 자료형의 값을 저장하는 공간. 변수



## 1) 숫자형

숫자형(Number)이란 숫자 형태로 이루어진 자료형으로, 우리가 이미 잘 알고 있는 것이다. 우리가 흔히 사용하는 숫자들을 생각해 보자. 123과 같은 정수, 12.34와 같은 실수, 8진수나 16진수도 있다.

항목	파이썬 사용 예
정수	123, -345, 0
실수	123.45, -1234.5, 3.4e10
8진수	0o34, 0o25
16진수	0x2A, 0xFF



# 정수형

정수형(**integer**)이란 말 그대로 정수를 뜻하는 자료형을 말한다. 다음은 양의 정수와 음의 정수, 숫자 0을 변수 **a**에 대입하는 예이다

ex)    >>> a = 53

```
>>> a = -158
```

```
>>> a = 0
```

# 실수형

파이썬에서 실수형(floating-point)은 소수점이 포함된 숫자를 말한다. 다음은 실수를 변수 `a`에 대입하는 예이다. 일반적으로 볼 수 있는 실수형의 소수점 표현 방식이다.

ex)    `>>> a = 6.4`

`>>> a = -4.2`

다음은 ‘컴퓨터식 지수 표현 방식’으로, 파이썬에서는 `4.24e10` 또는 `4.24E10`처럼 표현한다(`e`와 `E` 둘 중 어느 것을 사용해도 된다).

ex)    `>>> a = 4.24E10`    #  $4.24 \times (10\text{의 } 10\text{제곱})$

`>>> a = 4.24e-10`    #  $4.24 \times (10\text{의 } -10\text{제곱})$

# 8진수와 16진수

8진수(octal)를 만들기 위해서는 숫자가 0o 또는 0O(숫자 0 + 알파벳 소문자 o 또는 대문자 O)으로 시작하면 된다.

```
>>> a = 0o236  
>>> print(a)  
158
```

16진수(hexadecimal)를 만들기 위해서는 0x로 시작하면 된다.

```
>>> a = 0x8ff  
>>> b = 0xABc  
>>> print(b)  
2748
```

8진수나 16진수는 파이썬에서 잘 사용하지 않는 형태의 숫자 자료형이므로 간단히 눈으로만 익히고 넘어가자.

# 숫자형을 사용하기 위한 연산자

## 사칙 연산

프로그래밍을 한 번도 해 본 적이 없는 독자라도 사칙 연산(`+`, `-`, `*`, `/`)은 알고 있을 것이다. 파이썬 역시 계산기와 마찬가지로 다음처럼 연산자를 사용해 사칙 연산을 수행한다.

```
>>> a = 3
>>> b = 4
>>> a + b
7
>>> a - b
-1
>>> a * b
12
>>> a / b
0.75
```

## x의 y제곱을 나타내는 \*\* 연산자

다음으로 알아야 할 연산자로 `**`라는 연산자가 있다. 이 연산자는 `x ** y`처럼 사용했을 때 x의 y제곱( $x^y$ ) 값을 리턴한다. 다음 예를 통해 알아보자.

```
>>> a = 3  
>>> b = 4  
>>> a ** b  
81
```

## 나눗셈 후 나머지를 리턴하는 `%` 연산자

프로그래밍을 처음 접하는 독자라면 `%` 연산자는 본 적이 없을 것이다. `%`는 나눗셈의 나머지 값을 리턴하는 연산자이다. 7을 3으로 나누면 나머지는 1, 3을 7로 나누면 나머지는 3이 될 것이다. 다음 예로 확인해 보자.

A hand-drawn diagram of long division. Inside a rectangular frame, 7 is divided by 3. The quotient is written as 2 above the division bar. The remainder is 1, which is written below the division bar and followed by the text "→ 나머지".

```
>>> 7 % 3
```

```
1
```

```
>>> 3 % 7
```

```
3
```

## 나눗셈 후 몫을 리턴하는 // 연산자

/ 연산자를 사용하여 7 나누기 4를 하면 그 결과는 예상대로 1.75가 된다.

```
>>> 7 / 4  
1.75
```

이번에는 나눗셈 후 몫을 리턴하는 // 연산자를 사용한 경우를 살펴보자.

```
>>> 7 // 4  
1
```

1.75에서 몫에 해당하는 정수값 1만 리턴하는 것을 확인할 수 있다.

A hand-drawn diagram of a long division problem. It shows 7 divided by 4. The quotient is 1, and the remainder is 3. An arrow points from the quotient '1' to the label '몫' (quotient).



## 2. 문자열 자료형

문자열(string)이란 문자, 단어 등으로 구성된 문자들의 집합을 말한다. 예를 들면 다음과 같다.

```
"Life is too short, You need Python"  
"a"  
"123"
```

모든 예문이 큰따옴표(")로 둘러싸여 있다. ‘123은 숫자인데 왜 문자열이지?’라는 의문이 드는 독자도 있을 것이다. 따옴표로 둘러싸여 있으면 모두 문자열이라고 보면 된다.

# 문자열은 어떻게 만들고 사용할까?

앞에서는 문자열을 만들 때 큰따옴표만 사용했지만, 이 밖에도 문자열을 만드는 방법은 3가지가 더 있다. 즉, 파이썬에서 문자열을 만드는 방법은 총 4가지이다.

앞에서는 문자열을 만들 때 큰따옴표만 사용했지만, 이 밖에도 문자열을 만드는 방법은 3가지가 더 있다. 즉, 파이썬에서 문자열을 만드는 방법은 총 4가지이다.

### 1. 큰따옴표로 양쪽 둘러싸기

```
"Hello World"
```

### 2. 작은따옴표로 양쪽 둘러싸기

```
'Python is fun'
```

### 3. 큰따옴표 3개를 연속으로 써서 양쪽 둘러싸기

```
"""Life is too short, You need python"""
```

### 4. 작은따옴표 3개를 연속으로 써서 양쪽 둘러싸기

```
'''Life is too short, You need python'''
```

## 문자열 안에 작은따옴표나 큰따옴표를 포함시키고 싶을 때

문자열을 만들어 주는 주인공은 작은따옴표(')와 큰따옴표(")이다. 그런데 문자열 안에도 작은따옴표와 큰따옴표가 들어 있어야 할 경우가 있다. 이때는 좀 더 특별한 기술이 필요하다. 예제를 하나씩 살펴보면서 원리를 익혀 보자.

# 1. 문자열에 작은따옴표 포함하기

```
Python's favorite food is perl
```

위와 같은 문자열을 `food` 변수에 저장하고 싶다고 가정해 보자. 문자열 중 Python's에 작은따옴표(')가 포함되어 있다.

이 경우에는 문자열을 큰따옴표로 둘러싸야 한다. 큰따옴표 안에 들어 있는 작은따옴표는 문자열을 나타내기 위한 기호로 인식되지 않는다. 대화형 인터프리터를 열고 다음과 같이 입력해 보자.

```
>>> food = "Python's favorite food is perl"
```

프롬프트에 'food'를 입력해서 결과를 확인해 보면 변수에 저장된 문자열이 그대로 출력되는 것을 볼 수 있다.

```
>>> food  
"Python's favorite food is perl"
```

시험삼아 다음과 같이 문자열을 큰따옴표가 아닌 작은따옴표로 둘러싼 후 다시 실행해 보자. 'Python'이 문자열로 인식되어 **구문오류(SyntaxError)**가 발생할 것이다.

```
>>> food = 'Python's favorite food is perl'  
File "<stdin>", line 1  
    food = 'Python's favorite food is perl'  
          ^  
SyntaxError: invalid syntax
```

## 2. 문자열에 큰따옴표 포함하기

```
"Python is very easy." he says.
```

위와 같이 큰따옴표가 포함된 문자열이라면 어떻게 해야 큰따옴표가 제대로 표현될까?

문자열을 작은따옴표로 둘러싸면 된다. 다음과 같이 입력해 보자.

```
>>> say = '"Python is very easy." he says.'
```

이렇게 작은따옴표 안에 사용된 큰따옴표는 문자열을 만드는 기호로 인식되지 않는다.

### 3. 역슬래시를 사용해서 작은따옴표와 큰따옴표를 문자열에 포함하기

```
>>> food = 'Python\'s favorite food is perl'  
>>> say = "\"Python is very easy.\\" he says."
```

작은따옴표나 큰따옴표를 문자열에 포함시키는 또 다른 방법은 역슬래시(\)를 사용하는 것이다. 즉, 역슬래시를 작은따옴표나 큰따옴표 앞에 삽입하면 역슬래시 뒤의 작은따옴표나 큰따옴표는 문자열을 둘러싸는 기호의 의미가 아니라 '나' 자체를 뜻하게 된다.

어떤 방법을 사용해서 문자열 안에 작은따옴표(')와 큰따옴표(")를 포함시킬 것인지는 각자의 선택이다. 대화형 인터프리터를 실행한 후 위 예문을 꼭 직접 작성해 보자.

## 여러 줄인 문자열을 변수에 대입하고 싶을 때

문자열이 항상 한 줄짜리만 있는 것은 아니다. 다음과 같은 여러 줄의 문자열을 변수에 대입하려면 어떻게 해야 할까?

## 1. 줄을 바꾸기 위한 이스케이프 코드 `\n` 삽입하기

```
>>> multiline = "Life is too short\nYou need python"
```

위 예처럼 줄바꿈 문자인 `\n`을 삽입하는 방법이 있지만, 읽기가 불편하고 줄이 길어지는 단점이 있다.

## 2. 연속된 작은따옴표 3개 또는 큰따옴표 3개 사용하기

1번 방법의 단점을 극복하기 위해 파이썬에서는 다음과 같이 작은따옴표 3개('') 또는 큰따옴표 3개(")")를 사용한다.

```
>>> multiline=''  
... Life is too short  
... You need python  
... '''
```

작은따옴표 3개를 사용한 경우

```
>>> multiline=""  
... Life is too short  
... You need python  
... """
```

## 큰따옴표 3개를 사용한 경우

'print(multiline)'을 입력하면 어떻게 출력되는지 확인해 보자.

```
>>> print(multiline)
Life is too short
You need python
```

두 경우 모두 결과는 동일하다. 위 예에서도 확인할 수 있듯이 문자열이 여러 줄인 경우, 이스케이프 코드를 쓰는 것보다 따옴표 3 개를 사용하는 것이 훨씬 깔끔하다.

## 점프 투 파일

### 이스케이프 코드란?

문자열 예제에서 여러 줄의 문장을 처리할 때 역슬래시 문자와 소문자 n을 조합한 `\n` 이스케이프 코드를 사용했다. 이스케이프(escape) 코드란 프로그래밍할 때 사용할 수 있도록 미리 정의해 둔 ‘문자 조합’을 말한다. 주로 출력물을 보기 좋게 정렬하는 용도로 사용한다. 몇 가지 이스케이프 코드를 정리하면 다음과 같다.

코드	설명
<code>\n</code>	문자열 안에서 줄을 바꿀 때 사용
<code>\t</code>	문자열 사이에 탭 간격을 줄 때 사용
<code>\\\</code>	<code>\</code> 를 그대로 표현할 때 사용
<code>\'</code>	작은따옴표(')를 그대로 표현할 때 사용
<code>\"</code>	큰따옴표(")를 그대로 표현할 때 사용
<code>\r</code>	캐리지 리턴(줄 바꿈 문자, 커서를 현재 줄의 가장 앞으로 이동)
<code>\f</code>	폼 피드(줄 바꿈 문자, 커서를 현재 줄의 다음 줄로 이동)
<code>\a</code>	벨 소리(출력할 때 PC 스피커에서 '삑' 소리가 난다)
<code>\b</code>	백 스페이스
<code>\000</code>	널 문자

이 중에서 활용 빈도가 높은 것은 `\n`, `\t`, `\\\`, `\'`, `\"`이다. 나머지는 프로그램에서 잘 사용하지 않는다.

## 문자열 연산하기

파이썬에서는 문자열을 더하거나 곱할 수 있다. 이는 다른 언어에서는 쉽게 찾아볼 수 없는 재미있는 기능으로, 우리 생각을 그대로 반영해 주는 파이썬만의 장점이라고 할 수 있다. 문자열을 더하거나 곱하는 방법에 대해 알아보자.

## 문자열 더해서 연결하기

```
>>> head = "Python"  
>>> tail = " is fun!"  
>>> head + tail  
'Python is fun!'
```

위 소스 코드에서 세 번째 줄을 살펴보자. 복잡하게 생각하지 말고 눈에 보이는 대로 생각해 보자. "Python"이라는 head 변수와 " is fun!"이라는 tail 변수를 더한 것이다. 결과는 'Python is fun!'이다. 즉, head와 tail 변수가 +에 의해 합쳐진 것이다. 직접 실행해 보고 결괏값이 제시한 것과 똑같이 나오는지 확인해 보자.

## 문자열 곱하기

```
>>> a = "python"  
>>> a * 2  
'pythonpython'
```

위 소스 코드에서 `*`의 의미는 우리가 일반적으로 사용하는 숫자 곱하기의 의미와는 다르다. 위 소스 코드에서 `a * 2`라는 문장은 a를 2번 반복하라는 뜻이다. 즉, `*`는 문자열의 반복을 뜻하는 의미로 사용되었다. 굳이 코드의 의미를 설명할 필요가 없을 정도로 직관적이다.

## 문자열 곱하기를 응용하기

문자열 곱하기를 좀 더 응용해 보자. 다음 소스를 IDLE 에디터를 열어 작성해 보자.

```
# multistring.py

print("=" * 50)
print("My Program")
print("=" * 50)
```

입력한 소스는 `C:\doit` 디렉터리에 ‘multistring.py’라는 파일 이름으로 저장하자.

이제 프로그램을 실행해 보자. [실행 창 열기([윈도우 + R](#)) → 'cmd' 입력 → `Enter` 입력]으로 명령 프롬프트 창을 열고 다음을 따라 해 보자. 다음과 같은 결괏값이 나타날 것이다.

```
C:\Users>cd C:\doit
C:\doit>python multistring.py
=====
My Program
=====
```

이런 식의 표현은 앞으로 자주 사용하게 될 것이다. 프로그램을 만들어 실행시켰을 때 출력되는 화면의 제일 위쪽에 프로그램 제목을 이와 같이 표시하면 보기 좋지 않겠는가?

# 문자열 길이 구하기

문자열의 길이는 다음과 같이 `len` 함수를 사용하면 구할 수 있다. `len` 함수는 `print` 함수처럼 파이썬의 기본 내장 함수로, 별다른 설정 없이 바로 사용할 수 있다.

문자열의 길이에는 공백 문자도 포함된다.

```
>>> a = "Life is too short"
>>> len(a)
17
```

## 문자열 인덱싱과 슬라이싱

인덱싱(indexing)이란 무엇인가를 ‘가리킨다’, 슬라이싱(slicing)은 무엇인가를 ‘잘라 낸다’라는 의미이다. 이런 의미를 생각하면서 다음 내용을 살펴보자.

## 문자열 인덱싱

```
>>> a = "Life is too short, You need Python"
```

위 코드에서 변수 a에 저장한 문자열의 각 문자마다 번호를 매겨 보면 다음과 같다.

L	i	f	e		i	s		t	o	o		s	h	o	r	t	,		Y	o	u		n	e	e	d		P	y	t	h	o	n
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33

"Life is too short, You need Python" 문자열에서 L은 첫 번째 자리를 뜻하는 숫자 0, i는 1 이런 식으로 계속 번호를 붙인 것이다. 즉, 중간에 있는 short의 s는 12가 된다.

이제 다음 예를 실행해 보자.

```
>>> a = "Life is too short, You need Python"  
>>> a[3]  
'e'
```

a[3]이 뜻하는 것은 a라는 문자열의 네 번째 문자 e를 말한다. 프로그래밍을 처음 접하는 독자라면 a[3]에서 숫자 3이 왜 네 번째 문자를 뜻하는지 의아할 수도 있다. 사실 이 부분이 헷갈릴 수 있는 부분인데, 다음과 같이 생각하면 쉽게 알 수 있을 것이다.

## 문자열 인덱싱 활용하기

인덱싱의 예를 몇 가지 더 살펴보자.

```
>>> a = "Life is too short, You need Python"  
>>> a[0]  
'L'  
>>> a[12]  
's'  
>>> a[-1]  
'n'
```

앞의 a[0]과 a[12]는 쉽게 이해할 수 있는데, 마지막의 a[-1]이 뜻하는 것은 뭘까? 눈치 빠른 독자는 이미 알아챘겠지만, 문자열을 뒤에서부터 읽기 위해 -(빼기) 기호를 붙인 것이다. 즉, a[-1]은 뒤에서부터 세어 첫 번째가 되는 문자를 말한다. a의 값은 "Life is too short, You need Python" 문자열이므로 뒤에서부터 첫 번째 문자는 가장 마지막 문자 'n'이다.

뒤에서부터 첫 번째 문자를 표시할 때도 0부터 세어 'a[-0]'이라고 해야 하지 않을까?'라는 의문이 들 수도 있겠지만, 잘 생각해 보자. 0과 -0은 똑같은 것이기 때문에 a[-0]은 a[0]과 똑같은 값을 보여 준다.

```
>>> a[-0]  
'L'
```

## 문자열 슬라이싱

그렇다면 "Life is too short, You need Python" 문자열에서 단순히 한 문자만을 뽑아 내는 것이 아니라 'Life' 또는 'You'와 같은 단어를 뽑아 내는 방법은 없을까?

```
>>> a = "Life is too short, You need Python"  
>>> b = a[0] + a[1] + a[2] + a[3]  
>>> b  
'Life'
```

위와 같이 단순하게 접근할 수도 있지만 파이썬에서는 더 좋은 방법을 제공한다. 바로 슬라이싱(slicing) 기법이다. 위 예는 슬라이싱 기법으로 다음과 같이 간단하게 처리할 수 있다.

인덱싱 기법과 슬라이싱 기법은 뒤에서 배울 자료형인 리스트나 튜플에서도 사용할 수 있다.

```
>>> a = "Life is too short, You need Python"  
>>> a[0:4]  
'Life'
```

a[0:4]는 a 문자열, 즉 "Life is too short, You need Python" 문자열에서 자리 번호 0부터 4까지의 문자를 뽑아 낸다는 뜻이다.

하지만 'a[0]은 L, a[1]은 i, a[2]는 f, a[3]은 e이므로 a[0:3]으로도 Life라는 단어를 뽑아 낼 수 있지 않을까?'라는 의문이 생길 것이다. 다음 예로 확인해 보자.

```
>>> a[0:3]  
'Lif'
```

이렇게 되는 이유는 슬라이싱 기법으로 `a[시작_번호:끝_번호]`를 지정할 때 끝 번호에 해당하는 문자는 포함하지 않기 때문이다. 즉, `a[0:3]`을 수식으로 나타내면 다음과 같다.

```
0 <= a < 3
```

이 수식을 만족하는 것은 `a[0], a[1], a[2]`이다. 따라서 `a[0:3]`은 'Lif', `a[0:4]`는 'Life'가 되는 것이다. 이 부분이 문자열 연산에서 가장 혼동하기 쉬운 부분이므로 02장의 마지막 부분에 있는 되새김 문제를 많이 풀어 보면서 몸에 익히기 바란다.

## 문자열을 슬라이싱하는 방법

슬라이싱의 예를 좀 더 살펴보자.

```
>>> a[0:5]  
'Life '
```

위 예는  $a[0] + a[1] + a[2] + a[3] + a[4]$ 와 동일하다.  $a[4]$ 는 공백 문자이기 때문에 'Life'가 아닌 'Life '가 출력된다. 공백 문자 역시 L, i, f, e 와 같은 문자와 동일하게 취급되는 것을 잊지 말자. 'Life'와 'Life '는 완전히 다른 문자열이다.

슬라이싱할 때 항상 시작 번호가 0일 필요는 없다.

```
>>> a[0:2]  
'Li'  
>>> a[5:7]  
'is'  
>>> a[12:17]  
'short'
```

$a[\text{시작\_번호}:\text{끝\_번호}]$ 에서 끝 번호 부분을 생략하면 시작 번호부터 그 문자열의 끝까지 뽑아 낸다.

a[시작\_번호:끝\_번호]에서 시작 번호를 생략하면 문자열의 처음부터 끝 번호까지 뽑아 낸다.

```
>>> a[:17]
'Life is too short'
```

a[시작\_번호:끝\_번호]에서 시작 번호와 끝 번호를 생략하면 문자열의 처음부터 끝까지 뽑아 낸다.

```
>>> a[:]
'Life is too short, You need Python'
```

슬라이싱에서도 인덱싱과 마찬가지로 -(빼기) 기호를 사용할 수 있다.

```
>>> a[19:-7]
'You need'
```

a[19:-7]은 a[19]에서 a[-8]까지를 의미한다. 이때에도 a[-7]은 포함하지 않는다.

## 슬라이싱으로 문자열 나누기

다음은 자주 사용하는 슬라이싱 기법 중 하나이다.

```
>>> a = "20230331Rainy"
>>> date = a[:8]
>>> weather = a[8:]
>>> date
'20230331'
>>> weather
'Rainy'
```

위 예는 문자열 a를 두 부분으로 나누는 기법이다. 숫자 8을 기준으로 문자열 a를 양쪽으로 한 번씩 슬라이싱했다. a[:8]은 a[8]을 포함하지 않고 a[8:]은 a[8]을 포함하기 때문에 8을 기준으로 해서 두 부분으로 나눌 수 있는 것이다. 위 예에서는 "20230331Rainy" 문자열을 날짜를 나타내는 부분인 '20230331'과 날씨를 나타내는 부분인 'Rainy'로 나누는 방법을 보여 준다.

"20230331Rainy"를 연도인 2023, 월과 일을 나타내는 0331, 날씨를 나타내는 Rainy까지 세 부분으로 나누는 방법은 다음과 같다.

```
>>> a = "20230331Rainy"
>>> year = a[:4]
>>> day = a[4:8]
>>> weather = a[8:]
>>> year
'2023'
>>> day
'0331'
>>> weather
'Rainy'
```

위 예는 숫자 4와 8로 "20230331Rainy" 문자열을 세 부분으로 나누는 방법을 보여 준다.

지금까지 인덱싱과 슬라이싱에 대해서 살펴보았다. 인덱싱과 슬라이싱은 프로그래밍할 때 자주 사용하는 기법이므로 꼭 반복해서 연습해 두자.

## Python 문자열을 Python으로 바꾸려면?

Python 문자열을 Python으로 바꾸려면 어떻게 해야 할까? 제일 먼저 떠오르는 생각은 다음과 같을 것이다.

```
>>> a = "Pithon"
>>> a[1]
'i'
>>> a[1] = 'y'
```

즉, a 변수에 "Pithon" 문자열을 대입하고 a[1]의 값이 i이므로 a[1]을 y로 바꾸어 준다는 생각이다. 하지만 결과는 어떻게 나올까? 당연히 오류가 발생한다. 문자열의 요솟값은 바꿀 수 있는 값이 아니기 때문이다(그래서 문자열을 '변경 불가능한 (immutable) 자료형'이라고도 부른다).

하지만 앞에서 배운 슬라이싱 기법을 사용하면 Python 문자열을 사용해 Python 문자열을 만들 수 있다. 다음 예를 살펴보자.

```
>>> a = "Pithon"
>>> a[:1]
'P'
>>> a[2:]
'thon'
>>> a[:1] + 'y' + a[2:]
'Python'
```

슬라이싱을 사용하면 "Pithon" 문자열을 'P' 부분과 'thon' 부분으로 나눌 수 있고, 그 사이에 'y' 문자를 추가하면 'Python'이라는 새로운 문자열을 만들 수 있다.

# 문자열 포매팅이란?

문자열에서 또 하나 알아야 할 것으로는 ‘문자열 포매팅(string formatting)’이 있다. 문자열 포매팅을 공부하기 전에 다음과 같은 문자열을 출력하는 프로그램을 작성했다고 가정해 보자.

```
"현재 온도는 18도입니다."
```

시간이 지나서 20도가 되면 다음 문장을 출력한다.

```
"현재 온도는 20도입니다"
```

두 문자열은 모두 같은데 20이라는 숫자와 18이라는 숫자만 다르다. 이렇게 문자열 안의 특정한 값을 바꿔야 할 경우가 있을 때 이를 가능하게 해 주는 것이 바로 문자열 포매팅이다.

쉽게 말해 문자열 포매팅이란 문자열 안에 어떤 값을 삽입하는 방법이다. 다음 예를 직접 실행해 보면서 그 사용법을 알아보자.

# 문자열 포매팅 따라 하기

## 1. 숫자 바로 대입

결과값을 보면 알겠지만, 위 예제는 문자열 안에 정수 3을 삽입하는 방법을 보여 준다. 문자열 안의 숫자를 넣고 싶은 자리에 `%d` 문자를 넣어 주고 삽입할 숫자 3은 가장 뒤에 있는 `%` 문자 다음에 써 넣었다. 여기에서 `%d`는 ‘문자열 포맷 코드’라고 부른다.

```
>>> "I eat %d apples." % 3  
'I eat 3 apples.'
```

## 2. 문자열 바로 대입

문자열 안에 꼭 숫자만 넣으라는 법은 없다. 이번에는 숫자 대신 문자열을 넣어 보자.

```
>>> "I eat %s apples." % "five"  
'I eat five apples.'
```

문자열 안에 또 다른 문자열을 삽입하기 위해 앞에서 사용한 문자열 포맷 코드 `%d` 가 아닌 `%s` 를 썼다. 어쩌면 눈치 빠른 독자는 숫자를 넣기 위해서는 `%d` , 문자열을 넣기 위해서는 `%s` 를 써야 한다는 사실을 눈치챘을 것이다.

앞에서 배운 것처럼 문자열을 대입할 때는 반드시 큰따옴표나 작은따옴표를 써야 한다.

### 3. 숫자 값을 나타내는 변수로 대입

```
>>> number = 3  
>>> "I eat %d apples." % number  
'I eat 3 apples.'
```

1번처럼 숫자를 바로 대입하든, 위 예제처럼 숫자 값을 나타내는 변수를 대입하든 결과는 같다.

## 4. 2개 이상의 값 넣기

그렇다면 문자열 안에 1개가 아닌 여러 개의 값을 넣고 싶을 때는 어떻게 해야 할까?

```
>>> number = 10
>>> day = "three"
>>> "I ate %d apples. so I was sick for %s days." % (number, day)
'I ate 10 apples. so I was sick for three days.'
```

2개 이상의 값을 넣으려면 마지막 `%` 다음 괄호 안에 쉼표(,)로 구분하여 각각의 값을 넣어 주면 된다.

# 문자열 포맷 코드

문자열 포매팅 예제에서는 대입해 넣는 자료형으로 정수와 문자열을 사용했지만, 이 밖에도 다양한 것을 대입할 수 있다. 문자열 포맷 코드의 종류는 다음과 같다.

코드	설명
%s	문자열(String)
%c	문자 1개(character)
%d	정수(Integer)
%f	부동소수(floating-point)
%o	8진수
%x	16진수
%%	Literal % (문자 % 자체)

여기에서 재미있는 것은 `%s` 포맷 코드인데, 이 코드에는 어떤 형태의 값이든 변환해 넣을 수 있다. 무슨 말인지 예를 통해 확인해 보자.

```
>>> "I have %s apples" % 3  
'I have 3 apples'  
>>> "rate is %s" % 3.234  
'rate is 3.234'
```

3을 문자열 안에 삽입하려면 `%d`를 사용해야 하고 3.234를 삽입하려면 `%f`를 사용해야 한다. 하지만 `%s`를 사용하면 `%s`는 자동으로 `%` 뒤에 있는 3이나 3.234와 같은 값을 문자열로 바꾸어 대입하기 때문에 이런 것을 생각하지 않아도 된다.

## 점프 투 파이썬

포매팅 연산자 `%d` 와 `%` 를 같이 쓸 때는 `%%` 를 쓴다

```
>>> "Error is %d%." % 98
```

결과값으로 당연히 "Error is 98%." 가 출력될 것이라고 예상하겠지만, 파이썬은 '형식이 불완전하다'라는 오류 메시지를 보여 준다.

```
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
ValueError: incomplete format
```

그 이유는 '문자열 포맷 코드인 `%d` 와 `%` 가 같은 문자열 안에 존재하는 경우, `%` 를 나타내려면 반드시 `%%` 를 써야 한다'라는 법칙이 있기 때문이다. 이 점은 꼭 기억해 두어야 한다. 하지만 문자열 안에 `%d` 와 같은 포매팅 연산자가 없으면 `%` 는 홀로 쓰여도 상관없다. 따라서 위 예를 제대로 실행하려면 다음과 같이 작성해야 한다.

```
>>> "Error is %d%%." % 98
'Error is 98.'
```

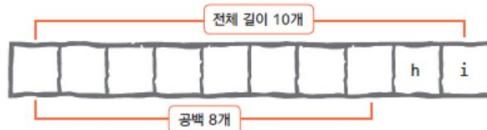
## 포맷 코드와 숫자 함께 사용하기

앞에서 살펴보았듯이 `%d`, `%s` 등과 같은 포맷 코드는 문자열 안에 어떤 값을 삽입할 때 사용한다. 하지만 포맷 코드를 숫자와 함께 사용하면 더 유용하다. 다음 예를 따라해 보자.

## 1. 정렬과 공백

```
>>> "%10s" % "hi"  
'          hi'
```

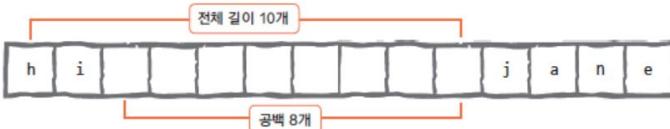
%10s 는 전체 길이가 10개인 문자열 공간에서 대입되는 값을 오른쪽으로 정렬하고 그 앞의 나머지는 공백으로 남겨 두라는 의미이다.



그렇다면 반대쪽인 왼쪽 정렬은 %-10s 가 될 것이다.

```
>>> "%-10sjane." % 'hi'  
'hi           jane.'
```

hi를 왼쪽으로 정렬하고 나머지는 공백으로 채웠다는 것을 알 수 있다.



## 2. 소수점 표현하기

```
>>> "%0.4f" % 3.42134234  
'3.4213'
```

3.42134234를 소수점 네 번째 자리까지만 나타내고 싶은 경우에는 위와 같이 작성한다. `%0.4f`에서 '.'는 소수점 포인트, 그 뒤의 숫자 4는 소수점 뒤에 나올 숫자의 개수를 말한다. 소수점 포인트 앞의 숫자는 문자열의 전체 길이를 의미하는데, `%0.4f`에서 사용한 숫자 0은 길이에 상관하지 않겠다는 의미이다.

`%0.4f`는 0을 생략하여 `.4f`처럼 사용하기도 한다.

다음 예를 살펴보자.

```
>>> "%10.4f" % 3.42134234  
' 3.4213'
```

위는 숫자 3.42134234를 소수점 네 번째 자리까지만 표시하고 전체 길이가 10개인 문자열 공간에서 오른쪽으로 정렬하는 예를 보여 준다.



# format 함수를 사용한 포매팅

## 숫자 바로 대입하기

```
>>> "I eat {0} apples".format(3)  
'I eat 3 apples'
```

"I eat {0} apples" 문자열 중 {0} 부분이 숫자 3으로 바뀌었다.

## 문자열 바로 대입하기

```
>>> "I eat {0} apples".format("five")  
'I eat five apples'
```

문자열의 {0} 항목이 'five'라는 문자열로 바뀌었다.

# format 함수를 사용한 포매팅

## 숫자 값을 가진 변수로 대입하기

```
>>> number = 3  
>>> "I eat {0} apples".format(number)  
'I eat 3 apples'
```

문자열의 {0} 항목이 number 변수의 값인 3으로 바뀌었다.

## 2개 이상의 값 넣기

```
>>> number = 10  
>>> day = "three"  
>>> "I ate {0} apples. so I was sick for {1} days.".format(number, day)  
'I ate 10 apples. so I was sick for three days.'
```

2개 이상의 값을 넣을 경우, 문자열의 {0}, {1} 과 같은 인덱스 항목이 format 함수의 입력값으로 순서에 맞게 바뀐다. 위 예에서 {0}은 format 함수의 첫 번째 입력값인 number, {1}은 format 함수의 두 번째 입력값인 day로 바뀐다.

# format 함수를 사용한 포매팅

## 이름으로 넣기

```
>>> "I ate {number} apples. so I was sick for {day} days.".format(number=10, day=3)
'I ate 10 apples. so I was sick for 3 days.'
```

{0}, {1} 과 같은 인덱스 항목 대신 더 편리한 {name} 형태를 사용하는 방법도 있다. {name} 형태를 사용할 경우, format 함수에는 반드시 name=value 와 같은 형태의 입력값이 있어야 한다. 위 예는 문자열의 {number}, {day} 가 format 함수의 입력값인 number=10, day=3 값으로 각각 바뀌는 것을 보여 주고 있다.

## 인덱스와 이름을 혼용해서 넣기

```
>>> "I ate {0} apples. so I was sick for {day} days.".format(10, day=3)
'I ate 10 apples. so I was sick for 3 days.'
```

인덱스 항목과 name=value 형태를 혼용하는 것도 가능하다.

# format 함수를 사용한 포매팅

## 왼쪽 정렬

```
>>> "{0:<10}".format("hi")
'hi      '
```

:<10 표현식을 사용하면 치환되는 문자열을 왼쪽으로 정렬하고 문자열의 총 자릿수를 10으로 맞출 수 있다.

## 오른쪽 정렬

```
>>> "{0:>10}".format("hi")
'      hi'
```

오른쪽 정렬은 :< 대신 :> 을 사용하면 된다. 화살표의 방향을 생각하면 어느 쪽으로 정렬되는지 바로 알 수 있을 것이다.

# format 함수를 사용한 포매팅

## 가운데 정렬

```
>>> "{0:^10}".format("hi")
'      hi      '
```

:^ 를 사용하면 가운데 정렬도 가능하다.

## 공백 채우기

```
>>> "{0:=^10}".format("hi")
'====hi===='
>>> "{0:!=<10}".format("hi")
'hi!!!!!!!'
```

정렬할 때 공백 문자 대신 지정한 문자 값으로 채워 넣을 수도 있다. 채워 넣을 문자 값은 정렬 문자 <, >, ^ 바로 앞에 넣어야 한다. 위 예에서 첫 번째 예제는 가운데(^)로 정렬하고 빈 공간을 = 로 채웠고, 두 번째 예제는 왼쪽(<)으로 정렬하고 빈 공간을 !로 채웠다.

# format 함수를 사용한 포매팅

## 소수점 표현하기

```
>>> y = 3.42134234  
>>> "{0:0.4f}".format(y)  
'3.4213'
```

위는 format 함수를 사용해 소수점을 4자리까지만 표현하는 방법을 보여 준다. 앞에서 살펴보았던 표현식 0.4f를 그대로 사용한 것을 알 수 있다.

```
>>> "{0:10.4f}".format(y)  
'      3.4213'
```

위와 같이 자릿수를 10으로 맞출 수도 있다. 이 또한 앞에서 살펴본 10.4f의 표현식을 그대로 사용한 것을 알 수 있다.

# format 함수를 사용한 포매팅

## { 또는 } 문자 표현하기

```
>>> "{{ and }}".format()  
'{ and }'
```

format 함수를 사용해 문자열을 포매팅할 경우, {} 와 같은 중괄호 문자를 포매팅 문자가 아닌 문자 그대로 사용하고 싶은 경우에는 위 예의 {{}} 처럼 2개를 연속해서 사용하면 된다.

# f 문자열 포매팅

---

파이썬 3.6 버전부터는 f 문자열 포매팅 기능을 사용할 수 있다. 파이썬 3.6 미만 버전에서는 사용할 수 없는 기능이므로 주의해야 한다.

다음과 같이 문자열 앞에 f 접두사를 붙이면 f 문자열 포매팅 기능을 사용할 수 있다.

```
>>> name = '홍길동'  
>>> age = 30  
>>> f'나의 이름은 {name}입니다. 나이는 {age}입니다.'  
'나의 이름은 홍길동입니다. 나이는 30입니다.'
```

f 문자열 포매팅은 위와 같이 name, age와 같은 변수값을 생성한 후에 그 값을 참조할 수 있다. 또한 f 문자열 포매팅은 표현식을 지원하기 때문에 다음과 같은 것도 가능하다.

표현식이란 중괄호 안의 변수를 계산식과 함께 사용하는 것을 말한다.

# f 문자열 포매팅

```
>>> age = 30  
>>> f'나는 내년이면 {age + 1}살이 된다.'  
'나는 내년이면 31살이 된다.'
```

딕셔너리는 f 문자열 포매팅에서 다음과 같이 사용할 수 있다.

```
>>> d = {'name':'홍길동', 'age':30}  
>>> f'나의 이름은 {d["name"]}입니다. 나이는 {d["age"]}입니다.'  
'나의 이름은 홍길동입니다. 나이는 30입니다.'
```

딕셔너리는 Key와 Value라는 것을 한 쌍으로 가지는 자료형이다. 02-5에서 자세히 알아본다.

정렬은 다음과 같이 할 수 있다.

```
>>> f'"hi":<10' # 左쪽 정렬  
'hi'  
>>> f'"hi":>10' # 오른쪽 정렬  
'    hi'  
>>> f'"hi":^10' # 가운데 정렬  
'  hi '
```

# f 문자열 포매팅

공백 채우기는 다음과 같이 할 수 있다.

```
>>> f'{"hi":^10}' # 가운데 정렬하고 '=' 문자로 공백 채우기  
'====hi===='  
>>> f'{"hi":!<10}' # 왼쪽 정렬하고 '!' 문자로 공백 채우기  
'hi!!!!!!!'
```

소수점은 다음과 같이 표현할 수 있다.

```
>>> y = 3.42134234  
>>> f'{y:0.4f}' # 소수점 4자리까지만 표현  
'3.4213'  
>>> f'{y:10.4f}' # 소수점 4자리까지 표현하고 총 자리수를 10으로 맞춤  
'     3.4213'
```

f 문자열에서 `{}` 를 문자 그대로 표시하려면 다음과 같이 2개를 동시에 사용해야 한다.

```
>>> f'{{ and }}'  
'{{ and }}'
```

# 문자열 관련 함수들

문자열 자료형은 자체적으로 함수를 가지고 있다. 이들 함수를 다른 말로 ‘문자열 내장 함수’라고 한다. 이 내장 함수를 사용하려면 문자열 변수 이름 뒤에 ‘.’를 붙인 후 함수 이름을 써 주면 된다. 이제 문자열의 내장 함수에 대해서 알아보자.

## 문자 개수 세기 - count

```
>>> a = "hobby"  
>>> a.count('b')  
2
```

count 함수로 문자열 중 문자 b의 개수를 리턴했다.

## 위치 알려 주기 1 - find

```
>>> a = "Python is the best choice"  
>>> a.find('b')  
14  
>>> a.find('k')  
-1
```

find 함수로 문자열 중 문자 b가 처음으로 나온 위치를 반환했다. 만약 찾는 문자나 문자열이 존재하지 않는다면 -1을 반환한다.

파이썬은 숫자를 0부터 세기 때문에 b의 위치는 15가 아닌 14가 된다.

# 문자열 관련 함수들

## 위치 알려 주기 2 - index

```
>>> a = "Life is too short"
>>> a.index('t')
8
>>> a.index('k')
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
ValueError: substring not found
```

index 함수로 문자열 중 문자 t가 맨 처음으로 나온 위치를 반환했다. 만약 찾는 문자나 문자열이 존재하지 않는다면 오류가 발생한다. 앞의 find 함수와 다른 점은 문자열 안에 존재하지 않는 문자를 찾으면 오류가 발생한다는 것이다.

# 문자열 관련 함수들

## 문자열 삽입 - join

```
>>> ",".join('abcd')  
'a,b,c,d'
```

join 함수로 abcd 문자열의 각각의 문자 사이에 ','를 삽입했다.

join 함수는 문자열뿐만 아니라 앞으로 배열 리스트나 튜플도 입력으로 사용할 수 있다(리스트와 튜플은 곧 배열 내용이므로 여기에서는 잠시 눈으로만 살펴보자). join 함수의 입력으로 리스트를 사용하는 예는 다음과 같다.

```
>>> ",".join(['a', 'b', 'c', 'd'])  
'a,b,c,d'
```

## 소문자를 대문자로 바꾸기 - upper

```
>>> a = "hi"  
>>> a.upper()  
'HI'
```

upper 함수는 소문자를 대문자로 바꾸어 준다. 만약 문자열이 이미 대문자라면 아무런 변화도 일어나지 않을 것이다.

# 문자열 관련 함수들

## 대문자를 소문자로 바꾸기 - lower

```
>>> a = "HI"  
>>> a.lower()  
'hi'
```

lower 함수는 대문자를 소문자로 바꾸어 준다.

## 왼쪽 공백 지우기 - lstrip

```
>>> a = " hi "  
>>> a.lstrip()  
'hi '
```

lstrip 함수는 문자열 중 가장 왼쪽에 있는 한 칸 이상의 연속된 공백들을 모두 지운다. lstrip에서 l은 left를 의미한다.

# 문자열 관련 함수들

## 오른쪽 공백 지우기 - `rstrip`

```
>>> a= " hi "
>>> a.rstrip()
' hi'
```

`rstrip` 함수는 문자열 중 가장 오른쪽에 있는 한 칸 이상의 연속된 공백을 모두 지운다. `rstrip`에서 r은 right를 의미한다.

## 양쪽 공백 지우기 - `strip`

```
>>> a = " hi "
>>> a.strip()
'hi'
```

`strip` 함수는 문자열 양쪽에 있는 한 칸 이상의 연속된 공백을 모두 지운다.

# 문자열 관련 함수들

## 문자열 바꾸기 - replace

```
>>> a = "Life is too short"
>>> a.replace("Life", "Your leg")
'Your leg is too short'
```

replace 함수는 replace(바뀔\_문자열, 바꿀\_문자열)처럼 사용해서 문자열 안의 특정한 값을 다른 값으로 치환해 준다.

## 문자열 나누기 - split

```
>>> a = "Life is too short"
>>> a.split()
['Life', 'is', 'too', 'short']
>>> b = "a:b:c:d"
>>> b.split(':')
['a', 'b', 'c', 'd']
```

split 함수는 a.split()처럼 괄호 안에 아무 값도 넣어 주지 않으면 공백( [Space] ), [Tab] , [Enter] )을 기준으로 문자열을 나누어 준다. 만약 b.split(':')처럼 괄호 안에 특정 값이 있을 경우에는 괄호 안의 값은 구분자로 해서 문자열을 나누어 준다. 이렇게 나눈 값은 리스트에 하나씩 들어간다. ['Life', 'is', 'too', 'short'] 나 ['a', 'b', 'c', 'd'] 가 리스트인데, 02-3에서 자세히 알아볼 것이므로 여기에서는 너무 신경 쓰지 않아도 된다.

# 문자열 관련 함수들

- 착각하기 쉬운 문자열 함수

소문자를 대문자로 바꾸어 주는 다음의 예를 보자.

```
>>> a = 'hi'  
>>> a.upper()  
'HI'
```

이와 같이 실행한 후에 a 변수의 값은 'HI'로 변했을까? 아니면 'hi' 값을 유지할까? 다음과 같이 확인해 보자.

```
>>> a  
'hi'
```

a.upper()를 수행하더라도 a 변수의 값은 변하지 않았다. 왜냐하면 a.upper()를 실행하면 upper 함수는 a 변수의 값 자체를 변경하는 것이 아니라 대문자로 바꾼 값을 리턴하기 때문이다. 문자열은 이전에도 잠깐 언급했지만 자체의 값을 변경할 수 없는 immutable 자료형이다. 따라서 a 값을 'HI'로 바꾸고 싶다면 다음과 같이 대입문을 사용해야 한다.

```
>>> a = a.upper()  
>>> a  
'HI'
```

upper 뿐만 아니라 lower, join, lstrip, rstrip, strip, replace, split 함수는 모두 이와 같은 규칙이 적용되어 문자열 자체의 값이 변경되는 것이 아니라 변경된 값을 리턴한다는 사실에 주의하자.



## 3. 리스트 자료형

지금까지 우리는 숫자와 문자열에 대해서 알아보았다. 하지만 숫자와 문자열만으로 프로그래밍을 하기에는 부족한 점이 많다. 예를 들어 1부터 10까지의 숫자 중 홀수의 모음인 1, 3, 5, 7, 9의 집합을 생각해 보자. 이런 숫자의 모음을 숫자나 문자열로 표현하기는 어렵다. 파이썬에는 이러한 불편함을 해소할 수 있는 자료형이 존재한다. 이것이 바로 지금부터 공부할 리스트(list)이다.

# 리스트를 만들고 사용하는 법

리스트를 사용하면 1, 3, 5, 7, 9의 숫자 모음을 다음과 같이 간단하게 표현할 수 있다.

```
>>> odd = [1, 3, 5, 7, 9]
```

리스트를 만들 때는 위에서 보는 것과 같이 대괄호([])로 감싸 주고 각 요소 값은 쉼표(,)로 구분해 준다.

리스트명 = [요소1, 요소2, 요소3, ...]

여러 가지 리스트의 생김새는 다음과 같다.

```
>>> a = []
```

```
>>> b = [1, 2, 3]
```

```
>>> c = ['Life', 'is', 'too', 'short']
```

```
>>> d = [1, 2, 'Life', 'is']
```

```
>>> e = [1, 2, ['Life', 'is']]
```

리스트는 a처럼 아무것도 포함하지 않아 비어 있는 리스트([])일 수도 있고, b처럼 숫자를 요솟값으로 가질 수도 있으며, c처럼 문자열을 요솟값으로 가질 수도 있다. 또한 d처럼 숫자와 문자열을 함께 요솟값으로 가질 수도 있고, e처럼 리스트 자체를 요솟값으로 가질 수도 있다. 즉, 리스트 안에는 어떠한 자료형도 포함할 수 있다.



# 리스트의 인덱싱과 슬라이싱

리스트 역시 문자열처럼 인덱싱을 적용할 수 있다. 먼저 `a` 변수에 `[1, 2, 3]` 값을 설정한다.

```
>>> a = [1, 2, 3]
```

```
>>> a [1, 2, 3]
```

`a[0]`은 리스트 `a`의 첫 번째 요솟값을 말한다.

```
>>> a[0]
```

1

다음 예는 리스트의 첫 번째 요소인 `a[0]`과 세 번째 요소인 `a[2]`의 값을 더한 것이다.

```
>>> a[0] + a[2]
```

4

이것은  $1 + 3$ 으로 해석되어 값 4를 출력한다.

# 리스트의 인덱싱과 슬라이싱

문자열을 공부할 때 이미 살펴보았지만, 파이썬은 숫자를 0부터 세기 때문에 `a[1]`이 리스트 `a`의 첫 번째 요소가 아니라 `a[0]`이 리스트 `a`의 첫 번째 요소라는 것을 명심하자. `a[-1]`은 문자열에서와 마찬가지로 리스트 `a`의 마지막 요솟값을 말한다.

```
>>> a[-1]
```

3

이번에는 다음 예처럼 리스트 `a`를 숫자 1, 2, 3과 또 다른 리스트인 `['a', 'b', 'c']`를 포함하도록 만들어 보자.

```
>>> a = [1, 2, 3, ['a', 'b', 'c']]
```

그리고 다음 예를 따라 해 보자.

```
>>> a[0] 1
```

```
>>> a[-1] ['a', 'b', 'c']
```

```
>>> a[3] ['a', 'b', 'c']
```

예상한 대로 `a[-1]`은 마지막 요솟값 `['a', 'b', 'c']`를 나타낸다. `a[3]`은 리스트 `a`의 네 번째 요소를 나타내기 때문에 마지막 요소를 나타내는 `a[-1]`과 동일한 결괏값을 보여 준다.

# 리스트의 인덱싱과 슬라이싱

그렇다면 리스트 `a`에 포함된 `['a', 'b', 'c']` 리스트에서 `'a'` 값을 인덱싱을 사용해 꺼집어 낼 수 있는 방법은 없을까? 다음 예를 살펴보자.

```
>>> a[-1][0]
```

```
'a'
```

위와 같이 하면 `'a'`를 꺼집어 낼 수 있다. `a[-1]`이 `['a', 'b', 'c']` 리스트라는 것은 이미 설명했다. 바로 이 리스트에서 첫 번째 요소를 불러오기 위해 `[0]`을 붙여 준 것이다.

# 리스트의 인덱싱과 슬라이싱

문자열과 마찬가지로 리스트에서도 슬라이싱 기법을 적용할 수 있다. 슬라이싱은 ‘잘라 낸다’라는 뜻이라고 했다. 리스트의 슬라이싱에 대해서 살펴보자.

```
>>> a = [1, 2, 3, 4, 5]
```

```
>>> a[0:2]
```

```
[1, 2]
```

앞의 예를 문자열에서 슬라이싱했던 예와 비교해 보자.

```
>>> a = "12345"
```

```
>>> a[0:2] '12'
```

2가지가 완전히 동일하게 사용되었다는 것을 눈치챘을 것이다. 문자열에서 했던 것과 사용법이 완전히 동일하다.

예시 ->

```
>>> a = [1, 2, 3, 4, 5]
```

```
>>> b = a[:2]
```

```
>>> c = a[2:]
```

```
>>> b
```

```
[1, 2]
```

```
>>> c
```

```
[3, 4, 5]
```

# 리스트 연산하기

## 1. 리스트 더하기

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> a + b
[1, 2, 3, 4, 5, 6]
```

리스트 사이에서 +는 2개의 리스트를 합치는 기능을 한다. 문자열에서 "abc" + "def" = "abcdef"가 되는 것과 같은 이치이다.

## 2. 리스트 반복하기

```
>>> a = [1, 2, 3]
>>> a * 3
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

위에서 볼 수 있듯이 [1, 2, 3] 리스트가 세 번 반복되어 새로운 리스트를 만들어 낸다. 문자열에서 "abc" \* 3 = "abcabcabc" 가 되는 것과 같은 이치이다.

# 리스트 연산하기

## 3. 리스트 길이 구하기

리스트 길이를 구하기 위해서는 다음처럼 `len` 함수를 사용해야 한다.

```
>>> a = [1, 2, 3]
>>> len(a)
3
```

`len`은 문자열, 리스트 외에 앞으로 배울 튜플과 딕셔너리에도 사용할 수 있는 함수이다. 실습에서 자주 사용하므로 잘 기억해 두자.

# 리스트의 수정과 삭제

리스트는 값을 수정하거나 삭제할 수 있다.

## 1. 리스트의 값 수정하기

```
>>> a = [1, 2, 3]
>>> a[2] = 4
>>> a
[1, 2, 4]
```

a[2]의 요솟값 3이 4로 바뀌었다.

## 2. del 함수를 사용해 리스트 요소 삭제하기

```
>>> a = [1, 2, 3]
>>> del a[1]
>>> a
[1, 3]
```

**del a[x]**는 x번째 요솟값을 삭제한다. 위에서는 a 리스트에서 a[1]을 삭제했다. **del** 함수는 파이썬이 자체적으로 가지고 있는 삭제함수이며 다음과 같이 사용한다.

# 리스트의 수정과 삭제

del 객체

다음처럼 슬라이싱 기법을 사용하여 리스트의 요소 여러 개를 한꺼번에 삭제할 수도 있다.

```
>>> a = [1, 2, 3, 4, 5]
>>> del a[2:]
>>> a
[1, 2]
```

a[2:]에 해당하는 리스트의 요소들이 삭제되었다.

리스트의 요소를 삭제하는 방법에는 2가지가 더 있다. 바로 리스트의 `remove`와 `pop` 함수를 사용하는 것인데, 이는 리스트 관련 함수에서 설명한다.

# 리스트 관련 함수

## 1. 리스트에 요소 추가하기 - append

`append`의 사전적 의미는 ‘덧붙이다, 첨부하다’이다. 이 뜻을 안다면 다음 예가 바로 이해될 것이다. `append(x)`는 리스트의 맨 마지막에 `x`를 추가하는 함수이다.

```
>>> a = [1, 2, 3]
>>> a.append(4)
>>> a
[1, 2, 3, 4]
```

리스트 안에는 어떤 자료형도 추가할 수 있다. 다음 예는 리스트에 다시 리스트를 추가한 결과이다.

```
>>> a.append([5, 6])
>>> a
[1, 2, 3, 4, [5, 6]]
```

# 리스트 관련 함수

## 1. 리스트 정렬 - sort

sort 함수는 리스트의 요소를 순서대로 정렬해 준다.

```
>>> a = [1, 4, 3, 2]
>>> a.sort()
>>> a
[1, 2, 3, 4]
```

문자 역시 알파벳 순서로 정렬할 수 있다.

```
>>> a = ['a', 'c', 'b']
>>> a.sort()
>>> a
['a', 'b', 'c']
```

# 리스트 관련 함수

## 1. 리스트 요소 끄집어 내기 - pop

`pop()`은 리스트의 맨 마지막 요소를 리턴하고 그 요소는 삭제한다.

```
>>> a = [1, 2, 3]
>>> a.pop()
3
>>> a
[1, 2]
```

`a` 리스트를 다시 보면 `[1, 2, 3]`에서 3을 끄집어 내고 최종적으로 `[1, 2]`만 남는 것을 확인할 수 있다.

`pop(x)`는 리스트의 `x`번째 요소를 리턴하고 그 요소는 삭제한다.

```
>>> a = [1, 2, 3]
>>> a.pop(1)
2
>>> a
[1, 3]
```

`a.pop(1)`은 `a[1]`의 값을 끄집어 내어 리턴한다. 다시 `a`를 출력해 보면 끄집어 낸 값이 삭제된 것을 확인할 수 있다.

# 리스트 관련 함수

1. 리스트에 포함된 요소 x의 개수 세기 - count

count(x)는 리스트 안에 x가 몇 개 있는지 조사하여 그 개수를 리턴하는 함수이다.

```
>>> a = [1, 2, 3, 1]
>>> a.count(1)
2
```

1이라는 값이 리스트 a에 2개 들어 있으므로 2를 리턴한다.

2. 리스트 확장 - extend

extend(x)에서 x에는 리스트만 올 수 있으며 원래의 a 리스트에 x 리스트를 더하게 된다.

a.extend([4, 5])는 a += [4, 5]와 동일하다.

```
>>> a = [1, 2, 3]
>>> a.extend([4, 5])
>>> a
[1, 2, 3, 4, 5]
>>> b = [6, 7]
>>> a.extend(b)
>>> a
[1, 2, 3, 4, 5, 6, 7]
```

# 리스트 관련 함수

## 1. 리스트에 요소 삽입 - insert

`insert(a, b)`는 리스트의 `a`번째 위치에 `b`를 삽입하는 함수이다. 파이썬은 숫자를 0부터 센다는 것을 반드시 기억하자.

```
>>> a = [1, 2, 3]
>>> a.insert(0, 4)
>>> a
[4, 1, 2, 3]
```

위 예는 0번째 자리, 즉 첫 번째 요소인 `a[0]` 위치에 값 4를 삽입하라는 뜻이다.

```
>>> a.insert(3, 5)
>>> a
[4, 1, 2, 5, 3]
```

위 예는 리스트 `a`의 `a[3]`, 즉 네 번째 요소 위치에 값 5를 삽입하라는 뜻이다.

# 리스트 관련 함수

## 1. 리스트 요소 제거 - remove

`remove(x)`는 리스트에서 첫 번째로 나오는 `x`를 삭제하는 함수이다.

```
>>> a = [1, 2, 3, 1, 2, 3]
>>> a.remove(3)
>>> a
[1, 2, 1, 2, 3]
```

`a`가 3이라는 값을 2개 가지고 있을 경우, 첫 번째 3만 제거되는 것을 알 수 있다.

```
>>> a.remove(3)
>>> a
[1, 2, 1, 2]
```

`remove(3)`을 한 번 더 실행하면 다시 3이 삭제된다.

# 리스트 관련 함수

## 1. 리스트 요소 제거 - remove

`remove(x)`는 리스트에서 첫 번째로 나오는 `x`를 삭제하는 함수이다.

```
>>> a = [1, 2, 3, 1, 2, 3]
>>> a.remove(3)
>>> a
[1, 2, 1, 2, 3]
```

`a`가 3이라는 값을 2개 가지고 있을 경우, 첫 번째 3만 제거되는 것을 알 수 있다.

```
>>> a.remove(3)
>>> a
[1, 2, 1, 2]
```

`remove(3)`을 한 번 더 실행하면 다시 3이 삭제된다.



## 4. 튜플 자료형

튜플(tuple)은 몇 가지 점을 제외하곤 리스트와 거의 비슷하며 리스트와 다른 점은 다음과 같다.

- 리스트는 [], 튜플은 ()으로 둘러싼다.
- 리스트는 요솟값의 생성, 삭제, 수정이 가능하지만, 튜플은 요솟값을 바꿀 수 없다.

## 튜플의 요솟값을 지우거나 변경하는 방법

### 1. 삭제하려 할 때

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

>>> del t1[0] -> 튜플 t1의 첫 번째 요소를 지우는 시도

ERROR (튜플의 요솟값은 삭제가 불가능해 에러 코드가 뜬다)

### 2. 변경하려 할 때

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

>>> t1[0] = 'c' -> 튜플 t1의 첫 번째 요솟값을 변경하려는 시도

ERROR (튜플의 요솟값은 변경이 불가능해 에러 코드가 뜬다)

# 튜플 다루기

## 1. 인덱싱하기

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

```
>>> t1[0]
```

1

## 2. 슬라이싱하기

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

```
>>> t1[2:] -> 2부터 끝까지
```

( 'a', 'b' )

# 튜플 다루기

## 3. 튜플 더하기

```
>>> t2 = (3, 4)
```

```
>>> t1 + t2
```

```
(1, 2, 'a', 'b', 3, 4)
```

## 4. 튜플 곱하기

```
>>> t2 * 3
```

```
(3, 4, 3, 4, 3, 4)
```

## 5. 튜플 길이 구하기

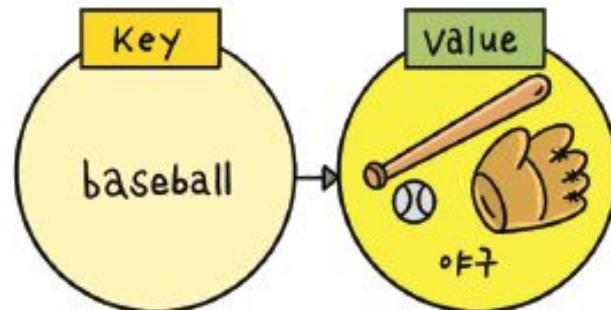
```
>>> t1 = (1, 2, 'a', 'b' )
```

```
>>> len(t1)
```



## 5. 딕셔너리 자료형

딕셔너리는 단어 그대로 '사전'이라는 뜻이다. 즉 "people"이라는 단어에 "사람", "baseball"이라는 단어에 "야구"라는 뜻이 부합되듯이 딕셔너리는 Key와 Value를 한 쌍으로 가지는 자료형이다. 예컨대 Key가 "baseball"이라면 Value는 "야구"가 될 것이다. 딕셔너리는 리스트나 튜플처럼 순차적으로(sequential) 해당 요솟값을 구하지 않고 Key를 통해 Value를 얻는다. 이것이 바로 딕셔너리의 가장 큰 특징이다. baseball이라는 단어의 뜻을 찾기 위해 사전의 내용을 순차적으로 모두 검색하는 것이 아니라 baseball이라는 단어가 있는 곳만 펼쳐 보는 것이다.



## 딕셔너리란?

<딕셔너리의 기본형>

```
>>> {Key1: Value1, Key2: Value2, Key3: Value3, ...}
```

ex) dic = {'name': 'pey', 'phone': '010-9999-1234', 'birth': '1118'} -> 여기서 Key는 각각 'name', 'phone', 'birth', 각각의 Key에 해당하는 Value는 'pey', '010-9999-1234', '1118'이 된다.

또한, >>> a = {'a': [1, 2, 3]} 처럼 리스트도 넣을 수 있다.

# 딕셔너리 다루기

## 1. 딕셔너리 쌍 추가하기

```
>>> a = {1: 'a'}
>>> a[2] = 'b'
>>> a
{1: 'a', 2: 'b'}
```

{1: 'a'} 딕셔너리에 a[2] = 'b'와 같이 입력하면 딕셔너리 a에 Key와 Value가 각각 2와 'b'인 {2: 'b'} 딕셔너리 쌍이 추가된다.

# 딕셔너리 다루기

## 2. 딕셔너리 요소 삭제하기

```
>>> del a[1]
>>> a
{2: 'b', 'name': 'pey', 3: [1, 2, 3]}
```

위 예제는 딕셔너리 요소를 지우는 방법을 보여 준다. `del` 함수를 사용해서 `del a[key]`를 입력하면 지정한 **Key**에 해당하는 **{Key: Value}** 쌍이 삭제된다.

## 딕셔너리를 사용하는 이유

예를 들어 4명의 사람이 있다고 가정하고 각자의 특기를 표현할 수 있는 좋은 방법에 대해서 생각해 보자. 리스트나 문자열로는 표현하기가 매우 까다로울 것이다. 하지만 파이썬의 딕셔너리를 사용하면 이 상황을 표현하기가 정말 쉽다.

```
{"김연아": "피겨스케이팅", "류현진": "야구", "손흥민": "축구", "귀도": "파이썬"}
```

사람 이름과 특기를 한 쌍으로 묶은 딕셔너리이다.

# 딕셔너리를 사용하는 방법

## 1. 딕셔너리에서 Key를 사용해 Value 얻기

```
>>> grade = {'pey': 10, 'julliet': 99}  
>>> grade['pey']  
10  
>>> grade['julliet']  
99
```

리스트나 튜플, 문자열은 요솟값을 얻고자 할 때 인덱싱이나 슬라이싱 기법 중 하나를 사용했다. 하지만 딕셔너리는 단 1가지 방법뿐이다. 그것은 바로 Key를 사용해서 Value를 구하는 방법이다. 위 예에서 'pey'라는 Key의 Value를 얻기 위해 **grade[ 'pey' ]**를 사용한 것처럼 어떤 Key의 Value를 얻기 위해서는 '딕셔너리\_변수\_이름[Key]'를 사용해야 한다.

# 딕셔너리를 사용하는 방법

## 2. 딕셔너리에서 주의할 사항

```
>>> a = {1:'a', 1:'b'}  
>>> a  
{1: 'b'}
```

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 2개 존재할 경우, 1: 'a' 쌍이 무시된다.

```
>>> a = {[1,2] : 'hi'}  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
TypeError: unhashable type: 'list'
```

+또 1가지 주의해야 할 점은 **Key**에 리스트는 쓸 수 없다는 것이다. 하지만 튜플은 **Key**로 쓸 수 있다. 다음 예처럼 리스트를 **Key**로 설정하면 리스트를 키 값으로 사용할 수 없다는 오류가 발생한다. 단, **value**에는 변하지 않는 값이나 변하는 값 둘다 넣을 수 있다.

# 딕셔너리를 사용하는 방법

## 3. 딕셔너리 관련 함수

```
>>> a = {'name': 'pey', 'phone': '010-9999-1234', 'birth': '1118'}  
>>> a.keys()  
dict_keys(['name', 'phone', 'birth'])
```

a.keys()는 딕셔너리 a의 Key만을 모아 dict\_keys 객체를 리턴한다.

```
>>> list(a.keys())  
['name', 'phone', 'birth']
```

dict\_keys 객체를 리스트로 변환하려면 다음과 같이 하면 된다.

## 딕셔너리를 사용하는 방법

### 4. VALUE 리스트 만들기

```
>>> a.values()
dict_values(['pey', '010-9999-1234', '1118'])
```

Key를 얻는 것과 마찬가지 방법으로 Value만 얻고 싶다면 values 함수를 사용하면 된다. values 함수를 호출하면 dict\_values 객체를 리턴한다.

### 5. Key, Value 쌍 얻기 - items

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

items 함수는 Key와 Value의 쌍을 튜플로 묶은 값을 dict\_items 객체로 리턴한다.

# 딕셔너리를 사용하는 방법

## 6. Key: Value 쌍 모두 지우기 - clear

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

clear 함수는 딕셔너리 안의 모든 요소를 삭제한다.

## 7. Key로 Value 얻기 - get

```
>>> a = {'name': 'pey', 'phone': '010-9999-1234', 'birth': '1118'}  
>>> a.get('name')  
'pey'  
>>> a.get('phone')  
'010-9999-1234'
```

get(x) 함수는 x라는 Key에 대응되는 Value를 리턴한다. 앞에서 살펴보았듯이 `a.get('name')`은 `a['name']`을 사용했을 때와 동일한 결괏값을 리턴한다.

## 딕셔너리를 사용하는 방법

### 8. 해당 Key가 딕셔너리 안에 있는지 조사하기 - in

```
>>> a = {'name':'pey', 'phone':'010-9999-1234', 'birth': '1118'}
>>> 'name' in a
True
>>> 'email' in a
False
```

'name' 문자열은 a 딕셔너리의 Key 중 하나이다. 따라서 'name' in a를 호출하면 참(True)을 리턴한다. 이와 반대로 'email'은 a 딕셔너리 안에 존재하지 않는 Key이므로 거짓(False)을 리턴한다.



## 6. 집합 자료형

집합(set)은 집합에 관련된 것을 쉽게 처리하기 위해 만든 자료형이다.

1. 집합 자료형은 어떻게 만들까?
2. 집합 자료형의 특징
3. 교집합, 합집합, 차집합 구하기
  1. 교집합 구하기
  2. 합집합 구하기
  3. 차집합 구하기
4. 집합 자료형 관련 함수
  1. 값 1개 추가하기 - `add`
  2. 값 여러 개 추가하기 - `update`
  3. 특정 값 제거하기 - `remove`



## 6 - 1 집합 자료형은 어떻게 만들까?

집합 자료형은 다음과 같이 `set` 키워드를 사용해 만들 수 있다.

```
>>> s1 = set([1, 5, 8])
>>> s1
{8, 1, 5}
```

위와 같이 `set()`의 괄호 안에 리스트를 입력하여 만들거나 다음과 같이 문자열을 입력하여 만들 수도 있다.

```
>>> s2 = set("Hello")
>>> s2
{'o', 'l', 'H', 'e'}
```

비어 있는 집합 자료형은 `s = set()`로 만들 수 있다.



## 6 - 2 집합 자료형의 특징

위에서 살펴본 `set("Hello")`의 결과가 좀 이상하지 않은가? 분명 "Hello" 문자열로 `set` 자료형을 만들었는데 생성된 자료형에는 1 문자가 하나 빠져 있고 순서도 뒤죽박죽이다. 그 이유는 `set`에 다음과 같은 2가지 특징이 있기 때문이다.

- 중복을 허용하지 않는다.
- 순서가 없다(Unordered).

리스트나 튜플은 순서가 있기(ordered) 때문에 인덱싱을 통해 요솟값을 얻을 수 있지만, `set` 자료형은 순서가 없기 (unordered) 때문에 인덱싱을 통해 요솟값을 얻을 수 없다. 이는 마치 02-5에서 살펴본 딕셔너리와 비슷하다. 딕셔너리 역시 순서가 없는 자료형이므로 인덱싱을 지원하지 않는다.

만약 `set` 자료형에 저장된 값을 인덱싱으로 접근하려면 다음과 같이 리스트나 튜플로 변환한 후에 해야 한다.



## 6 - 2 집합 자료형의 특징

만약 `set` 자료형에 저장된 값을 인덱싱으로 접근하려면 다음과 같이 리스트나 튜플로 변환한 후에 해야 한다.

```
>>> s1 = set([1, 5, 8])
>>> l1 = list(s1)
>>> l1
[8, 1, 5]
>>> l1[0]
8
>>> t1 = tuple(s1)
>>> t1
(8, 1, 5)
>>> t1[0]
8
```



## 6 - 3 교집합, 합집합, 차집합 구하기

set 자료형을 정말 유용하게 사용하는 경우는 교집합, 합집합, 차집합을 구할 때이다.

먼저 다음과 같이 2개의 set 자료형을 만든 후 따라 해 보자. s1에는 1부터 7까지의 값, s2에는 5부터 9까지의 값을 주었다.

```
>>> s1 = set([1, 2, 3, 4, 5, 6, 7])
>>> s2 = set([5, 6, 7, 8, 9])
```



## 6 - 3 - 1 교집합 구하기

s1과 s2의 교집합을 구해 보자. 다음과 같이 '&'를 이용하면 교집합을 간단히 구할 수 있다.

```
>>> s1 & s2  
{5, 6, 7}  
>>> s1.intersection(s2)  
{5, 6, 7}
```

또는 다음과 같이 intersection 함수를 사용해도 결과는 동일하다.

s2.intersection(s1)을 사용해도 결과는 동일하다.



## 6 - 3 - 2 합집합 구하기

이번에는 합집합을 구해 보자. ‘|’를 사용하면 합집합을 구할 수 있다. 이때 4, 5, 6처럼 중복해서 포함된 값은 1개씩만 표현된다.

```
>>> s1 | s2  
{1, 2, 3, 4, 5, 6, 7, 8, 9}  
>>> s1.union(s2)  
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

<<< **union** 함수를 사용해도 된다.

s2.union(s1)을 사용해도 결과는 동일하다.



## 6 - 3 - 3 차집합 구하기

마지막으로 차집합을 구해 보자. -(빼기)를 사용하면 차집합을 구할 수 있다.

```
>>> s1 - s2  
{1, 2, 3, 4}  
>>> s2 - s1  
{8, 9}
```

difference 함수를 사용해도 차집합을 구할 수 있다.

```
>>> s1.difference(s2)  
{1, 2, 3, 4}
```

s2.difference(s2)를 사용해도 결과는 동일하다.



## 6 - 4 집합 자료형 관련 함수

이번에는 **set** 자료형에 값을 지우거나 추가하는 함수를 알아보자



## 6 - 4 - 1 값 1개 추가하기

이미 만들어진 `set` 자료형에 값을 추가할 수 있다. 1개의 값만 추가`add`할 때는 다음과 같이 하면 된다.

```
>>> s1 = set([1, 3, 5])
>>> s1.add(4)
>>> s1
{1, 3, 4, 5}
```



## 6 - 4 - 2 값 여러개 추가하기

여러 개의 값을 한꺼번에 추가(update)할 때는 다음과 같이 하면 된다.

```
>>> s1 = set([1, 5, 6])
>>> s1.update([2, 3, 4])
>>> s1
{1, 2, 3, 4, 5, 6}
```



## 6 - 4 - 3 특정 값 제거하기

특정 값을 제거하고 싶을 때는 다음과 같이 하면 된다.

```
>>> s1 = set([1, 3, 4])
>>> s1.remove(3)
>>> s1
{1, 4}
```



## 7. 불 자료형

불(bool) 자료형이란 참(True)과 거짓(False)을 나타내는 자료형이다. 불 자료형은 다음 2가지 값만을 가질 수 있다.

True나 False는 파이썬의 예약어로, true, false와 같이 작성하면 안 되고 첫 문자를 항상 대문자로 작성해야 한다.



## 7 - 1 불 자료형은 어떻게 사용할까?

다음과 같이 변수 **a**에는 **True**, 변수 **b**에는 **False**를 지정해 보자.

```
>>> a = True
```

```
>>> b = False
```

따옴표로 감싸지 않은 문자열을 변수에 지정해서 오류가 발생할 것 같지만, 잘 실행된다. **type** 함수를 변수 **a**와 **b**에 사용하면 두 변수의 자료형이 **bool**로 지정된 것을 확인할 수 있다

```
>>> type(a)  
<class 'bool'>  
>>> type(b)  
<class 'bool'>
```



## 7 - 1 불 자료형은 어떻게 사용할까?

불 자료형은 조건문의 리턴값으로도 사용된다. 조건문에 대해서는 if 문에서 자세히 배우겠지만 잠시 살펴보고 넘어가자.

```
>>> 1 == 1  
True
```

`1 == 1` 은 '1과 1이 같은가?'를 묻는 조건문이다. 이런 조건문은 결과로 `True` 또는 `False`에 해당하는 불 자료형을 리턴한다. 1과 1은 같으므로 `True`를 리턴한다.

```
>>> 2 > 1  
True  
>>> 2 < 1  
False
```

2는 1보다 크므로 `2 > 1` 조건문은 참이다. 즉, `True`를 리턴한다.

2는 1보다 작지 않으므로 `2 < 1` 조건문은 거짓이다. 즉, `False`를 리턴한다.



## 7 - 1 - 1 자료형의 참과 거짓

'자료형에 참과 거짓이 있다?'라는 말이 조금 이상하게 들리겠지만,

참과 거짓은 분명히 있다. 이는 매우 중요한 특징이며 실제로도 자주 쓰인다.

자료형의 참과 거짓을 구분하는 기준은 다음과 같다. →

문자열, 리스트, 튜플, 딕셔너리 등의 값이 비어 있으면 ("", [], (), {}) 거짓이 되고 비어 있지 않으면 참이 된다. 숫자에서는 그 값이 0일 때 거짓이 된다. 위 표를 보면 `None`이 있는데, 이것에 대해서는 뒷부분에서 배우므로 아직은 신경 쓰지 말자. 그저 `None`은 거짓을 뜻한다는 것만 알아 두자.

값	참 or 거짓
"python"	참
""	거짓
[1, 2, 3]	참
[]	거짓
(1, 2, 3)	참
()	거짓
{'a': 1}	참
{}	거짓
1	참
0	거짓
None	거짓



## 7 - 1 - 2 자료형의 참과 거짓

다음 예를 보고 자료형의 참과 거짓이 프로그램에서 어떻게 쓰이는지 간단히 알아보자.

```
>>> a = [1, 2, 3, 4]
>>> while a:
...     print(a.pop())
...
4
3
2
1
```

먼저 `a = [1, 2, 3, 4]`라는 리스트를 만들었다. `while` 문은 03장에서 자세히 다루지만, 간단히 알아보면 다음과 같다. 조건문이 참인 동안 조건문 안에 있는 문장을 반복해서 수행한다.

`while` 조건문:  
수행할\_문장

즉, 위 예를 보면 `a`가 참인 경우, `a.pop()`를 계속 실행하여 출력하라는 의미이다. `a.pop()` 함수는 리스트 `a`의 마지막 요소를 끌어내는 함수이므로 리스트 안에 요소가 존재하는 한(`a`가 참인 동안) 마지막 요소를 계속 끌어낼 것이다. 결국 더 이상 끌어낼 것이 없으면 `a`가 빈 리스트(`[]`)가 되어 거짓이 된다. 따라서 `while` 문에서 조건문이 거짓이 되므로 `while` 문을 빠져나가게 된다. 이는 파이썬 프로그래밍에서 매우 자주 사용하는 기법 중 하나이다.



## 7 - 1 - 3 자료형의 참과 거짓

위 예가 너무 복잡하다고 생각하는 독자는 다음 예를 보면 쉽게 이해될 것이다.

```
>>> if []:
...     print("참")
... else:
...     print("거짓")
거짓
```

[]는 앞의 표에서 볼 수 있듯이 비어 있는 리스트이므로 거짓이다. 따라서 "거짓"이라는 문자열이 출력된다. if 문에 대해서 잘 모르는 독자라도 위 문장을 해석하는 데는 무리가 없을 것이다.

(if 문에 대해서는 03장에서 자세히 다룬다.)



## 7 - 2 불 연산

자료형에 참과 거짓이 있다는 것을 이제 알게 되었다. `bool` 함수를 사용하면 자료형의 참과 거짓을 보다 정확하게 식별할 수 있다.

다음 예제를 따라 해 보자.

```
>>> bool('python') 'python' 문자열은 비어 있지 않으므로 bool 연산의 결과로 불 자료형인 True를 리턴한다.
```

```
True
```

```
>>> bool('')
```

```
False
```

← "문자열은 비어 있으므로 bool 연산의 결과로 불 자료형인 False를 리턴한다."



## 7 - 2 - 2 불 연산

앞에서 알아본 몇 가지 예제를 더 수행해 보자.

```
>>> bool([1, 2, 3])  
True  
>>> bool([])  
False  
>>> bool(0)  
False  
>>> bool(3)  
True
```

앞에서 알아본 것과 동일한 참과 거짓에 대한 결과를 리턴하는 것을 확인할 수 있다.

지금까지 파이썬의 가장 기본이 되는 자료형인 숫자, 문자열, 리스트, 튜플, 딕셔너리, 집합, 불에 대해서 알아보았다. 여기까지 잘 따라온 독자라면 파이썬에 대해서 대략 50% 정도 습득했다고 보아도 된다. 그만큼 자료형은 중요하고 프로그램의 근간이 되기 때문에 확실하게 해 놓지 않으면 좋은 프로그램을 만들 수 없다. 이 책의 예제만 따라 하지 말고 직접 여러 가지 예들을 테스트해 보면서 02-1~02-7에 나오는 자료형에 익숙해지기 바란다.



## 8. 자료형의 값을 저장하는 공간, 변수

1. 변수는 어떻게 만들까?
2. 변수란?
3. 리스트를 복사하고자 할 때
  1. [:] 이용하기
  2. copy 모듈 이용하기
4. 변수를 만드는 여러 가지 방법



## 8 - 1 변수는 어떻게 만들까?

우리는 앞에서 이미 변수를 사용해 왔다. 다음 예와 같은 **a**, **b**, **c**를 ‘변수’라고 한다.

```
>>> a = 1  
>>> b = "python"  
>>> c = [1, 2, 3]
```

변수를 만들 때는 옆에 있는 예처럼 **=**(assignment) 기호를 사용한다.

변수\_이름 = 변수에\_저장할\_값

다른 프로그래밍 언어인 **C**나 **JAVA**에서는 변수를 만들 때 자료형의 타입을 직접 지정해야 한다. 하지만 파이썬은 변수에 저장된 값을 스스로 판단하여 자료형의 타입을 지정하기 때문에 더 편리하다.



## 8 - 2 변수란?

파이썬에서 사용하는 변수는 객체를 가리키는 것이라고도 말할 수 있다. 객체란 우리가 지금까지 보아 온 자료형의 데이터(값)와 같은 것을 의미하는 말이다(객체에 대해서는 05-1절에서 자세하게 공부한다).

```
a = [1, 2, 3]
```

만약 위 코드처럼 `a = [1, 2, 3]`이라고 하면 `[1, 2, 3]` 값을 가지는 리스트 데이터(객체)가 자동으로 메모리에 생성되고 변수 `a`는 `[1, 2, 3]` 리스트가 저장된 메모리의 주소를 가리키게 된다.

`a` 변수가 가리키는 메모리의 주소는 다음과 같이 확인할 수 있다.

```
>>> a = [1, 2, 3]
>>> id(a)
4303029896
```

`id`는 변수가 가리키고 있는 객체의 주소 값을 리턴하는 파이썬의 내장 함수이다. 즉, 여기에서 필자가 만든 변수 `a`가 가리키는 `[1, 2, 3]` 리스트의 주소 값은 `4303029896`이라는 것을 알 수 있다.



## 8 - 3 리스트를 복사하고자 할 때

이번에는 리스트 자료형에서 가장 혼동하기 쉬운 ‘복사’에 대해 설명한다. 다음 예를 통해 알아보자.

```
>>> a = [1, 2, 3]
>>> b = a
```

b 변수에 a 변수를 대입하면 어떻게 될까? b와 a는 같은 걸까, 다른 걸까? 결론부터 말하면 b는 a와 완전히 동일하다고 할 수 있다. 다만 [1, 2, 3]이라는 리스트 객체를 참조하는 변수가 a 변수 1개에서 b 변수가 추가되어 2개로 늘어났다는 차이만 있을 뿐이다.

id 함수를 사용하면 이러한 사실을 확인할 수 있다.

```
>>> id(a)
4303029896
>>> id(b)
4303029896
```

id(a)의 값이 id(b)의 값과 동일하다는 것을 확인할 수 있다. 즉, a가 가리키는 대상과 b가 가리키는 대상이 동일하다는 것을 알 수 있다.



## 8 - 3 리스트를 복사하고자 할 때

동일한 객체를 가리키고 있는지에 대해서 판단하는 파이썬 명령어 `is`를 다음과 같이 실행해도 역시 참을 리턴해준다.

```
>>> a is b  
True
```

다음 예

```
>>> a[1] = 5  
>>> a  
[1, 5, 3]  
>>> b  
[1, 5, 3]
```

`a` 리스트의 두 번째 요소를 값 4로 바꾸었더니 `a`만 바뀌는 것이 아니라 `b`도 똑같이 바뀌었다. 그 이유는 앞에서 살펴본 것처럼 `a`, `b` 모두 동일한 리스트를 가리키고 있기 때문이다.

그렇다면 `b` 변수를 생성할 때 `a` 변수의 값을 가져오면서 `a`와는 다른 주소를 가리키도록 만들 수는 없을까? 다음 2가지 방법이 있다.



## 8 - 3 - 1 [:] 이용하기

첫 번째 방법은 다음과 같이 리스트 전체를 가리키는 `[:]`을 사용해서 복사하는 것이다.

```
>>> a = [1, 2, 4]
>>> b = a[:]
>>> a[1] = 5
>>> a
[1, 5, 4]
>>> b
[1, 2, 4]
```

위 예에서 볼 수 있듯이 **a** 리스트 값을 바꾸더라도 **b** 리스트에는 아무런 영향이 없다.



## 8 - 3 - 2 copy 모듈 이용하기

두 번째 방법은 `copy` 모듈을 사용하는 것이다. 다음 예를 보면 `from copy import copy`라는 처음 보는 형태의 문장이 나오는데, 이것은 뒤에 설명할 파이썬 모듈 부분에서 자세히 다룬다. 여기에서는 단순히 `copy` 함수를 쓰기 위해서 사용하는 것이라고만 알아 두자.

```
>>> from copy import copy  
>>> a = [1, 2, 3]  
>>> b = copy(a)  
>>> b is a  
False
```

위 예에서 `b = copy(a)`는 `b = a[:]`과 동일하다.

위 예에서 `b is a`가 `False`를 리턴하므로 `b`와 `a`가 가리키는 객체는 서로 다르다는 것을 알 수 있다.



## 8 - 4 변수를 만드는 여러가지 방법

다음과 같이 튜플로 a, b에 값을 대입할 수 있다.

```
>>> a, b = ('python', 'life')
```

이 방법은 다음 예문과 완전히 동일하다.

```
>>> (a, b) = 'python', 'life'
```

튜플 부분에서도 언급했지만, 튜플은 괄호를 생략해도 된다.

다음처럼 리스트로 변수를 만들 수도 있다.

```
>>> [a, b] = ['python', 'life']
```



## 8 - 4 변수를 만드는 여러가지 방법

또한 여러 개의 변수에 같은 값을 대입할 수도 있다.

```
>>> a = b = 'python'
```

파이썬에서는 위 방법을 사용하여 두 변수의 값을 매우 간단하게 바꿀 수 있다.

```
>>> a = 4
>>> b = 8
>>> a, b = b, a
>>> a
8
>>> b
4
```

처음에 **a**에 값 3, **b**에는 값 5가 대입되어 있었지만 **a, b = b, a** 문장을 수행한 후에는 그 값이 서로 바뀌었다는 것을 확인할 수 있다.



# 3장

## -제어문-

팀장: 박민우 팀원: 김민석, 진예성



# 제어문

1. if문
2. while문
  1. if문과 차이점
  2. while문의 구조와 활용
3. for문

## 1. if是



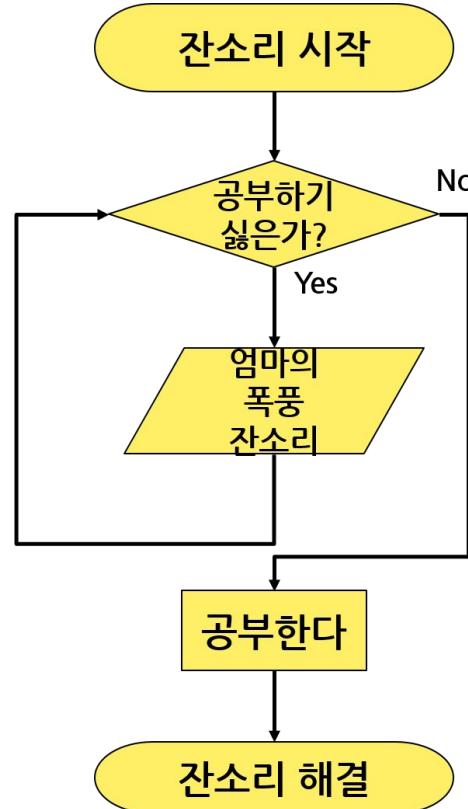
1) if문은 왜 필요할까?

인생은 B와 D 사이의 ( )다.

선택을 위해서는 조건이 늘 필요하다.

# if문은 뭘까?

>>> 프로그래밍에서 조건을 판단하여 해당 조건에 맞는 상황을 수행하는데 쓰는 것이 바로 if 문이다.



# if문은 어떻게 쓰는 것일까?

```
if 조건문:  
    수행할_문장1  
    수행할_문장2  
    ...  
  
else:  
    수행할_문장A  
    수행할_문장B  
    ...
```

# 적용

```
1 mom = True
2
3 if mom:
4     print("공부")
5 else:
6     print("잔소리 ")
7
```

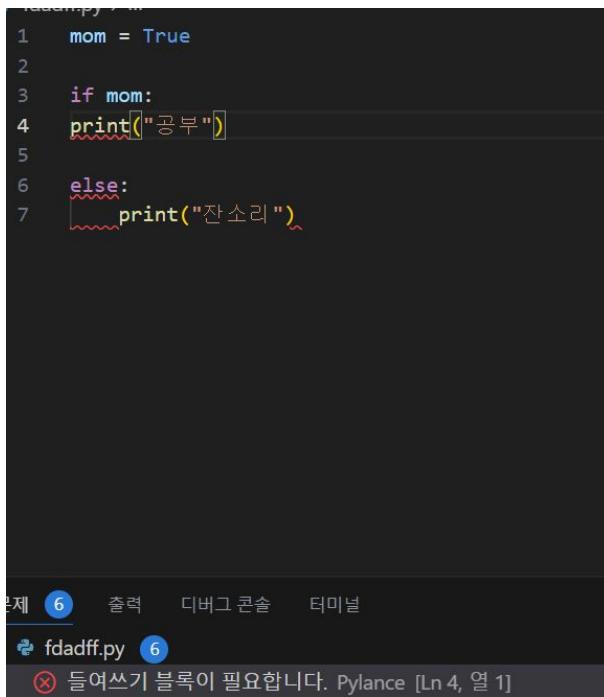
문제 출력 디버그 콘솔 터미널

PS C:\Users\libe0\OneDrive\바탕 화면\pyttt> & C:/Users/libe0/AppData/Local/OneDrive/바탕 화면 /pyttt/fdadff.py"

공부

PS C:\Users\libe0\OneDrive\바탕 화면\pyttt>

# 주의할점

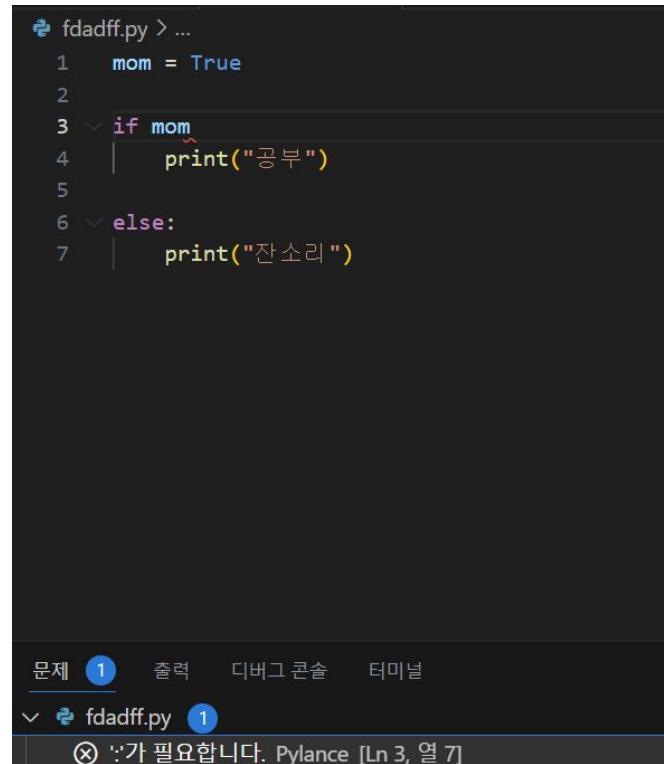


```
fdadff.py > ...
1 mom = True
2
3 if mom:
4     print("공부")
5
6 else:
7     print("잔소리")
```

문제 6 출력 디버그 콘솔 터미널

fdadff.py 6

☒ 들여쓰기 블록이 필요합니다. Pylance [Ln 4, 열 1]



```
fdadff.py > ...
1 mom = True
2
3 if mom
4     print("공부")
5
6 else:
7     print("잔소리")
```

문제 1 출력 디버그 콘솔 터미널

fdadff.py 1

☒ ::가 필요합니다. Pylance [Ln 3, 열 7]

# 조건문은 무엇인가?

if 조건문에서 '조건문'이란 참과 거짓을 판단하는 문장을 말한다.

앞에서 살펴본 예제에서 조건문은 mom이 된다.

```
1 mom = True  
2  
3 v if mom:
```

# 비교 연산자

비교연산자	설명	
$x < y$	$x$ 가 $y$ 보다 작다.	<pre>&gt;&gt;&gt; x = 3 &gt;&gt;&gt; y = 2 &gt;&gt;&gt; x &gt; y True &gt;&gt;&gt;</pre>
$x > y$	$x$ 가 $y$ 보다 크다.	<p><math>x</math>에 3, <math>y</math>에 2를 대입한 후 <math>x &gt; y</math>라는 조건문을 수행하면 True를 리턴한다. <math>x &gt; y</math> 조건문이 참이기 때문이다.</p> <pre>&gt;&gt;&gt; x &lt; y False</pre>
$x == y$	$x$ 와 $y$ 가 같다.	<p>위 조건문은 거짓이기 때문에 False를 리턴한다.</p> <pre>&gt;&gt;&gt; x == y False</pre>
$x != y$	$x$ 와 $y$ 가 같지 않다.	
$x >= y$	$x$ 가 $y$ 보다 크거나 같다.	
$x <= y$	$x$ 가 $y$ 보다 작거나 같다.	

# 적용

```
pyttt.py
1 mom = 100000
2 me = 10000
3
4 if mom > me:
5     print("공부")
6
7 else:
8     print("잔소리")
```

문제 출력 디버그 콘솔 터미널

```
PS C:\Users\libe0\OneDrive\바탕 화면\pyttt> & C:/Users/libe0//OneDrive/바탕 화면/pyttt/fdadff.py"
    공부
PS C:\Users\libe0\OneDrive\바탕 화면\pyttt>
```

## 또 다른 연산자 And, or, not

연산자	설명
$x \text{ or } y$	$x$ 와 $y$ 둘 중 하나만 참이어도 참이다.
$x \text{ and } y$	$x$ 와 $y$ 모두 참이어야 참이다.
$\text{not } x$	$x$ 가 거짓이면 참이다.

# 적용 (or)

```
fdadff.py > ...
1 mom = 100000
2 me = 10000
3 momsay = True
4
5 if mom > me or momsay:
6     print("공부")
7
8 else:
9     print("잔소리")
```

문제 출력 디버그 콘솔 터미널

```
PS C:\Users\libe0\OneDrive\바탕 화면\pyttt> & C:/User
/OneDrive/바탕 화면/pyttt/fdadff.py
공부
PS C:\Users\libe0\OneDrive\바탕 화면\pyttt>
```

# in, not in

in	not in
x in 리스트	x not in 리스트
x in 튜플	x not in 튜플
x in 문자열	x not in 문자열

```
>>> 1 in [1, 2, 3]
```

True

```
>>> 1 not in [1, 2, 3]
```

False

영어 뜻이 ~안에 라는 것을 생각해보면 쉽게 이해할 수 있다. 최근 배운 명제와 비슷하다.

# 적용(in)

```
fdadff.py > ...
1 mom = ['momsay', 'attack']
2 if 'momsay' in mom:
3     print("공부")
4
5 else:
6     print("잔소리")
```

문제 출력 디버그 콘솔 터미널

```
PS C:\Users\libe0\OneDrive\바탕 화면\pyttt> &
/OneDrive/바탕 화면/pyttt/fdadff.py"
공부
PS C:\Users\libe0\OneDrive\바탕 화면\pyttt>
```

# pass

아무 행동도 취하지 않는 것. if나 else의 레인지에 pass라고 적으면 아무 행동도 취하지 않는다.

```
1 mom = ['momsay', 'attack']
2 if 'momsay' in mom:
3     pass
4
5 else:
6     print("잔소리 ")
```

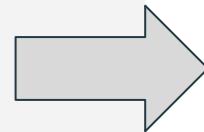
제 출력 디버그 콘솔 터미널

```
S C:\Users\libe0\OneDrive\바탕 화면\pyttt> &
OneDrive/바탕 화면/pyttt/fdadff.py"
S C:\Users\libe0\OneDrive\바탕 화면\pyttt>
```

# 다중 if문을 사용하고 싶다면?

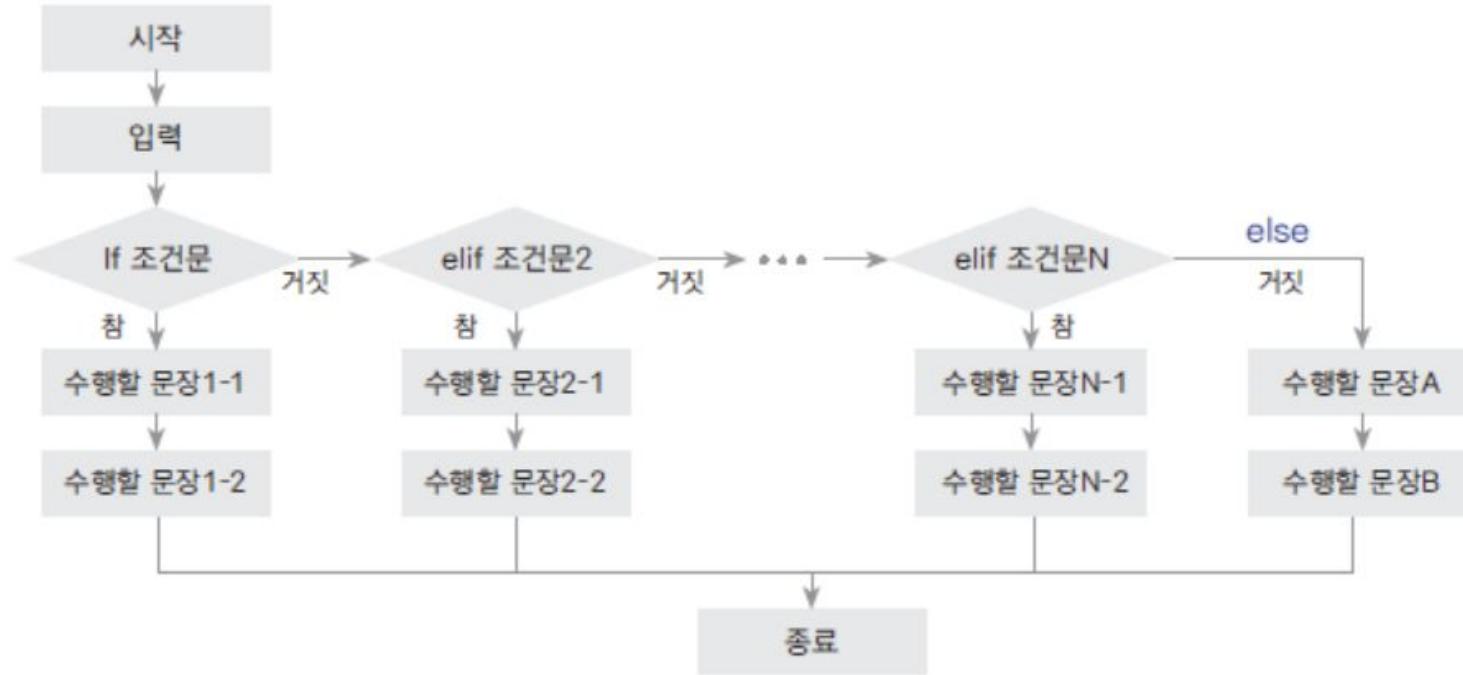
주머니에 돈이 있으면 택시를 타고 가고, 주머니에 돈은 없지만 카드가 있으면 택시를 타고 가고, 돈도 없고 카드도 없으면 걸어가라.

```
>>> pocket = ['paper', 'cellphone']
>>> card = True
>>> if 'money' in pocket:
...     print("택시를 타고가라")
... else:
...     if card:
...         print("택시를 타고가라")
...     else:
...         print("걸어가라")
...
택시를 타고가라
>>>
```



```
>>> pocket = ['paper', 'cellphone']
>>> card = True
>>> if 'money' in pocket:
...     print("택시를 타고가라")
... elif card:
...     print("택시를 타고가라")
... else:
...     print("걸어가라")
...
택시를 타고가라
```

# elif



## 조건부 표현식

```
if score >= 60:  
    message = "success"  
  
else:  
    message = "failure"
```

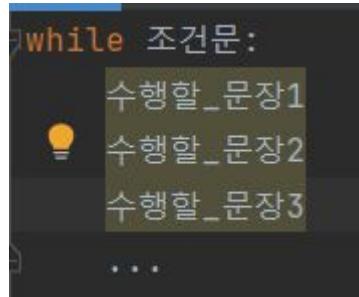


```
message = "success" if score >= 60 else "failure"
```

변수 = 조건문이\_참인\_경우의\_값 if 조건문 else 조건문이\_거짓인\_경우의\_값

조건부 표현식은 가독성과 한줄로 쓸수 있다는 점에서 유용하다.

# while문의 의미



“**while**” 문은 파이썬에서 반복적으로 코드 블록을 실행하는 데 사용되는 제어 구조입니다. 조건이 참(**True**)인 동안에는 반복적으로 코드 블록을 실행합니다. 조건이 거짓(**False**)이 되면 반복이 중단됩니다.

# while문의 구조

```
minwoo = 0
while minwoo < 10:
    minwoo = minwoo +1
    print("이 실행키가 %d번 반복 되었습니다." % minwoo)
    if minwoo == 10:
        print("10번이 끝났습니다.")
```

이 실행키가 1번 반복 되었습니다.  
이 실행키가 2번 반복 되었습니다.  
이 실행키가 3번 반복 되었습니다.  
이 실행키가 4번 반복 되었습니다.  
이 실행키가 5번 반복 되었습니다.  
이 실행키가 6번 반복 되었습니다.  
이 실행키가 7번 반복 되었습니다.  
이 실행키가 8번 반복 되었습니다.  
이 실행키가 9번 반복 되었습니다.  
이 실행키가 10번 반복 되었습니다.  
10번이 끝났습니다.

끝나는 지점을 정의한 뒤 주어진 조건이 만족되면 계속 반복함

`minwoo = minwoo + 1`은 프로그래밍을 할 때 매우 자주 사용하는 기법이다.

`minwoo` 값을 1만큼씩 증가시킬 목적으로 사용하며 `minwoo += 1`처럼 작성해도 된다.

## while문의 입력

```
num = int(input("양의 정수를 입력하세요: "))
factorial = 1
while num > 0:
    factorial *= num
    num -= 1
print("팩토리얼:", factorial)
```

# while문을 끝내는 break

```
products = {"cola": 150, "chips": 100, "candy": 50}
money = int(input("돈을 넣어주세요: "))
while money:
    print("판매 가능한 물건 목록:")
    for item in products:
        print(f"{item.capitalize()} - {products[item]}원")
    choice = input("물건을 선택하세요 (끝내려면 '끝' 입력): ")
    if choice == "끝":
        print("판매를 종료합니다.")
        break
    if choice in products:
        price = products[choice]
        if money >= price:
            print(f"{choice.capitalize()}를 판매합니다.")
            money -= price
            print(f"남은 돈: {money}원")
        else:
            print("돈이 부족합니다. 다른 물건을 선택하세요.")
    else:
        print("잘못된 선택입니다. 다시 선택하세요.")
```

break = while문을 강제로 종료

### 3. for 문

리스트 안의 모든 원소를 수행할 문장에  
대입할 때 까지 작업을 계속하는 반복문.

**for** 변수 **in** 리스트(또는 튜플, 문자열):

수행할\_문장1

수행할\_문장2

...

## for문- 목표



$$0! = 1$$

$$1! = 1$$

$$2! = 1 \times 2 = 2$$

$$3! = 1 \times 2 \times 3 = 6$$

$$4! = 1 \times 2 \times 3 \times 4 = 24$$

$$5! = 1 \times 2 \times 3 \times 4 \times 5 = 120$$

# for문- 전형적인 for문

```
#리스트 예제  
ex1=[ '1번 과자','2번 사탕','3번 초콜릿','4번 아이스크림' ]
```

```
#리스트 안의 요소 출력  
for i in ex1:  
    print(i)
```

```
>>> 1번 과자  
2번 사탕  
3번 초콜릿  
4번 아이스크림
```

# 다양한 for 문의 사용

```
#리스트에 튜플 활용  
ex2=[('1번','과자'),('2번','사탕'),('3번','초콜릿'),('4번','아이스크림')]  
  
#리스트 안의 튜플 출력  
a=0  
b=0  
for i in ex2:  
    print(ex2[a][b]+ex2[a][b+1])  
    a+=1
```

```
>>> 1번 과자  
2번 사탕  
3번 초콜릿  
4번 아이스크림
```

# for 문의 응용

```
snacks=[1000,1500,2000,2500]

#간식의 가격이 비싸지, 적당한지, 싼지 알려주기
number=0
a=0
b=1
for snack in snacks:
    number+=1
    if snack >2000:
        print(f'{number}번 {ex2[a][b]} 은/는 비싸다.')
        a+=1

    elif snack==2000:
        print(f'{number}번 {ex2[a][b]} 은/는 적당하다.')
        a+=1

    elif snack <2000:
        print(f'{number}번 {ex2[a][b]} 은/는 싼다.')
        a+=1
```

>>> 1번 과자 은/는 싸다.

2번 과자 은/는 싸다.

3번 과자 은/는 적당하다.

4번 과자 은/는 비싸다.

# for 문과 continue 문

```
#과자의 종류와 가격
ex2=[('1번','과자'),('2번','사탕'),('3번','초콜릿'),('4번','아이스크림')]
snacks=[1000,1500,2000,2500]

#적당한 간식만 알려주기(continue 응용)
number=0
a=0
b=1
for snack in snacks:
    number+=1
    if snack !=2000:
        continue
    print(f'{number}번 {ex2[a][b]} 은/는 적당하다')
    a+=1
```

continue

: 다음 차례의 반복문으로  
넘어가기

>>>3번 과자 은/는 적당하다

# for 문과 함께 자주 사용하는 range 함수

```
#range 함수  
a=range(10)  
print(a)
```

```
#range 응용  
add = 0  
for i in range(1, 11):  
    add = add + i  
print(add)
```

range(a,b): a부터 b미만의 수

```
>>>range(0, 10)  
55
```

# for와 range를 이용한 팩토리얼 계산기

```
#range와 for문 활용
factorial=1
a=int(input('몇 팩토리얼을 구하고 싶은지 적으시오 \n값: '))
for i in range(1,a+1):
    factorial=factorial*i
print(f'\n{a} 팩토리얼은 {factorial} 입니다.')
```

>>> 몇 팩토리얼을 구하고  
싶은지 적으시오 (n입력)  
n 팩토리얼은 ~입니다.

## 리스트 컴프리헨션 사용하기

```
a = [1,2,3,4]
result = []
for num in a:
    result.append(num*3)
print(result)
```

>>

```
a = [1,2,3,4]
result = [num*3 for num in a]
print(result)
```

>>> [3,6,9,12]

리스트 컴프리헨션: for문+리스트

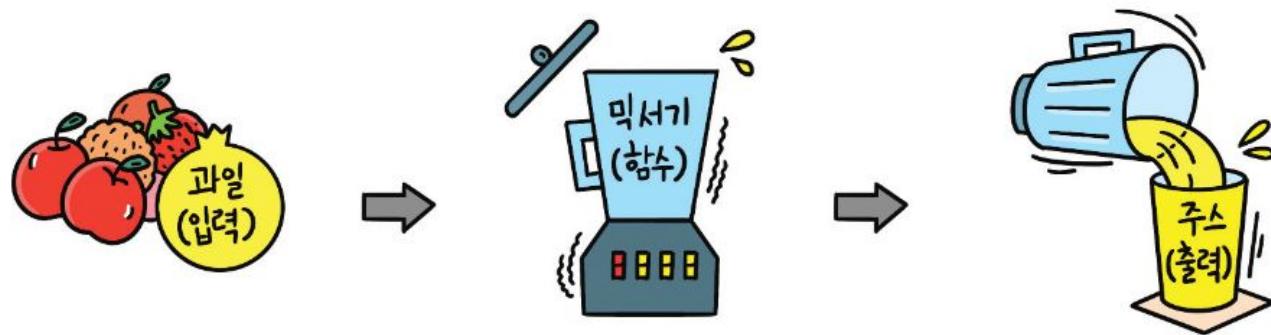


# 4장

## -입출력-

팀장: 이경민 팀원: 하정원, 안상호

## 4.1 함수



## 4.1.1 파이썬 함수의 구조

```
def 함수_이름(매개변수):
    수행할_문장1
    수행할_문장2
    ...
```

`def`는 함수를 만들 때 사용하는 예약어이며, 함수 이름 뒤 괄호 안의 매개변수는 이 함수에 입력으로 전달되는 값을 받는 변수이다. 이렇게 함수를 정의한 후 `if`, `while`, `for` 문 등과 마찬가지로 함수에서 수행할 문장을 입력한다.

예)

```
>>> def add(a, b):
...     return a+b
...
>>>
```

## 4.1.2 함수의 형태

일반적인 함수

입력값이 없는 함수

리턴값이 없는 함수

입력값도 리턴값도 없는 함수

## 4.1.2 입력값이 없는 함수

예)

```
>>> def say():
...     return 'Hi'
```

```
>>> a = say()
>>> print(a)
Hi
```

이 함수는 입력값은 없지만, 리턴값으로 "Hi"라는 문자열을 리턴한다. 즉, `a = say()`처럼 작성하면 `a`에 "Hi"라는 문자열이 대입되는 것이다.

## 4.1.2 리턴값이 없는 함수

```
>>> def add(a, b):
...     print("%d, %d의 합은 %d입니다." % (a, b, a+b))
```

```
>>> add(3, 4)
3, 4의 합은 7입니다.
```

3, 4의 합은 7입니다.'라는 문장을 출력했는데 왜 리턴값이 없다는 것인지 의아하게 생각할 수도 있다. 하지만 `print` 문은 함수의 구성 요소 중 하나인 '수행할\_문장'에 해당하는 부분일 뿐이다. 리턴값은 당연히 없다. 리턴값은 오직 `return` 명령어로만 돌려받을 수 있다.

## 4.1.2 입력값도 리턴값도 없는 함수

```
>>> def say():
...     print('Hi')
```

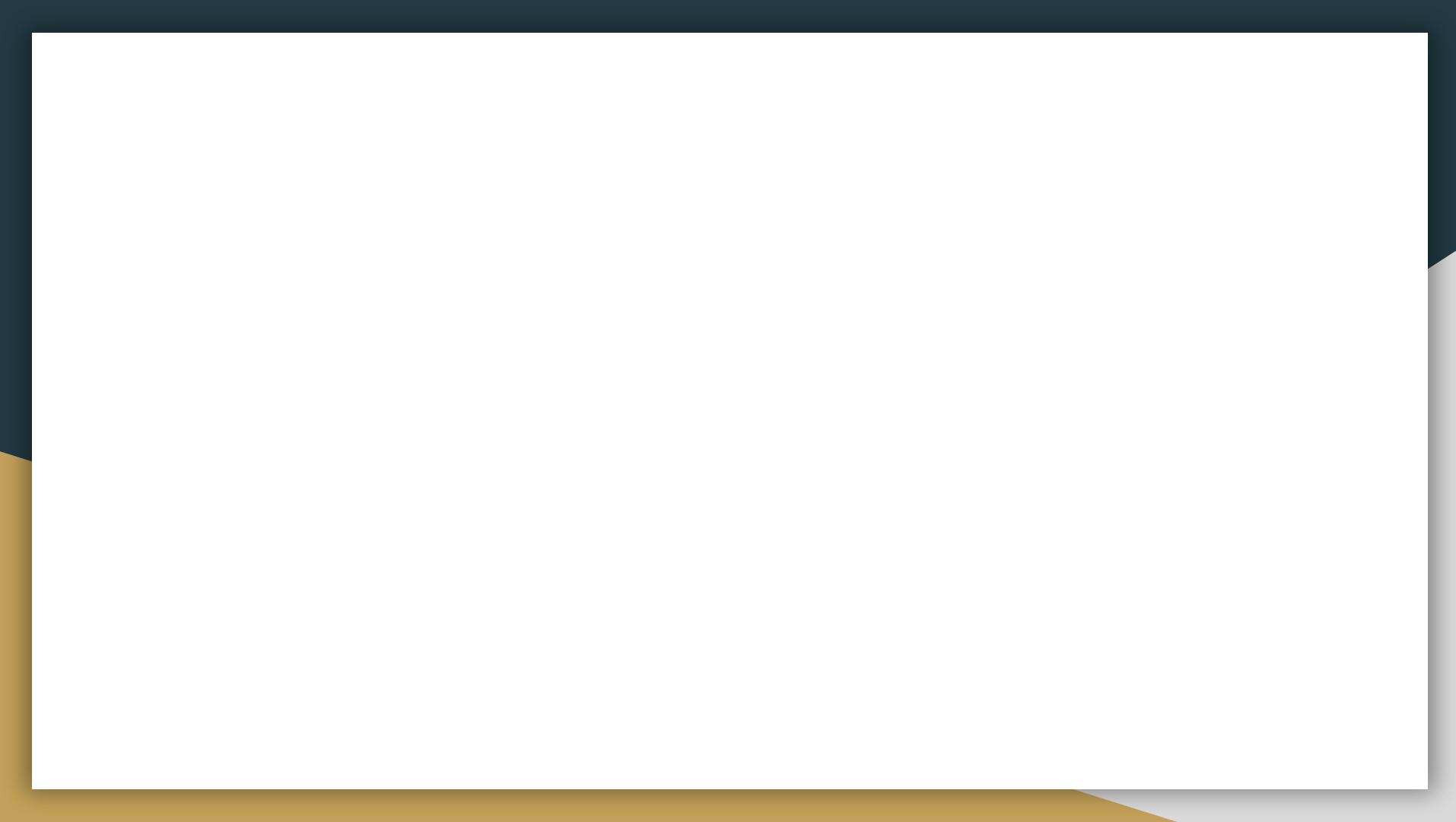
```
>>> say()
Hi
```

### 4.1.3 매개변수 지정하여 호출하기

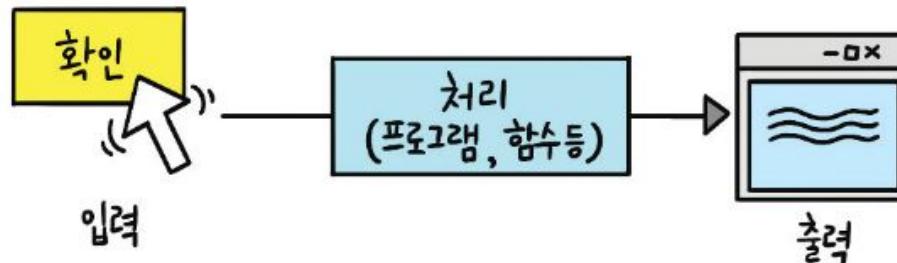
```
>>> def sub(a, b): *매개변수
...     return a - b

>>> result = sub(a=7, b=3) # a에 7, b에 3을 전달
>>> print(result)
4

>>> result = sub(b=5, a=3) # b에 5, a에 3을 전달
>>> print(result)
-2
```



## 2-1 사용자 입출력



사용자 입출력은 컴퓨터 프로그래밍에서 프로그램과 사용자 간의 정보 교환을 의미한다. 일반적으로 프로그램은 사용자로부터 입력을 받아 처리하고, 그 결과를 사용자에게 출력으로 제공한다.

## 2-2 input

1. `input()` : 사용자로부터 입력을 받는 함수
2. `input`은 사용자가 키보드로 입력한 모든 것을 문자열로 저장한다

## 2-3 프롬프트를 띄워 사용자 입력받기

사용자에게 질문을 보여주고 싶을 때

```
password = input("암호를 입력하시오:")  
print(password)
```



```
암호를 입력하시오:12  
12
```

input은 입력되는 모든 것을 문자열로 취급하기 때문에  
**password**는 숫자가 아닌 문자열이라는 것에 주의하자.

## 2-4 print

print : 많은 프로그래밍 언어에서 사용되는 함수로, 프로그램이 결과나 정보를 화면에 출력하는데 사용된다.

print문의 예시

```
number = input("숫자를 입력하시오: ")
print(number)
숫자를 입력하시오: 112
112
```

## 2-5 큰따옴표를 둘러싸인 문자열은 + 연산과 동일하다.

```
print("a"+"b"+"c") → abc
```

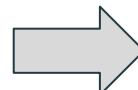
```
print("a" "b" "c") → abc
```

완전히 동일한 형식

## 2-6 문자열 띄어쓰기는 쉼표로 한다

```
print("a" , "b" , "c")
```

쉼표 (,)를 사용하면 문자열을 띄어  
쓸 수 있다



```
a b c
```

## 2-7 한 줄에 결과값 출력하기

```
>>> for i in range(10):
...     print(i, end='(\n)')
...
0 1 2 3 4 5 6 7 8 9 >>>
```

end 매개변수의 초기값은  
줄 바꿈 (\n) 이다

## 4-3 파일 읽고 쓰기

- 1.파일 생성하기
- 2.파일을 쓰기 모드로 열어 내용 쓰기
- 3.파일을 읽는 여러 가지 방법
- 4.파일에 새로운 내용 추가하기
- 5.with 문과 함께 사용하기

# 파일 생성하기

```
# newfile.py  
f = open("새파일.txt", 'w')  
f.close()
```

파일열기모드	설명
r	읽기 모드: 파일을 읽기만 할 때 사용한다.
w	쓰기 모드: 파일에 내용을 쓸 때 사용한다.
a	추가 모드: 파일의 마지막에 새로운 내용을 추가할 때 사용한다.

파일을 'C:/doit' 디렉터리에 생성하고 싶다면 다음과 같이 작성해야 한다.

```
# newfile2.py
f = open("C:/doit/새파일.txt", 'w')
f.close()
```

파일을 쓰기 모드로 열어 내용 쓰기

파이썬에서는 sys 모듈을 사용하여 프로그램에 인수를 전달할 수 있다.

sys 모듈을 사용하려면 다음 예의 import sys처럼 import 명령어를 사용해야 한다.

```
파이썬> asdf.py > ...
```

```
1 # sys1.py
2 import sys
3
4 args = sys.argv[1:]
5 for i in args:
6     print(i)
```

위는 프로그램 실행 시 전달받은 인수를 for 문을 사용해 차례대로 하나씩 출력하는 예이다. sys 모듈의 argv는 프로그램 실행 시 전달된 인수를 의미한다. 즉, 다음과 같이 입력했다면 argv[0]은 파일 이름 sys1.py가 되고 argv[1]부터는 뒤에 따라오는 인수가 차례대로 argv의 요소가 된다.

위는 프로그램 실행 시 전달받은 인수를 for 문을 사용해 차례대로 하나씩 출력하는 예이다. sys 모듈의 argv는 프로그램 실행 시 전달된 인수를 의미한다. 즉, 다음과 같이 입력했다면 argv[0]은 파일 이름 sys1.py가 되고 argv[1]부터는 뒤에 따라오는 인수가 차례대로 argv의 요소가 된다.



이 프로그램을 **C:\doit** 디렉터리에 저장한 후 인수를 전달하여 실행하면 다음과 같은 결과를 볼 수 있다.

명령 프롬프트를 열고 **cd C:\doit** 명령을 실행하여 **C:\doit** 디렉터리로 이동한 후 다음 명령을 실행해 보자.

```
C:\doit>python sys1.py aaa bbb ccc
aaa
bbb
ccc
```

위 예를 응용하여 다음과 같은 간단한 프로그램을 만들어 보자.

```
# sys2.py
import sys
args = sys.argv[1:]
for i in args:
    print(i.upper(), end=' ')
```

Copy

문자열 관련 함수인 `upper()`를 사용하여 프로그램 실행 시 전달된 인수를 모두 대문자로 바꾸어 주는 간단한 프로그램이다. 명령 프롬프트 창에서 다음과 같이 실행해 보자.

`sys2.py` 파일이 `C:\doit` 디렉터리 안에 있어야만 한다.

```
C:\doit>python sys2.py life is too short, you need python
```

출력 결과는 다음과 같다.

```
LIFE IS TOO SHORT, YOU NEED PYTHON
```



# 5장

# -파이썬 날개달기-

팀장: 김민호 팀원: 안재민, 장윤호



# Python 클래스

1. 클래스는 왜 필요한가?
2. 클래스와 객체
3. 사칙 연산 클래스 만들기
  1. 클래스를 어떻게 만들지 먼저 구상하기
  2. 클래스 구조 만들기
  3. 객체에 연산할 숫자 지정하기
  4. 더하기 기능 만들기
  5. 곱하기, 빼기, 나누기 기능 만들기
4. 생성자
5. 클래스의 상속
6. 메서드 오버라이딩
7. 클래스변수

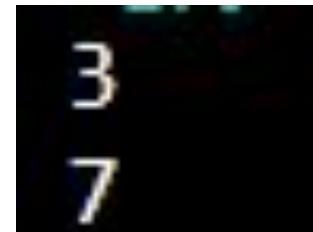


# 1. 클래스는 왜 필요한가? - 계산기프로그램 만들어보기

```
# calculator.py
result = 0

def add(num):
    global result
    result += num # 결과값(result)에 입력값(num) 더하기
    return result # 결과값 리턴

print(add(3))
print(add(4))
```



```
3
7
```

입력값을 이전에 계산한 결과값에 더한 후 리턴하는 `add` 함수를 위와 같이 작성했다. 이전에 계산한 결과값을 유지하기 위해서 `result` 전역 변수(`global`)를 사용했다. 프로그램을 실행하면 예상한 대로 다음과 같은 결과값이 출력된다.



```
# calculator2.py
result1 = 0
result2 = 0

def add1(num): # 계산기1
    global result1
    result1 += num
    return result1

def add2(num): # 계산기2
    global result2
    result2 += num
    return result2

print(add1(3))
print(add1(4))
print(add2(3))
print(add2(7))
```

```
3
7
3
10
```

똑같은 일을 하는 `add1`과 `add2` 함수를 만들고 각 함수에서 계산한 결괏값을 유지하면서 저장하는 전역 변수 `result1`과 `result2`를 정의했다.

결괏값은 다음과 같이 의도한 대로 출력된다.



```
# calculator3.py
class Calculator:
    def __init__(self):
        self.result = 0

    def add(self, num):
        self.result += num
        return self.result

cal1 = Calculator()
cal2 = Calculator()

print(cal1.add(3))
print(cal1.add(4))
print(cal2.add(3))
print(cal2.add(7))
```

```
3
7
3
10
```

이건 클래스를 배우지 않았을 때 쓸 수 있는 함수이다.

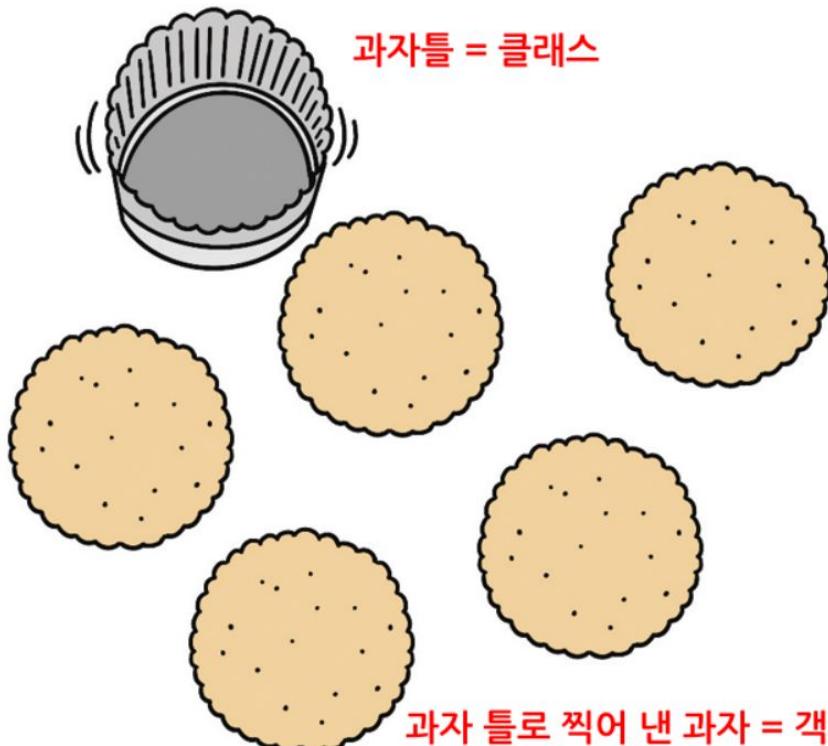


```
class Calculator:  
    def __init__(self):  
        self.result = 0  
  
    def add(self, num):  
        self.result += num  
        return self.result  
  
    def sub(self, num):  
        self.result -= num  
        return self.result
```

Calculator 클래스로 만든  
별개의 계산기 cal1,  
cal2(파이썬에서는 이것을  
'객체'라고 부른다)가 각각의  
역할을 수행한다. 그리고  
계산기의 결괏값 역시 다른  
계산기의 결괏값과 상관없이  
독립적인 값을 유지한다.  
이렇게 클래스를 사용하면  
계산기 대수가 늘어나도  
객체를 생성하면 되므로  
함수만 사용할 때보다  
간단하게 프로그램을 작성할  
수 있다. 빼기 기능을 더하고  
싶다면 Calculator 클래스에  
다음과 같이 빼기 기능을 가진  
함수를 추가하면 된다.



## 2. 클래스와 객체



```
class Cookie:  
    |     pass  
  
a = Cookie()  
b = Cookie()
```

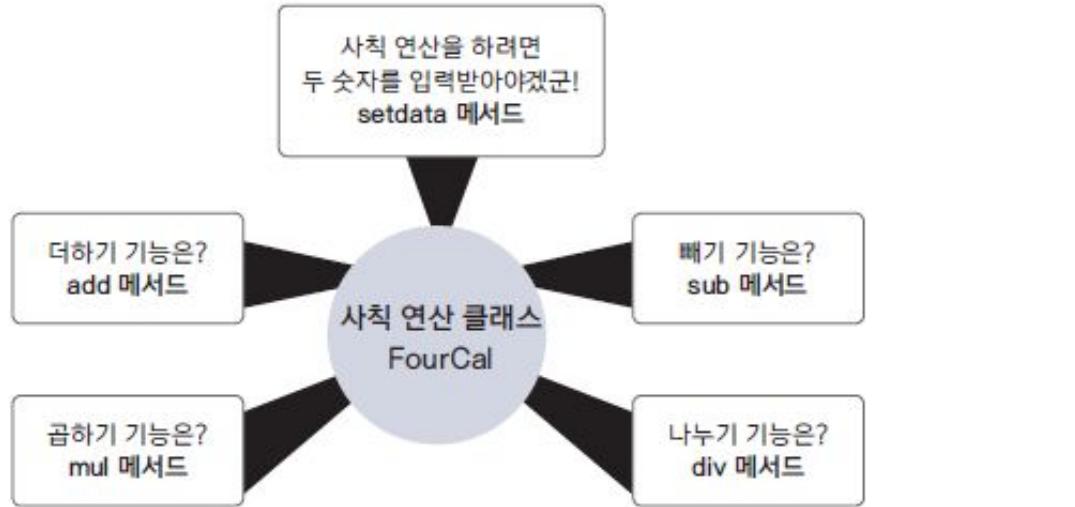
위에서 작성한 Cookie 클래스는 아무런 기능도 가지고 있지 않은 껍질뿐인 클래스이다. 하지만 이렇게 껍질뿐인 클래스도 객체를 생성하는 기능이 있다. ‘과자 틀’로 ‘과자’를 만드는 것처럼 말이다.



### 3. 사칙연산 클래스 만들기

#### 3-1. 클래스를 어떻게 만들지 먼저 구상하기

어떤 클래스를 만들지 대충 그림을 그려 보자.





## 3-2. 클래스 구조 만들기

```
class Fourcal:  
    pass
```

```
>>> a = FourCal()  
>>> type(a)  
<class '__main__.FourCal'>
```

pass는 아무것도 수행하지 않는 문법으로, 임시의 코드를 사용할 때 주로 쓰인다.



### 3-3. 객체에 연산할 숫자 지정하기

```
class FourCal:  
    def setdata(self, first, second):  
        self.first = first  
        self.second = second
```

객체

```
a = FourCal()  
a.setdata(4,2)
```



```
a.setdata(4, 2)
```

```
def setdata(self, first, second):  
    self.first = first  
    self.second = second
```



## 3-4. 더하기 기능 만들기

```
class Fourcal:  
    def setdata(self, first, second):  
        self.first = first  
        self.second = second  
    def add(self):  
        result = self.first + self.second  
        return result
```

```
a = Fourcal()  
a.setdata(4,2)  
print(a.add())
```

위와 같이 호출하면 앞에서 살펴보았듯이 a 객체의 first, second 객체변수에는 각각 값 4와 2가 저장될 것이다.

이제 add 메서드를 호출해 보자.



## 3-5. 곱하기, 빼기, 나누기 기능 만들기

```
def mul(self):
    result = self.first * self.second
    return result
def sub(self):
    result = self.first - self.second
    return result
def div(self):
    result = self.first / self.second
    return result
```

```
a = Fourcal()
a.setdata(4,2)
print(a.add())
print(a.mul())
print(a.sub())
print(a.div())
```

```
6
8
2
2.0
```



## 4. 생성자

```
class Fourcal:  
    def __init__(self, first, second):  
        self.first = first  
        self.second = second  
    def setdata(self, first, second):  
        self.first = first  
        self.second = second  
    def add(self):  
        result = self.first + self.second  
        return result  
    def mul(self):  
        result = self.first * self.second  
        return result  
    def sub(self):  
        result = self.first - self.second  
        return result  
    def div(self):  
        result = self.first / self.second  
        return result  
  
a = Fourcal()
```

```
def __init__(self, first, second):  
    self.first = first  
    self.second = second
```

```
a = Fourcal(4,2)
```

```
Traceback (most recent call last):  
  File "d:\coding\study\ppt.py", line 87, in <module>  
    a = Fourcal()  
TypeError: Fourcal.__init__() missing 2 required positional arguments: 'first' and 'second'
```

FourCal 클래스의 인스턴스 a에 setdata 메서드를 수행하지 않고 add 메서드를 먼저 수행하면 'AttributeError: 'FourCal' object has no attribute 'first'' 오류가 발생한다. setdata 메서드를 수행해야 객체 a의 객체변수 first와 second가 생성되기 때문이다. 이렇게 객체에 first, second와 같은 초기값을 설정해야 할 필요가 있을 때는 setdata와 같은 메서드를 호출하여 초기값을 설정하기보다 생성자를 구현하는 것이 안전한 방법이다.



## 5. 클래스의 상속

```
class MoreFourcal(Fourcal):
    pass

b = MoreFourcal(4,2)

print(b.add())
print(b.mul())
print(b.sub())
print(b.div())
```

6  
8  
2  
2

```
class MoreFourcal(Fourcal):
    def pow(self):
        result = self.first ** self.second
        return result

b = MoreFourcal(4,2)

print(b.pow())
```

16



## 6. 메서드 오버라이딩

```
class SafeFourCal(FourCal):
    def div(self):
        if self.second == 0: # 나누는 값이 0인 경우 0을 리턴하도록 수정
            return 0
        else:
            return self.first / self.second
```



## 7. 클래스변수

```
class Family:  
    lastname = "김"  
  
Family.lastname
```



클래스 패밀리에 “박”을 넣으면 클래스로 만든 instname  
값도 모두 바뀐다. 즉, 클래스의 변수는 모두 공유한다.



# Python 모듈

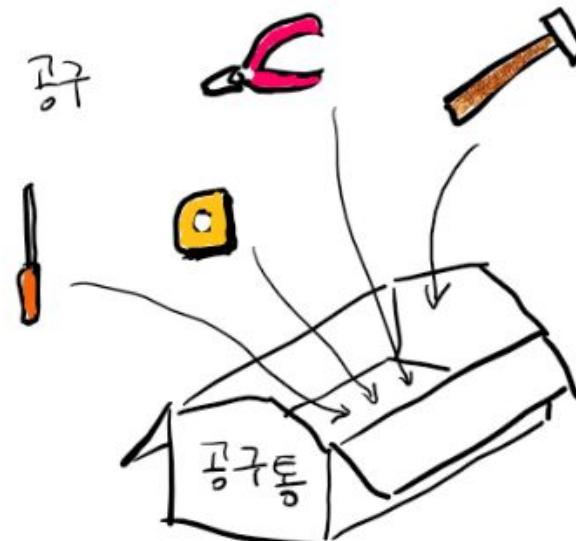
- 1.모듈 만들기
- 2.모듈 불러오기
- 3.if \_\_name\_\_ == "\_\_main\_\_":의 의미
- 4.클래스나 변수 등을 포함한 모듈
- 5.다른 파일에서 모듈 불러오기



# 0.모듈

- 함수, 변수, 클래스를 모아 놓은 파이썬 파일
- 다른 파이썬 프로그램에서 불러와 사용할 수 있도록 만든 파이썬 파일

```
Python 3.6.1 Shell
File Edit Shell Debug Options Window ...
>>> import random
>>>
>>> random.choice([1, 2, 3, 4, 5])
4
```



# 1. 모듈 만들기 - 사칙연산 모듈 만들기

```
# mod1.py
def add(a, b):
    return a + b

def sub(a, b):
    return a-b

def mul(a, b):
    return a * b

def div(a, b):
    return a / b
```

## 2. 모듈 불러오기

```
import 모듈_이름
```

```
# 5-2  
import mod1  
  
print(mod1.add(3,4))  
print(mod1.mul(3,4))
```

7  
12

```
from 모듈_이름 import 모듈_함수
```

```
# 5-2  
from mod1 import add  
  
print(add(3, 4))
```

7

### 3.if \_\_name\_\_ == "\_\_main\_\_":의 의미

```
# mod1.py
def add(a, b):
    return a + b

def sub(a, b):
    return a-b

def mul(a, b):
    return a * b

def div(a, b):
    return a / b

print(add(1, 4))
print(sub(4, 2))
```

5  
2

```
# 5-2
import mod1
mod1.add(3, 4)
mod1.mul(3, 4)
```

5  
2  
7  
12

## 3.if \_\_name\_\_ == "\_\_main\_\_":의 의미

```
# mod1.py
def add(a, b):
    return a + b

def sub(a, b):
    return a-b

def mul(a, b):
    return a * b

def div(a, b):
    return a / b

if __name__ == "__main__":
    print(add(1, 4))
    print(sub(4, 2))
```

5  
2

```
# 5-2
import mod1
print(mod1.add(3,4))
print(mod1.mul(3,4))
```

7  
12

## 4. 클래스나 변수 등을 포함한 모듈

mod2.py > ...

```
1 # mod2.py
2
3 PI = 3.141592 # 변수
4
5 # 클래스
6 class Math:
7     def solv(self, r):
8         return PI * (r ** 2)
9
10 # 함수
11 def add(a, b):
12     return a+b
```

```
# 5-2
import mod2
print(mod2.PI)
```

```
a = mod2.Math()
print(a.solv(2))
print(mod2.add(3,4))
```

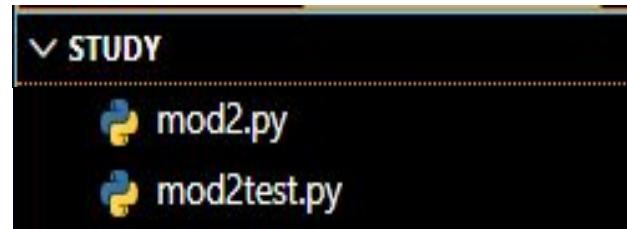
3.141592  
12.566368  
7

# 5. 다른 파일에서 모듈 불러오기

파일이 동일한 디렉터리(C:\doit)에 있는 경우

```
mod2.py > ...
1 # mod2.py
2
3 PI = 3.141592 # 변수
4
5 # 클래스
6 class Math:
7     def solv(self, r):
8         return PI * (r ** 2)
9
10 # 함수
11 def add(a, b):
12     return a+b
```

```
mod2test.py > ...
1 import mod2
2
3 a = mod2.add(3,4)
4 print(a)
5
```



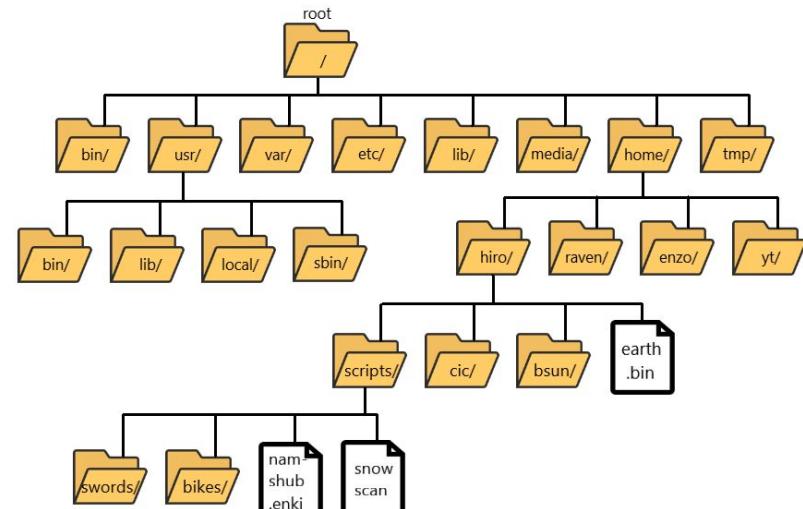


# Python 패키지

1. 패키지 만들기
2. 패키지 안의 함수 실행하기
3. `__init__.py`의 용도
  1. 패키지 변수 및 함수 정의
  2. 패키지 내 모듈을 미리 import
  3. 패키지 초기화
  4. `__all__`
4. relative 패키지



# O. 파일기자





# 1. 패키지 만들기

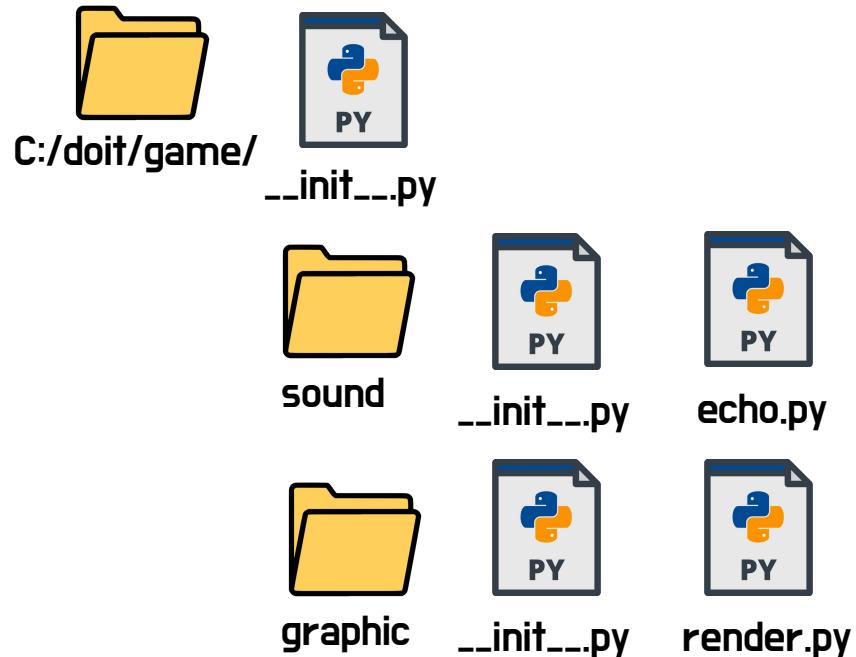
C:/doit/game/\_\_init\_\_.py

C:/doit/game/sound/\_\_init\_\_.py

C:/doit/game/sound/echo.py

C:/doit/game/graphic/\_\_init\_\_.py

C:/doit/game/graphic/render.py



# 1. 테스트 만들기

```
1 # render.py  
2 def render_test():  
3     print("render")
```

```
1 # echo.py  
2 def echo_test():  
3     print("echo")
```



## 2. 패키지 안의 함수 실행하기

<pre>명령 프롬프트 - python Microsoft Windows [Version 10.0.19045.3324] (c) Microsoft Corporation. All rights reserved.  C:\Users\cwrqgw&gt;cd.. C:\Users&gt;cd.. C:\&gt;set PYTHONPATH=C:/doit</pre>	<pre>C:\&gt;python Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 15:45:37) Type "help", "copyright", "credits" or "license" for more information &gt;&gt;&gt; from game.sound import echo &gt;&gt;&gt; echo.echo_test() echo &gt;&gt;&gt;</pre>
<pre>C:\&gt;python Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 15:45:37) Type "help", "copyright", "credits" or "license" for more information &gt;&gt;&gt; import game.sound.echo &gt;&gt;&gt; game.sound.echo.echo_test() echo &gt;&gt;&gt;</pre>	<pre>C:\&gt;python Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 15:45:37) Type "help", "copyright", "credits" or "license" for more information &gt;&gt;&gt; from game.sound.echo import echo_test &gt;&gt;&gt; echo_test() echo &gt;&gt;&gt;</pre>



## 3. \_\_init\_\_.py의 용도

\_\_init\_\_.py 파일은 해당 디렉터리가 패키지 일부임을 알려준다.

만약 \_\_init\_\_.py 파일이 없다면 패키지로 인식되지 않는다.

패키지 o

\_\_init\_\_.py

bulabula.py

패키지 x

bulibuli.py

bulabula.py



## 3-1. 패키지 변수 및 함수 정의

패키지의 요소가 아니라 패키지 자체의 수준에서 변수와 함수를 지정할 수 있습니다.

예를 들어 game 파일의 `__init__.py`에 공통변수나 함수를 정의할 수 있습니다.

```
# _init_.py > ...
1 # C:/doit/game/__init__.py
2 VERSION = 3.5
3
4 def print_version_info():
5     print(f"The version of this game is {VERSION}.")
6
```

```
C:\>python
Python 3.11.4 (tags/v3.11.4:d2340ef,
Type "help", "copyright", "credits"
>>> import game
>>> print(game.VERSION)
3.5
>>> game.print_version_info()
The version of this game is 3.5.
>>>
```



## 3-2. 패키지 내 모듈을 미리 import

패키지 `__init__.py`에서 미리 import를 해두면 간편하게 코드를 사용할 수 있음

```
1 # C:/doit/game/__init__.py
2 from .graphic.render import render_test
3
4 VERSION = 3.5
5
```

```
Python 3.11.4 (tags/v3.11.4:d23efb1, Jun  5 2023, 15:23:26)
Type "help", "copyright", "credits" or "license" for more information
>>> import game
>>> game.render_test()
render
>>>
```



## 3-3. 패키지 초기화

패키지를 처음 import 할 때 초기화 코드가 실행이 됩니다.

다시 import 한다고 해도 초기화 코드는 실행되지 않습니다.

```
1 # C:/doit/game/__init__.py
2 from .graphic.render import render_test
3
4 VERSION = 3.5
5
6 def print_version_info():
7     print(f"The version of this game is {VERSION}.")
8
9 print("Initializing game ...")
```

```
>>> import game
Initializing game ...
>>> game.render_test()
render
>>> import game
>>>
```



## 3-4. \_\_all\_\_

특정 디렉터리의 모듈을 \*를 사용하여 import할 때는 다음과 같이 해당 디렉터리의 `__init__.py` 파일에 `__all__` 변수를 설정하고 import할 수 있는 모듈을 정의해 주어야 한다.

\*를 사용하여 import할 경우, 이곳에 정의된 모듈만 import된다는 의미이다.

```
>>> from game.sound import *
Initializing game ...
Initializing game ...
>>> echo.echo_test()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'echo' is not defined
>>>
```

```
10    __all__ = ['echo']
>>> from game.sound import *
Initializing game ...
Initializing game ...
>>> echo.echo_test()
echo
>>>
```



## 4.relative 패키지

만약 graphic 디렉터리의 render.py 모듈에서 sound 디렉터리의 echo.py 모듈을 사용하고 싶다면 다음과 같이 render.py를 수정하면 가능합니다.

```
1 # render.py
2 from game.sound.echo import echo_test
3 def render_test():
4     print("render")
5     echo_test()
```

```
>>> from game.graphic.render import render_test
Initializing game ...
Initializing game ...
>>> render_test()
render
echo
>>>
```



# Python 예외 처리

1. 오류는 언제 발생하는가?
2. 오류 예외 처리 기법
  1. try-except 문
  2. try-finally 문
  3. 여러 개의 오류 처리하기
  4. try-else 문
3. 오류 회피하기
4. 오류 일부러 발생시키기
5. 예외 만들기



# 1. 오류는 언제 발생하는가?

**ValueError**

**IndexError**

**SyntaxError**

**NameError**

**ZeroDivisionError**

**FileNotFoundException**

**TypeError**

**AttributeError**

**KeyError**

...

...

함수의 인자로 사용되는 값이 자료형은 맞지만 처리할 수 없는 경우

Python의 range() 함수에서 잘못된 정수를 사용했을 경우

파이썬에 정의된 문법에 맞지 않는 코드가 사용되는 경우

for 루프 내에서 변수 x를 정의하지 않은 경우

값이 0인 것을 나누거나 나누려는 경우

입력한 경로에 파일이나 디렉터리가 존재하지 않는 경우

잘못된 데이터의 유형이 연산이나 함수에 적용될 때 발생

...

..

.

**etc**



## 2. 오류 예외 처리 기법

```
try:  
    ...  
except [발생오류 [as 오류변수]]:  
    ...
```

오류 처리를 위한 try-except 구문의 기본형태이다.

try의 코드를 실행하고 에러가 발생했을 경우,  
except문의 코드를 실행한다.



## 2-1.try-except 문

1. try-except만 쓰는 방법
2. 발생 오류만 포함한 except 문
3. 발생 오류와 오류 변수까지 포함한 except 문

```
try:  
    ...  
except:  
    ...
```

```
try:  
    ...  
except 발생오류:  
    ...
```

```
try:  
    ...  
except 발생오류 as 오류변수:  
    ...
```

오류가 발생한 일반적 경우  
미리 설정한 오류와 같은 오류일  
경우에만 실행을 함

오류가 발생했을 때 오류의 내용까지 알고 싶을 경우  
사용함



## 2-2.try-finally 문

```
# try_finally.py

try:
    f = open('foo.txt', 'w')
    # 무언가를 수행한다.

    (... 생략 ...)

finally:
    f.close() # 중간에 오류가 발생하더라도 무조건 실행된다.
```

try문 수행 도중.

예외가 발생해도 상관하지  
않고 항상 수행한다.

보통 리소스를 close를 할 때  
많이 사용한다.



## 2-3. 여러 개의 오류 처리하기

```
# many_error.py
try:
    a = [1,2]
    print(a[3])
    4/0
except ZeroDivisionError:
    print("0으로 나눌 수 없습니다.")
except IndexError:
    print("인덱싱 할 수 없습니다.")
```

```
PS C:\Users\cwrgw\Desktop\폴더\대신고vs
음/3.py"
인덱싱 할 수 없습니다.
```

```
PS C:\Users\cwrgw\Desktop\폴더\대신고vs
```

```
try:
    a = [1,2]
    print(a[3])
    4/0
except ZeroDivisionError as e:
    print(e)
except IndexError as e:
    print(e)
```

```
PS C:\Users\cwrgw\Desktop\폴더\대
음/3.py"
list index out of range
```

```
PS C:\Users\cwrgw\Desktop\폴더\대
```



## 2-4.try-else 문

```
try:  
    ...  
except [발생오류 [as 오류변수]]:  
    ...  
else: # 오류가 없을 경우에만 수행  
    ...
```

```
# try_else.py  
try:  
    age=int(input('나이를 입력하세요: '))  
except:  
    print('입력이 정확하지 않습니다.')  
else:  
    if age <= 18:  
        print('미성년자는 출입금지입니다.')  
    else:  
        print('환영합니다.')
```

```
PS C:\Users\cwrsgw\Desktop\  
음/3.py"  
나이를 입력하세요: 999  
환영합니다.  
PS C:\Users\cwrsgw\Desktop\
```



## 3. 오류 회피하기

```
# error_pass.py
try:
    f = open("나없는파일", 'r')
except FileNotFoundError:
    pass
```

try 문 안에서 FileNotFoundError가 발생할 경우, pass를 사용하여 오류를 그냥 회피하도록 작성한 예제이다.

```
PS C:\Users\cwrqgw\Desktop\풀더\대신고vs\
음/3.py"
PS C:\Users\cwrqgw\Desktop\풀더\대신고vs\
```



## 4. 오류 일부러 발생시키기

```
# error_raise.py
class Bird:
    def fly(self):
        raise NotImplementedError

class Eagle(Bird):
    pass

eagle = Eagle()
eagle.fly()
```

```
Traceback (most recent call last)
File "c:\Users\cwrugw\Desktop\error_raise.py", line 10, in <module>
    eagle.fly()
File "c:\Users\cwrugw\Desktop\error_raise.py", line 5, in fly
    raise NotImplementedError
NotImplementedError
PS C:\Users\cwrugw\Desktop\풀더
```

```
class Eagle(Bird):
    def fly(self):
        print("very fast")
eagle = Eagle()
eagle.fly()
```

- PS C:\Users\cwrugw\Desktop\풀더\3.py
- very fast
- PS C:\Users\cwrugw\Desktop\풀더\3.py

어떤 클래스에서 무조건 어떤 함수가 포함되어 있는 상태로 실행되게 할 때

`raise NotImplementedError`을 이용할 수 있다.



# 5. 예외 만들기

```
# error_make.py
class MyError(Exception):
    pass

def say_nick(nick):
    if nick == '바보':
        raise MyError()
    print(nick)

say_nick("천사")
say_nick("바보")
```

천사  
Traceback (most recent call last)  
File "c:\Users\cwrng\Desktop\error\_make.py", line 10, in <module>  
 say\_nick("바보")  
File "c:\Users\cwrng\Desktop\error\_make.py", line 5, in say\_nick  
 raise MyError()  
MyError

클래스 MyError를 선언한 후에  
raise를 이용해서 예러상황을 직접  
만들 수 있다.

이외에도 try except 문을 이용해서  
예러상황을 만들 수도 있다



# 내장함수

파이썬 안에 이미 내장되어 있는 함수

1. abs
2. all
3. any
4. chr
5. dir
6. divmod
7. enumerate
8. eval
9. filter
10. hex
11. id
12. input
13. int
14. isinstance
15. len
16. list
17. map
18. max
19. min
20. oct
21. open
22. ord
23. pow
24. range
25. round
26. sorted
27. str
28. sum
29. tuple
30. type
31. zip



# 1. abs

어떤 숫자를 입력받았을 때 그 숫자의 절댓값을 리턴하는 함수

```
# abs  
print(abs(-7))  
print(abs(8.5))  
print(0)
```

7  
8.5  
0



## 2. all

반복 가능한 데이터(리스트, 튜플, 문자열, 딕셔너리, 집합 등) x를 입력값으로 받으며 이 x의 요소가 모두 참이면 True, 거짓이 하나라도 있으면 False를 리턴한다.

```
# all
print(all([1,2,3])) # 1, 2, 3 : 모두 참
print(all([0,1,2,3])) # 0 : 거짓 // 1, 2, 3 : 참
print(all([])) # 빈리스트[] : 원래는 거짓이지만 참으로 판별
```

```
True
False
True
```



## 3. any

반복 가능한 데이터 x를 입력으로 받아 x의 요소 중 하나라도 참이 있으면 True를 리턴하고 x가 모두 거짓일 때만 False를 리턴한다. 즉, all(x)의 반대

```
# any
print(any([0,1,2,3])) # 0 : 거짓 // 1, 2, 3 : 참
print(any([0,""])) # 0, "" : 거짓
print(any([])) # 빈리스트[] : 거짓
```

True

False

False



## 4. chr

아스키코드 숫자 값을 입력받아 그 코드에 해당하는 문자를 리턴한다.

0	NUL	32	SPACE	64	@	96	'
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	\$	68	D	100	d

```
# chr  
print(chr(97))
```



## 5. dir

객체가 지닌 변수나 함수를 보여 준다. 즉 그 객체를 사용할 수 있는 방법을 알려준다

```
# dir
a = dir([1,2,3])
print(a)
```

```
b = dir(1)
print(b)
```

```
>>> a = dir([1,2,3])
>>> print(a)
['__add__', '__class__', '__class_getitem__', '__contains__',
 '__delitem__', '__eq__', '__ge__', '__hash__', '__iadd__', '__imul__',
 '__ceiling__', '__repr__', '__reversed__', '__rmul__',
 'insert', 'pop', 'remove', 'reverse', 'sort']
```

```
>>> b = dir(1)
>>> print(b)
['__abs__', '__add__', '__and__', '__bool__', '__ceil__',
 '__class__', '__contains__', '__delitem__', '__eq__', '__ge__',
 '__getattribute__', '__getnewargs__', '__gt__', '__mod__',
 '__mul__', '__ne__', '__neg__', '__new__', '__or__',
 '__ord__', '__radd__', '__rdiv__', '__rmod__', '__rmul__',
 '__ror__', '__rsub__', '__str__', '__sub__', '__subclasshook__',
 '__truediv__', '__trunc__', 'conjugate', 'denominator',
 'fromhex', 'is_integer', 'numerator', 'real', 'to_bytes']
```



## 6. divmod

`divmod(a, b)`는 2개의 숫자 `a`, `b`를 입력으로 받는다. 그리고 `a`를 `b`로 나눈 몫과 나머지를 튜플로 리턴한다.

```
# divmod  
print(divmod(7,3))
```

$$7 = 3 * 2 + 1$$

(2, 1)



## 7. enumerate

enumerate는 '열거하다'라는 뜻이다. 이 함수는 순서가 있는 데이터(리스트, 튜플, 문자열)를 입력으로 받아 인덱스 값을 포함하는 enumerate 객체를 리턴한다. 보통 for문과 같이 사용한다

```
# enumerate
for i, name in enumerate(['apple', 'banana', 'lemon']):
    print(i, name)
```

```
0 apple
1 banana
2 lemon
```

```
i = 0
for letter in ['apple', 'banana', 'lemon']:
    print(i, letter)
    i += 1
```



## 8. eval

문자열로 구성된 표현식을 입력으로 받아 해당 문자열을 실행한 결과값을 리턴하는 함수

```
# eval  
print(eval('1+2'))  
print(eval("hi" + "a"))  
print(eval('divmod(4, 3)'))
```

```
3  
hia  
(1, 1)
```



## 9. filter

첫 번째 인수로 함수, 두 번째 인수로 그 함수에 차례로 들어갈 반복 가능한 데이터를 받는다. 그리고 반복 가능한 데이터의 요소를 순서대로 함수를 호출했을 때 리턴값이 참인 것만 끌어서(걸러 내서) 리턴한다.

```
filter(함수, 반복_가능한_데이터)
```



## 9. filter

리스트를 입력으로 받아 각각의 요소를 판별해서 양수 값만 리턴하는 함수

```
# filter
def positive(l):
    result = []
    for i in l:
        if i > 0:
            result.append(i)
    return result

print(positive([1,-3,2,0,-5,6]))
```

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

print(list(filter(positive, [1, -3, 2, 0, -5, 6])))
```

[1, 2, 6]



## 10. hex

정수를 입력받아 16진수(hexadecimal) 문자열로 변환하여 리턴하는 함수

```
# hex  
print(hex(234))  
print(hex(3))
```

0xea

0x3



## 11. id

객체를 입력받아 객체의 고유 주소값(레퍼런스)을 리턴한다.

```
# id  
print(id(3))  
print(id(4))
```

```
3062586212656  
3062586212688
```



## 12. input

사용자 입력을 받는 함수

```
# input  
a = input()  
print(a)  
  
b = input("Q: ")  
print(b)
```

```
apple  
apple  
Q: banana  
banana
```



## 13. int

정수, 문자열 형태의 숫자, 소수점이 있는 숫자를 정수로 리턴한다

`int(x, radix)`은 radix 진수로 표현된 문자열 x를 10진수로 변환하여 리턴한다.

```
# int  
print(int(3))  
print(int('3'))  
print(int(3.4))
```

3  
3  
3

```
print(int('11',2))  
print(int('1A',16))
```

3  
26



# 14. isinstance

`isinstance(object, class)` 함수는 첫 번째 인수로 객체, 두 번째 인수로 클래스를 받는다. 입력으로 받은 객체가 그 클래스의 인스턴스인지를 판단하여 참이면 `True`, 거짓이면 `False`를 리턴한다.

```
>>> class Person: pass  
...  
>>> a = Person()  
>>> isinstance(a, Person)  
True
```

위 예는 `a` 객체가 `Person` 클래스에 의해 생성된 인스턴스라는 것을 확인시켜 준다.

```
>>> b = 3  
>>> isinstance(b, Person)  
False
```

`b`는 `Person` 클래스로 만든 인스턴스가 아니므로 `False`를 리턴한다.



## 15. len

`len(x)`는 입력값 `x`의 길이(요소의 전체 개수)를 리턴한다.

```
# len
print(len("python"))
print(len([1,2,3]))
print(len((1, 'a')))
```

6  
3  
2



## 16. list

반복 가능한 데이터를 입력받아 리스트로 만들어 리턴한다.

```
# list  
print(list("python"))  
print(list((1,2,3)))
```

```
['p', 'y', 't', 'h', 'o', 'n']  
[1, 2, 3]
```



# 17. map

map(f, iterable)은 함수(f)와 반복 가능한 데이터를 입력으로 받고 입력받은 데이터의 각 요소에 함수 f를 적용한 결과를 리턴한다.

```
# two_times.py
def two_times(numberList):
    result = []
    for number in numberList:
        result.append(number*2)
    return result

result = two_times([1, 2, 3, 4])
print(result)
```

```
# map
def two_times2(x):
    return x*2

a = list(map(two_times2, [1, 2, 3, 4]))
print(a)
```

[2, 4, 6, 8]



## 18. max

인수로 반복 가능한 데이터를 입력받아 그 최댓값을 리턴한다.

```
# max  
print(max([1, 2, 3]))  
print(max("python"))
```

3  
y



## 19. min

인수로 반복 가능한 데이터를 입력받아 그 최솟값을 리턴한다.

```
# min  
print(min([1, 2, 3]))  
print(min("python"))
```

1  
h



## 20. oct

정수를 8진수 문자열로 바꾸어 리턴한다.

```
# oct  
print(oct(234))  
print(oct(34))
```

```
0o352  
0o42
```



# 21. open

`open(filename, [mode])`은 '파일 이름'과 '읽기 방법'을 입력받아 파일 객체를 리턴한다.

mode	설명
w	쓰기 모드로 파일 열기
r	읽기 모드로 파일 열기
a	추가 모드로 파일 열기
b	바이너리 모드로 파일 열기

`b`는 `w`, `r`, `a`와 함께 사용한다. 예를 들어 `rb`는 '바이너리 읽기 모드'를 의미한다. 바이너리 모드는 이진정보 0과 1로만 저장된 파일이다.



# 21. open

```
# open
f = open("파이썬 강의.txt", "w") # w는 새로운 파일이 생기거나
f.write("Hello World") # 파일에 텍스트 쓰기
f.close() # 파일 닫기

f = open("파이썬 강의.txt", "a") # 파일 수정하기, 추가하기
f.write("python")
f.close()
```

> 열려 있는 편집기		2이(가) 저장되지 않음
▽ STUDY	▶ 🔍 ⌂ ⌂	파이썬 강의.txt
▽ _pycache_		1 Hello Worldpython
mod1.cpython-310.pyc		
mod2.cpython-310.pyc		
메모장.txt		
새파일2.txt		
파이썬 강의.txt		



## 22. ord

문자의 유니코드 숫자 값을 리턴한다. chr함수와 반대

```
# ord  
print(ord('a'))  
print(ord('가'))
```

```
97  
44032
```



## 23. pow

pow(x, y)는 x를 y제곱한 결괏값을 리턴한다.

```
# pow  
print(pow(2,3))  
print(pow(7,2))
```

```
8  
49
```



## 24. range

`range([start,] stop [,step])`은 `for` 문과 함께 자주 사용하는 함수이다. 이 함수는 입력받은 숫자에 해당하는 범위 값을 반복 가능한 객체로 만들어 리턴한다.

[`start`]는 시작하는 숫자. 없으면 0부터 시작함.

`stop`은 끝나는 숫자

[`step`]은 숫자 사이의 간격(거리)로 없으면 간격은 1임.



## 24. range

```
# range  
a = list(range(5))  
print(a)  
  
b = list(range(3,8))  
print(b)  
  
c = list(range(3, 10,2))  
print(c)
```

[0, 1, 2, 3, 4]  
[3, 4, 5, 6, 7]  
[3, 5, 7, 9]

```
for i in [0, 1, 2, 3, 4]:  
    print(i, end=", ")
```

```
for i in range(5):  
    print(i, end=", ")
```

0, 1, 2, 3, 4,



## 25. round

`round(number [,ndigits])`는 숫자를 입력받아 반올림해 리턴한다.

[,ndigits]는 ndigits가 있을 수도 있고, 없을 수도 있다라는 뜻이다.

```
# round  
print(round(4.6))  
print(round(1.76))
```

5  
2

소수점 2자리까지 나타낼 수도 있다.

```
print(round(5.6873,2))
```

5.69



## 26. sorted

입력 데이터를 정렬한 후 그 결과를 리스트로 리턴한다.

```
# sorted
print(sorted([3,1,2]))
print(['a','c','b'])
print(sorted("python"))
print(sorted((3,2,1)))
```

```
[1, 2, 3]
['a', 'c', 'b']
['h', 'n', 'o', 'p', 't', 'y']
[1, 2, 3]
```



## 27. str

문자열 형태로 객체를 변환하여 리턴한다.

```
# str  
print(str(1))
```





## 28. sum

입력 데이터의 합을 리턴한다.

```
# sum  
print(sum([1,2,3,4,5]))  
print(sum((1,2,3,4,5)))
```

15  
15



## 29. tuple

반복 가능한 데이터를 튜플로 바꾸어 리턴한다.

```
# tuple  
print(tuple([1,2,3]))  
print(tuple("python"))
```

```
(1, 2, 3)  
('p', 'y', 't', 'h', 'o', 'n')
```



## 30. type

입력값의 자료형이 무엇인지 알려 준다.

```
# type  
print(type('abc'))  
print(type(1))  
print(type((1,2,3)))  
print(type([]))
```

```
<class 'str'>  
<class 'int'>  
<class 'tuple'>  
<class 'list'>
```



# 31. zip

동일한 개수로 이루어진 데이터들을 묶어서 리턴한다.

```
# zip
print(list(zip([1, 2, 3], [4, 5, 6])))
print(list(zip([1, 2, 3], [4, 5, 6], [7, 8, 9])))
print(list(zip("abc", "def")))
print(list(zip([1, 2, 3], ['a', 'b', 'c'])))
```

```
[(1, 4), (2, 5), (3, 6)]
[(1, 4, 7), (2, 5, 8), (3, 6, 9)]
[('a', 'd'), ('b', 'e'), ('c', 'f')]
[(1, 'a'), (2, 'b'), (3, 'c')]
```

# 05-6 표준 라이브러리

- 1. [datetime.date](#)
- 2. [time](#)
  - 1. [time.time](#)
  - 2. [time.localtime](#)
  - 3. [time.asctime](#)
  - 4. [time.ctime](#)
  - 5. [time.strftime](#)
  - 6. [time.sleep](#)
- 3. [math.gcd](#)
- 4. [math.lcm](#)
- 5. [random](#)
- 6. [itertools.zip\\_longest](#)
- 7. [itertools.permutation](#)
- 8. [itertools.combination](#)
- 9. [functools.reduce](#)
- 10. [operator.itemgetter](#)
- 11. [shutil](#)
- 12. [glob](#)
  - 1. 디렉터리에 있는 파일들을 리스트로 만들기 - `glob(pathname)`
- 13. [pickle](#)
- 14. [os](#)
  - 1. 내 시스템의 환경 변수값을 알고 싶을 때 - `os.environ`
  - 2. 디렉터리 위치 변경하기 - `os.chdir`
  - 3. 디렉터리 위치 돌려받기 - `os.getcwd`
  - 4. 시스템 명령어 호출하기 - `os.system`
  - 5. 실행한 시스템 명령어의 결과값 돌려받기 - `os.popen`
- 15. [zipfile](#)
- 16. [threading](#)
- 17. [tempfile](#)
- 18. [traceback](#)
- 19. [json](#)
- 20. [urllib](#)
- 21. [webbrowser](#)

# 1. datetime.date

연, 월, 일로 다음과 같이 `datetime.date` 객체를 만들 수 있다.

```
₩ pytonh.py > ...
1 import datetime
2 day1 = datetime.date(2021, 12, 14)
3 day2 = datetime.date(2023, 4, 5)
4 diff = day2 - day1
5 diff.days
```

477 days

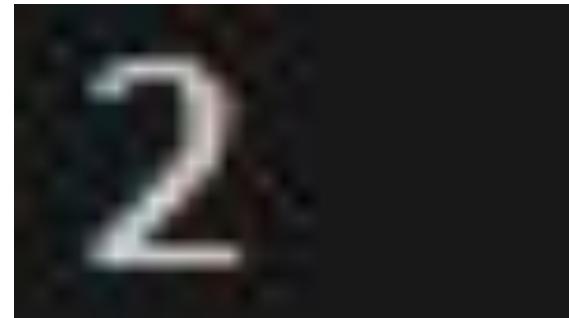
요일은 `datetime.date` 객체의 `weekday` 함수를 사용하면 쉽게 구할 수 있다.

```
₩ pytonh.py > ...
1 import datetime
2
3 day = datetime.date(2021, 12, 14)
4 d=day.weekday()
5
6 print(d)
```

1

0은 월요일을 의미하며 순서대로 1은 화요일, 2는 수요일, ..., 6은 일요일이 된다. 이와 달리 월요일은 1, 화요일은 2, ..., 일요일은 7을 리턴하려면 다음처럼 `isoweekday` 함수를 사용하면 된다.

```
pytonh.py > [실행] d
1 import datetime
2
3 day = datetime.date(2021, 12, 14)
4 d=day.isoweekday()
5
6 print(d)
```



## 2. time

시간과 관련된 `time` 모듈에는 함수가 매우 많다.

1. `time.time`
2. `time.localtime`
3. `time.asctime`
4. `time.ctime`
5. `time.strftime`
6. `time.sleep`

## 2-1. time.time

time.time()은 UTC(universal time coordinated, 협정 세계 표준시)를 사용하여 현재 시간을 실수 형태로 리턴하는 함수이다. 1970년 1월 1일 0시 0분 0초를 기준으로 지난 시간을 초 단위로 리턴해 준다.

```
pythonh.py > ...
1 import time
2 t=time.time()
```

```
1698828719.007713
```

## 2-2. time.localtime

time.localtime은 time.time()이 리턴한 실수값을 사용해서 연, 월, 일, 시, 분, 초, ... 의 형태로 바꾸어 주는 함수이다. 입력 인수 없이 사용할 경우 현재 시각을 기준으로 함수가 수행된다.

pythonh.py > ...

```
1 import time  
2 time.time()  
3 t=time.localtime(time.time())  
4  
5 print(t)
```

```
time.struct_time(tm_year=2023, tm_mon=11, tm_mday=1, tm_hour=18, tm_min=28,  
tm_sec=0, tm_wday=2, tm_yday=305, tm_isdst=0)
```

## 2-3. time.asctime

timeasctime은 time.localtime가 리턴된 튜플 형태의 값을 인수로 받아서 날짜와 시간을 알아보기 쉬운 형태로 리턴하는 함수이다. 입력 인수 없이 사용할 경우 현재 시각을 기준으로 함수가 수행된다.

```
pytonh.py > ...
1 import time
2 time.time()
3 t=time.asctime(time.localtime(time.time()))
4
5 print(t)
```

```
Wed Nov 1 18:31:21 2023
```

## 2-4. time.ctime

`time.asctime(time.localtime(time.time()))`은 간단하게 `time.ctime()`으로 표시할 수 있다.  
`ctime()`과 `asctime`과 다른 점은 항상 현재 시간만을 리턴한다는 점이다.

```
pytonh.py > ...
1 import time
2 time.time()
3 t=time.ctime()
4
5 print(t)
```

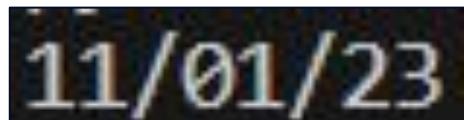
```
Wed Nov  1 18:34:25 2023
```

## 2-5. time.strftime

**strftime** 함수는 시간에 관계된 것을 세밀하게 표현하는 여러 가지 포맷 코드를 제공한다. 입력 인수 없이 사용할 경우 현재 시각을 기준으로 함수가 수행된다.

**ex**

```
pytonh.py > ...
1 import time
2 time.time()
3 t=time.strftime('%x', time.localtime(time.time()))
4 print(t)
```



11/01/23

포맷코드	설명	예
%a	요일의 줄임말	Mon
%A	요일	Monday
%b	달의 줄임말	Jan
%B	달	January
%c	날짜와 시간을 출력함.	Thu May 25 10:13:52 2023
%d	일(day)	[01,31]
%H	시간(hour): 24시간 출력 형태	[00,23]
%I	시간(hour): 12시간 출력 형태	[01,12]
%j	1년 중 누적 날짜	[001,366]
%m	달	[01,12]
%M	분	[01,59]
%p	AM or PM	AM
%S	초	[00,59]
%U	1년 중 누적 주(일요일 시작)	[00,53]
%w	숫자로 된 요일	[0(일), 6(토)]
%W	1년 중 누적 주(월요일 시작)	[00,53]
%x	현재 설정된 지역에 기반한 날짜 출력	05/25/23
%X	현재 설정된 지역에 기반한 시간 출력	17:22:21
%Y	연도 출력	2023
%Z	시간대 출력	대한민국 표준시
%%	문자 %	%
%y	세기 부분을 제외한 연도 출력	01

## 2-6. time.sleep

`time.sleep` 함수는 주로 루프 안에서 많이 사용한다. 이 함수를 사용하면 일정한 시간 간격을 두고 루프를 실행할 수 있다.

```
pythonh.py > ...
1 # sleep1.py
2 import time
3 for i in range(10):
4     print(i)
5     time.sleep(1)
```

위 사진은 1초 간격으로 0부터 9까지의 숫자를 출력한다. `time.sleep` 함수의 인수는 실수 형태를 쓸 수 있다. 즉 10이면 1초, 0.5면 0.5초가 되는 것이다.

### 3. math.gcd

파이썬 3.9버전 이상은 여러개의 인수 사용 가능, 미만은 2개까지 허용

math.gcd 함수를 이용하면 최대 공약수(gcd, greatest common divisor)를 쉽게 구할 수 있다.

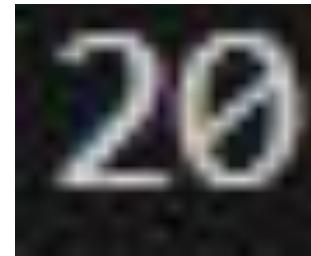
- math.gcd 함수는 파이썬 3.5 버전부터 사용할 수 있다.
- 공약수란 두 수 이상의 여러 수의 공통된 약수를 말하며 공약수 중 가장 큰 수를 최대 공약수라고 말한다. 예를 들어 30과 15의 약수는 1, 3, 5, 15, 최대 공약수는 15이다.

어린이집에서 사탕 60개, 초콜릿 100개, 젤리 80개를 준비했다. 아이들이 서로 싸우지 않도록 똑같이 나누어 봉지에 담는다고 하면 최대 몇 봉지까지 만들 수 있을까? 단, 사탕, 초콜릿, 젤리는 남기지 않고 모두 담도록 한다.



이 문제는 60, 100, 80의 최대 공약수를 구하면 바로 해결된다. 즉, 똑같이 나눌 수 있는 봉지 개수가 최대가 되는 수를 구하면 된다.

```
pytonh.py > ...
1 import math
2 m=math.gcd(60, 100, 80)
3
4 print(m)
```



math.gcd() 함수로 최대 공약수를 구했더니 20이었다. 따라서 최대 20봉지를 만들 수 있다. 각 봉지에 들어가는 사탕, 초콜릿, 젤리의 개수는 다음과 같이 전체 개수를 최대 공약수 20으로 나누면 구할 수 있다.

## 4. math.lcm

math.lcm은 최소 공배수(lcm, least common multiple)를 구할 때 사용하는 함수이다.

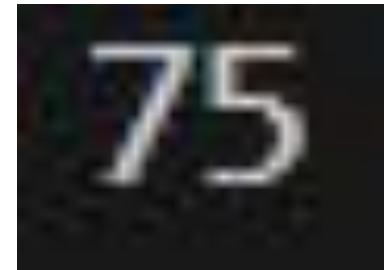
파이썬 3.9버전 이상만 사용 가능

어느 버스 정류장에 시내버스는 15분마다 도착하고 마을버스는 25분마다 도착한다고 한다. 오후 1시에 두 버스가 동시에 도착했다고 할 때 두 버스가 동시에 도착할 다음 시각을 알려면 어떻게 해야 할까?



이 문제는 15와 25의 공통 배수 중 가장 작은 수, 즉 최소 공배수를 구하면 바로 해결된다.

```
pythonh.py > ...
1 import math
2 m=math.lcm(15, 25)
3
4 print(m)
```

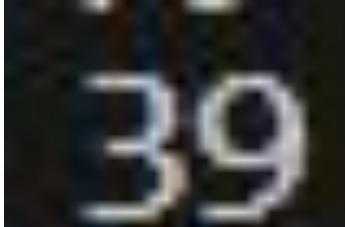


math.lcm 함수를 사용하여 최소 공배수 75를 구했다. 따라서 두 버스가 동시에 도착할 다음 시각은 75분 후인 오후 2시 15분이다.

## 5. random

random은 난수(규칙이 없는 임의의 수)를 발생시키는 모듈이다. 먼저 random과 randint 함수에 대해 알아보자.

```
❷ pytonh.py > ...
1   import random
2   r=random.random()
3
4   print(r)
```



0.4424735479800359

다음 예는 1에서 75 사이의 정수 중에서 난수 값을 리턴해 준다.

```
❷ pytonh.py > ...
1   import random
2   r=random.randint(1, 75)
3
4   print(r)
```

# 6. itertools.zip\_longest

`itertools.zip_longest(*iterables, fillvalue=None)` 함수는 같은 개수의 자료형을 묶는 파이썬 내장 함수인 `zip` 함수와 똑같이 동작한다. 하지만 `itertools.zip_longest()` 함수는 전달한 반복 가능 객체(`*iterables`)의 길이가 서로 다르다면 긴 객체의 길이에 맞춰 `fillvalue`에 설정한 값을 짧은 객체에 채울 수 있다.

예시로 유치원생 5명에게 간식을 나누어 주고자 다음과 같은 파이썬 코드를 작성해 보자.

```
pythond.py > ...
1 # itertools_zip.py
2 students = ['정민서', '정창영', '박민우', '안재민', '진예성']
3 snacks = ['사탕', '초콜릿', '젤리']
4
5 result = zip(students, snacks)
6 print(list(result))
```

간식의 개수가 유치원생보다 적으므로 이 파이썬 코드를 실행하면 다음과 같은 결과가 나온다.

```
[('정민서', '사탕'), ('정창영', '초콜릿'), ('박민우', '젤리')]
```

`students`와 `snacks`의 요소 개수가 다르므로 더 적은 `snacks`의 개수만큼만 `zip()`으로 묶게 된다.

`students`의 요소 개수가 `snacks`보다 많을 때 그만큼을 ‘새우깡’으로 채우려면 어떻게 해야 할까? 이럴 때 요소 개수가 많은 것을 기준으로 자료형을 묶는 `itertools.zip_longest()`를 사용하면 된다. 부족한 항목은 `None`으로 채우는데, 다음처럼 `fillvalue`로 값을 지정하면 `None` 대신 다른 값으로 채울 수 있다.

```
❷ pythonh.py > ...
1  import itertools
2
3  students = ['정민서', '안재민', '박민우', '정창영', '진예성']
4  snacks = ['사탕', '초콜릿', '젤리', ]
5
6  result = itertools.zip_longest(students, snacks, fillvalue='새우깡')
7  print(list(result))
```

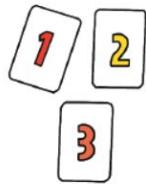
실행 결과는 다음과 같다.

```
[('정민서', '사탕'), ('안재민', '초콜릿'), ('박민우', '젤리'), ('정창영', '새우깡'), ('진예성', '새우깡')]
```

# 7. itertools.permutation

`itertools.permutations(iterator, r)`은 반복 가능한 객체 중에서 r개를 선택한 순열을 이터레이터로 리턴하는 함수이다.

1, 2, 3이라는 숫자가 적힌 3장의 카드에서 2장의 카드를 꺼내 만들 수 있는 2자리 숫자를 모두 구하려면 어떻게 해야 할까?



```
pythonh.py > ...
1   import itertools
2   l=list(itertools.permutations(['1', '2', '3'], 2))
3
4   print(l)
```

[1, 2, 3]이라는 3장의 카드 중 순서에 상관없이 2장을 뽑는 경우의 수는 모두 3가지이다(조합).

- 1, 2
- 2, 3
- 1, 3

하지만 이 문제에서는 2자리 숫자이므로 이 3가지에 순서를 더해 다음처럼 6가지가 된다(순열).

- 1, 2
- 2, 1
- 2, 3
- 3, 2
- 1, 3
- 3, 1

```
[(1, 2), (1, 3), (2, 1), (2, 3), (3, 1), (3, 2)]
```

이 순열은 `itertools.permutations()`를 사용하면 간단히 구할 수 있다.

따라서 만들 수 있는 2자리 숫자는 다음과 같이 모두 6가지이다.

```
# pythonh.py > ...
1 import itertools
2 list(itertools.permutations(['1', '2', '3'], 2))
3 for a, b in itertools.permutations(['1', '2', '3'], 2):
4     print(a+b)
```

12  
13  
21  
23  
31  
32

점프 투 파이썬

### 조합을 사용하는 함수

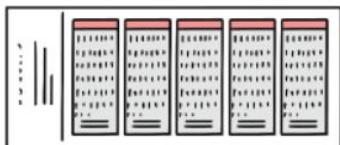
3장의 카드에서 순서에 상관없이 2장을 고르는 조합은 다음처럼 `itertools.combinations()`를 사용하면 된다.

```
>>> import itertools
>>> list(itertools.combinations(['1', '2', '3'], 2))
[('1', '2'), ('1', '3'), ('2', '3')]
```

# 8. itertools.combination

`itertools.combinations(iterable, r)`은 반복 가능한 객체 중에서 r개를 선택한 조합을 이터레이터로 리턴하는 함수이다.

1~45 중 서로 다른 숫자 6개를 뽑는 로또 번호의 모든 경우의 수(조합)를 구하고 그 개수를 출력하려면 어떻게 해야 할까?



다음과 같이 `itertools.combinations()`를 사용하면 45개의 숫자 중 6개를 선택하는 경우의 수를 구할 수 있다.

```
pytonh.py > ...
1 import itertools
2 it = itertools.combinations(range(1, 46), 6)
3
4 for num in it:
5     print(num)
```

순환하지 않고 개수만 세려면 다음과 같이 하면 된다.

```
pytonh.py > ...
1 import itertools
2 it = itertools.combinations(range(1, 46), 6)
3
4 l=len(list(itertools.combinations(range(1, 46), 6)))
5 print(l)
```

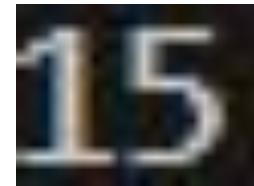
8145060

## 9. functools.reduce

`functools.reduce(function, iterable)`은 함수(function)를 반복 가능한 객체(iterable)의 요소에 차례대로 (왼쪽에서 오른쪽으로) 누적 적용하여 이 객체를 하나의 값으로 줄이는 함수이다.

다음은 입력 인수 `data`의 요소를 모두 더하여 리턴하는 `add` 함수이다.

```
⚡ pytonh.py > ...
1  def add(data):
2      result = 0
3      for i in data:
4          result += i
5      return result
6
7  data = [1, 2, 3, 4, 5]
8  result = add(data)
9  print(result)
```

A large, stylized number 15, rendered in a dark background with a glowing, metallic or digital effect around its edges.

`functools.reduce()`를 사용하여 마찬가지로 동작하는 코드를 작성하려면 어떻게 해야 할까? `functools.reduce()`를 사용한 코드는 다음과 같다.

```
pytonh.py > ...
1 import functools
2
3 data = [1, 2, 3, 4, 5]
4 result = functools.reduce(lambda x, y: x + y, data)
5 print(result)
```



`functools.reduce()`를 사용하면 `reduce()`에 선언한 람다 함수를 `data` 요소에 차례대로 누적 적용하여 다음과 같이 계산한다.

$$(((1+2)+3)+4)+5$$

위에 것과 동일

# 10. operator.itemgetter

operator.itemgetter는 주로 sorted와 같은 함수의 key 매개변수에 적용하여 다양한 기준으로 정렬할 수 있도록 도와주는 모듈이다.

예를 들어 학생의 이름, 나이, 성적 등의 정보를 저장한, 다음과 같은 students 리스트가 있다고 가정해 보자.

```
students = [
    ("jane", 22, 'A'),
    ("dave", 32, 'B'),
    ("sally", 17, 'B'),
]
```

students 리스트에는 3개의 튜플이 있으며 각 튜플은 순서대로 이름, 나이, 성적에 해당하는 데이터로 이루어졌다. 이 리스트를 나이순으로 정렬하려면 어떻게 해야 할까?



이 파일을 실행하여 출력해 보면 다음과 같이 나이 순서대로 정렬한 것을 확인할 수 있다.

```
[('sally', 17, 'B'), ('jane', 22, 'A'), ('dave', 32, 'B')]
```

itemgetter(1)은 students의 아이템인 튜플의 두 번째 요소를 기준으로 정렬하겠다는 의미이다. 만약 itemgetter(2)와 같이 사용한다면 성적순으로 정렬한다.

이 문제는 다음처럼 sorted 함수의 key 매개변수에 itemgetter()를 적용하면 쉽게 해결할 수 있다.

```
python.py > ...
1  from operator import itemgetter
2
3
4  students = [
5      ("jane", 22, 'A'),
6      ("dave", 32, 'B'),
7      ("sally", 17, 'B')
8  ]
9
10 result = sorted(students, key=itemgetter(1))
11 print(result)
```

이번에는 students의 요소가 다음처럼 딕셔너리일 때를 생각해 보자.

```
students = [  
    {"name": "jane", "age": 22, "grade": 'A'},  
    {"name": "dave", "age": 32, "grade": 'B'},  
    {"name": "sally", "age": 17, "grade": 'B'},  
]
```

딕셔너리일 때도 마찬가지로 age를 기준으로 정렬해 보자. 이때도 마찬가지로 itemgetter()를 적용하면 된다. 단, 이번에는 itemgetter('age') 처럼 딕셔너리의 키를 사용해야 한다. itemgetter('age') 는 딕셔너리의 키인 age를 기준으로 정렬하겠다는 의미이다.



pytonh.py > ...

```
1  from operator import itemgetter  
2  
3  students = [  
4      {"name": "경민", "age": 22, "grade": "A"},  
5      {"name": "혜성", "age": 32, "grade": "B"},  
6      {"name": "연우", "age": 17, "grade": "B"},  
7  ]  
8  
9  result = sorted(students, key=itemgetter('age'))  
10 print(result)
```

```
[{'name': '연우', 'age': 17, 'grade': 'B'}, {'name': '경민', 'age': 22, 'grade': 'A'}, {'name': '혜성', 'age': 32, 'grade': 'B'}]
```

age 순으로 정렬이 된 것을 볼 수 있다.

# 11. shutil

shutil은 파일을 복사(copy)하거나 이동(move)할 때 사용하는 모듈이다.

작업 중인 파일을 자동으로 백업하는 기능을 구현하고자 `c:\doit\a.txt`를 `c:\temp\a.txt.bak`이라는 이름으로 복사하는 프로그램을 만들고자 한다. 어떻게 만들어야 할까? `c:\doit` 디렉터리에 `a.txt`를 만드는 중이며 백업용 `c:\temp` 디렉터리는 이미 만들었다고 가정한다.

다음은 shutil을 사용한 방법이다.

```
python.py > ...
1 import shutil
2 s=shutil.copy("c:/doit/a.txt", "c:/temp/a.txt.bak")
3
4 print(s)
```

# 12. glob

가끔 파일을 읽고 쓰는 기능이 있는 프로그램을 만들다 보면 특정 디렉터리에 있는 파일 이름 모두를 알아야 할 때가 있다. 이럴 때 사용하는 모듈이 바로 glob이다.

## 디렉터리에 있는 파일들을 리스트로 만들기 - glob(pathname)

glob 모듈은 디렉터리 안의 파일들을 읽어서 리턴한다. \*, ? 등 메타 문자를 써서 원하는 파일만 읽어 들일 수도 있다. 다음은 C:/doit 디렉터리에 있는 파일 중 이름이 mark로 시작하는 파일을 모두 찾아서 읽어들이는 예이다.

```
FileNotFoundException: [Errno 2] No such file or directory: 'c:/doit/a.txt'  
PS C:\Users\saysa\Desktop\vscode> & C:/Users/saysa/AppData/Local/Microsoft/WindowsApps/p  
ython3.11.exe c:/Users/saysa/Desktop/vscode/python.py  
[]
```

전 mark로 시작하는 파일이 없어 나오지 않는 모습입니다.

# 13. pickle

pickle은 객체의 형태를 그대로 유지하면서 파일에 저장하고 불러올 수 있게 하는 모듈이다. 다음 예는 pickle 모듈의 dump 함수를 사용하여 딕셔너리 객체인 data를 그대로 파일에 저장하는 방법을 보여 준다.

```
❶ python.py > ...
1 import pickle
2 f = open("test.txt", "wb")
3 data = {1: 'python', 2: 'you need'}
4 pickle.dump(data, f)
5 f.close()
```

다음은 pickle.dump로 저장한 파일을 pickle.load를 사용해서 원래 있던 딕셔너리 객체(data) 상태 그대로 불러오는 예이다.

```
❷ python.py > ...
1 import pickle
2 f = open("test.txt", "wb")
3 data = {1: 'python', 2: 'you need'}
4 pickle.dump(data, f)
5 f.close()
6
7 f = open("test.txt", 'rb')
8 data = pickle.load(f)
9 print(data)
```

위 예에서는 딕셔너리 객체를 사용했지만, 어떤 자료형이든 저장하고 불러올 수 있다.

# 14. OS

OS 모듈은 환경 변수나 디렉터리, 파일 등의 OS 자원을 제어할 수 있게 해 주는 모듈이다.

## 내 시스템의 환경 변수를 알고 싶을 때 - os.environ

시스템은 제각기 다른 환경 변수를 가지고 있는데, `os.environ`은 현재 시스템의 환경 변수를 리턴한다. 다음을 따라 해 보자.

```
environ('ALLUSERSPROFILE': 'C:\ProgramData', 'APPDATA': 'C:\Users\saysa\AppData\Roaming', 'CHROME_CRASHPAD_PIPE_NAME': '\\\\.\\pipe\\crashpad_1388_PBIGMURTFBSPV', 'COMONPROGRAMFILES(X86)': 'C:\Program Files(x86)\Common Files', 'COMONPROGRAMFILES': 'C:\Program Files\Common Files', 'COMPUTERNAME': 'DESKTOP-SV8BRQG', 'COMSPEC': 'C:\Windows\system2\cmd.exe', 'DRIVERDATA': 'C:\Windows\System2\Drivers\DriverData', 'FPS_BROWSER_APP_PROFILE_STRING': 'Internet Explorer', 'FPS_BROWSER_USER_PROFILE_STRING': 'Default', 'HOMEDRIVE': 'C:', 'HOMEPATH': '\\Users\\saysa', 'LOCALAPPDATA': 'C:\Users\\saysa\\AppData\\Local', 'LOGONSERVER': '\\\\DESKTOP-SV8BRQG', 'NUMBER_OF_PROCESSORS': '8', 'ONEDRIVE': 'C:\Users\\saysa\\OneDrive', 'ORIGINAL_XDG_CURRENT_DESKTOP': 'undefined', 'OS': 'Windows_NT', 'PATH': 'C:\Program Files(x86)\Razer Chroma SDK\bin;C:\Program Files\\Razer\\chromabroadcast\\bin;C:\Windows\\System32;C:\\Windows\\System32\\Wbem;C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\;C:\\Windows\\System32\\OpenSSH;C:\\Windows\\System32\\config\\systemprofile\\Appdata\\Local\\Microsoft\\WindowsApps', 'PATHEXT': '.COM;.EXE;.BAT;.CMD;.CPL;.VBScript;.JS;.WSF;.WSH;.MSC;.CPL', 'PROCESSOR_ARCHITECTURE': 'AMD64', 'PROCESSOR_IDENTIFIER': 'Intel Family 6 Model 158 Stepping 9, GenuineIntel', 'PROCESSOR_LEVEL': '6', 'PROCESSOR_REVISION': '9e09', 'PROGRAMDATA': 'C:\ProgramData', 'PROGRAMFILES': 'C:\Program Files', 'PROGRAMFILES(X86)': 'C:\Program Files(x86)', 'PROGRAMW6432': 'C:\Program Files\\WindowsPowerShell\\v1.0\\Modules\\Windows\\System32\\WindowsPowerShell\\v1.0\\Modules\\Public', 'PUBLIC': 'C:\Users\\Public', 'SESSIONNAME': 'Console', 'SYSTEMDRIVE': 'C:\\Windows', 'TEMP': 'C:\\Users\\saysa\\AppData\\Local\\Temp', 'TMP': 'C:\\Users\\saysa\\AppData\\Local\\Temp', 'USERDOMAIN_ROAMINGPROFILE': 'DESKTOP-SV8BRQG', 'USERNAME': 'saysa', 'USERPROFILE': 'C:\\Users\\saysa', 'WINDOW': 'C:\\Windows', 'TERM_PROGRAM': 'vscode', 'TERM_PROGRAM_VERSION': '1.84.0', 'LANG': 'ko_KR.UTF-8', 'COLORTERM': 'truecolor', 'VSCode_INJECTION': '1', 'VSCode_Nonce': 'f5e1ec6-5d7-427-d-a1c-882-7ba4cb3', 'PYTHONUSERBASE': 'C:\\Users\\saysa\\Appdata\\Local\\Packages\\PythonSoftwareFoundation.Python.3.11_qbz5n2kfrgpo\\LocalCache\\Local-packages')
```

이 결과값은 필자의 시스템 정보이다. `os.environ`은 환경 변수에 대한 정보를 딕셔너리 형태로 구성된 `environ` 객체로 리턴한다. 자세히 보면 여러 가지 유용한 정보를 찾을 수 있다.

리턴받은 객체는 다음과 같이 호출하여 사용할 수 있다. 다음은 필자 시스템의 PATH 환경 변수 내용이다.

```
python.py
1 import os
2 os.environ['PATH']
3
4 print(os.environ)
```

```
python.py
1 import os
2 os.environ
3
4 print(os.environ)
```

```
environ('ALLUSERSPROFILE': 'C:\ProgramData', 'APPDATA': 'C:\Users\saysa\AppData\Roaming', 'CHROME_CRASHPAD_PIPE_NAME': '\\\\.\\pipe\\crashpad_22904_TAPBKGMPZ0DPX', 'COMONPROGRAMFILES(X86)': 'C:\Program Files(x86)\Common Files', 'COMONPROGRAMFILES': 'C:\Program Files\Common Files', 'COMPUTERNAME': 'DESKTOP-SV8BRQG', 'COMSPEC': 'C:\Windows\system2\cmd.exe', 'DRIVERDATA': 'C:\Windows\System2\Drivers\DriverData', 'FPS_BROWSER_APP_PROFILE_STRING': 'Internet Explorer', 'FPS_BROWSER_USER_PROFILE_STRING': 'Default', 'HOMEDRIVE': 'C:', 'HOMEPATH': '\\Users\\saysa', 'LOCALAPPDATA': 'C:\Users\\saysa\\AppData\\Local', 'LOGONSERVER': '\\\\DESKTOP-SV8BRQG', 'NUMBER_OF_PROCESSORS': '8', 'ONEDRIVE': 'C:\Users\\saysa\\OneDrive', 'ORIGINAL_XDG_CURRENT_DESKTOP': 'undefined', 'OS': 'Windows_NT', 'PATH': 'C:\Program Files(x86)\Razer Chroma SDK\bin;C:\Program Files\\Razer\\chromabroadcast\\bin;C:\Windows\\System32;C:\\Windows\\System32\\Wbem;C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\;C:\\Windows\\System32\\config\\systemprofile\\Appdata\\Local\\Microsoft\\WindowsApps', 'PATHEXT': '.COM;.EXE;.BAT;.CMD;.CPL;.VBScript;.JS;.WSF;.WSH;.MSC;.CPL', 'PROCESSOR_ARCHITECTURE': 'AMD64', 'PROCESSOR_IDENTIFIER': 'Intel Family 6 Model 158 Stepping 9, GenuineIntel', 'PROCESSOR_LEVEL': '6', 'PROCESSOR_REVISION': '9e09', 'PROGRAMDATA': 'C:\ProgramData', 'PROGRAMFILES': 'C:\Program Files', 'PROGRAMFILES(X86)': 'C:\Program Files(x86)', 'PROGRAMW6432': 'C:\Program Files\\WindowsPowerShell\\v1.0\\Modules\\Windows\\System32\\WindowsPowerShell\\v1.0\\Modules\\Public', 'PUBLIC': 'C:\Users\\Public', 'SESSIONNAME': 'Console', 'SYSTEMDRIVE': 'C:\\Windows', 'TEMP': 'C:\\Users\\saysa\\AppData\\Local\\Temp', 'TMP': 'C:\\Users\\saysa\\AppData\\Local\\Temp', 'USERDOMAIN_ROAMINGPROFILE': 'DESKTOP-SV8BRQG', 'USERNAME': 'saysa', 'USERPROFILE': 'C:\\Users\\saysa', 'WINDOW': 'C:\\Windows', 'TERM_PROGRAM': 'vscode', 'TERM_PROGRAM_VERSION': '1.84.0', 'LANG': 'ko_KR.UTF-8', 'COLORTERM': 'truecolor', 'VSCode_INJECTION': '1', 'VSCode_Nonce': '0fc8b8a3-995-43e1-95f4-2cfarbs1dd0', 'PYTHONUSERBASE': 'C:\\Users\\saysa\\Appdata\\Local\\Packages\\PythonSoftwareFoundation.Python.3.11_qbz5n2kfrgpo\\LocalCache\\Local-packages')
```

## 디렉터리 위치 변경하기 - os.chdir

os.chdir를 사용하면 다음과 같이 현재 디렉터리의 위치를 변경할 수 있다.

```
python.py
1 import os
2 os.environ
3 os.environ['PATH']
4 os.chdir("C:\\WINDOWS")
```

## 디렉터리 위치 돌려받기 - os.getcwd

os.getcwd는 현재 자신의 디렉터리 위치를 리턴한다.

```
python.py
1 import os
2 os.environ
3 os.environ['PATH']
4 os.chdir("C:\\WINDOWS")
5 os.getcwd()
```

```
<built-in function chdir>
PS C:\\Users\\saysa\\Desktop\\scode\\python.py
<built-in function getcwd>
```

## 시스템 명령어 호출하기 - os.system

시스템 자체의 프로그램이나 기타 명령어를 파이썬에서 호출할 수도 있다. `os.system("명령어")`처럼 사용한다. 다음은 현재 디렉터리에서 시스템 명령어 dir을 실행하는 예이다.

```
python.py
1 import os
2 os.environ
3 os.environ['PATH']
4 os.chdir("C:\\WINDOWS")
5 os.getcwd()
6 os.system("dir")
```

```
C:\\WINDOWS 디렉터리
2023-10-12 오후 11:44 <DIR> .
2023-10-12 오후 11:44 <DIR> ..
2022-07-16 오전 04:26 <DIR> addins
2022-08-29 오후 11:42 287,178 AhnInst.log
2023-08-18 오후 03:51 <DIR> appcompat
2023-10-12 오후 11:33 <DIR> apppatch
2023-11-16 오후 09:12 <DIR> AppReadiness
2022-10-12 오후 11:33 <DIR> broadcastv
2023-10-12 오후 08:20 81,408 bfsvc.exe
2022-07-16 오전 04:22 <DIR> Boot
2022-07-16 오전 04:22 <DIR> Branding
2023-10-28 오전 11:47 <DIR> CbsTemp
2022-07-16 오전 04:31 <DIR> Containers
2022-07-16 오전 04:30 <DIR> CSC
2022-07-16 오전 04:22 <DIR> Cursors
2022-07-16 오전 04:55 <DIR> debug
2022-07-16 오전 04:22 <DIR> diagnostics
```

## 실행한 시스템 명령어의 결괏값 돌려받기 - os.popen

os.popen은 시스템 명령어를 실행한 결괏값을 읽기 모드 형태의 파일 객체로 리턴한다.

```
❶ python.py > ...
1  import os
2  os.environ
3  os.environ['PATH']
4  os.chdir("C:\WINDOWS")
5  os.getcwd()
6  os.system("dir")
7
8  f = os.popen("dir")
9  print(f.read())
```

이 밖에 유용한 os 관련 함수는 다음과 같다.

함수	설명
os.mkdir(디렉터리)	디렉터리를 생성한다.
os.rmdir(디렉터리)	디렉터리를 삭제한다. 단, 디렉터리가 비어 있어야 삭제할 수 있다.
os.remove(파일)	파일을 지운다.
os.rename(src, dst)	src라는 이름의 파일을 dst라는 이름으로 바꾼다.

# 15. zipfile

zipfile은 여러 개의 파일을 zip 형식으로 합치거나 이를 해제할 때 사용하는 모듈이다.

다음과 같은 3개의 텍스트 파일이 있다고 가정해 보자.

```
a.txt  
b.txt  
c.txt
```

이 3개의 텍스트 파일을 하나로 합쳐 'mytext.zip'이라는 파일을 만들고 이 파일을 원래의 텍스트 파일 3개로 해제하는 프로그램을 만들려면 어떻게 해야 할까?

zipfile.ZipFile()을 사용하여 해결해 보자.

```
❶ python.py > ...  
1  # zipfile_test.py  
2  import zipfile  
3  
4  # 파일 합치기  
5  with zipfile.ZipFile('mytext.zip', 'w') as myzip:  
6      myzip.write('a.txt')  
7      myzip.write('b.txt')  
8      myzip.write('c.txt')  
9  
10 # 해제하기  
11 with zipfile.ZipFile('mytext.zip') as myzip:  
12     myzip.extractall()
```

```
# 특정 파일만 해제하기  
with zipfile.ZipFile('mytext.zip') as myzip:  
    myzip.extract('a.txt')
```

```
# 압축하여 루기  
with zipfile.ZipFile('mytext.zip', 'w', compression=zipfile.ZIP_LZMA, compresslevel=9) as myzip:  
    (... 생략 ...)
```

compression에는 4가지 종류가 있다.

- ZIP\_STORED: 압축하지 않고 파일을 zip으로만 묶는다. 속도가 빠르다.
- ZIP\_DEFLATED: 일반적인 zip 압축으로 속도가 빠르고 압축률은 낮다(호환성이 좋다).
- ZIP\_BZIP2: bzip2 압축으로 압축률이 높고 속도가 느리다.
- ZIP\_LZMA: lzma 압축으로 압축률이 높고 속도가 느리다(7zip과 동일한 알고리즘으로 알려져 있다).

compressionlevel은 압축 수준을 의미하는 숫자값으로, 1~9를 사용한다. 1은 속도가 가장 빠르지만 압축률이 낮고, 9는 속도는 가장 느리지만 압축률이 높다.

# 16. threading

컴퓨터에서 동작하고 있는 프로그램을 프로세스(process)라고 한다. 보통 1개의 프로세스는 1가지 일만 하지만, 스레드(thread)를 사용하면 한 프로세스 안에서 2가지 또는 그 이상의 일을 동시에 수행할 수 있다

```
❶ python.py > ...
1 # thread_test.py
2 import time
3
4 def long_task(): # 5초의 시간이 걸리는 함수
5     for i in range(5):
6         time.sleep(1) # 1초간 대기한다.
7         print("working:%s\n" % i)
8
9 print("Start")
10
11 for i in range(5): # long_task를 5회 수행한다.
12     long_task()
13
14 print("End")
```

```
Start
working:0
working:1
working:2
working:3
working:4
```

`long_task`는 수행하는 데 5초의 시간이 걸리는 함수이다. 위 프로그램은 이 함수를 총 5번 반복해서 수행하는 프로그램이다. 이 프로그램은 5초가 5번 반복되므로 총 25초의 시간이 걸린다.

하지만 앞에서 설명했듯이 스레드를 사용하면 5초의 시간이 걸리는 `long_task` 함수를 동시에 실행할 수 있으므로 시간을 줄일 수 있다.

```
❶ python.py > ...
1 # thread_test.py
2 import time
3 import threading # 스레드를 생성하기 위해서는 threading 모듈이 필요하다.
4
5 def long_task():
6     for i in range(5):
7         time.sleep(1)
8         print("working:%s\n" % i)
9
10 print("Start")
11
12 threads = []
13 for i in range(5):
14     t = threading.Thread(target=long_task) # 스레드를 생성한다.
15     threads.append(t)
16
17 for t in threads:
```

```
for t in threads:
    t.start() # 스레드를 실행한다.

print("End")
```

```
Start
End
working:0
working:0
working:0
working:0
working:0
working:1
working:1
working:1
working:1
working:1
working:2
```

이와 같이 프로그램을 수정하고 실행해 보면 25초 걸리던 작업이 5초 정도에 수행되는 것을 확인할 수 있다. `threading.Thread`를 사용하여 만든 스레드 객체가 동시에 작업을 가능하게 해 주기 때문이다.

하지만 프로그램을 실행해 보면 "Start"와 "End"가 먼저 출력되고 그 이후에 스레드의 결과가 출력되는 것을 확인할 수 있다. 그리고 프로그램이 정상 종료되지 않는다. 우리가 기대하는 것은 "Start"가 출력되고 그다음에 스레드의 결과가 출력된 후 마지막으로 "End"가 출력되는 것이다.

이 문제를 해결하기 위해서는 프로그램을 다음과 같이 수정해야 한다.

```
❶ python.py > ...
1   # thread_test.py
2 v import time
3  import threading
4
5 v def long_task():
6 v   for i in range(5):
7   |     time.sleep(1)
8   |     print("working:%s\n" % i)
9
10  print("Start")
11
12  threads = []
13 v for i in range(5):
14   | t = threading.Thread(target=long_task)
15   | threads.append(t)
16
17 v for t in threads:
18   | t.start()
19
20 v for t in threads:
21   | t.join() # join으로 스레드가 종료될때까지 기다린다.
22
23  print("End")
```

```
Start
working:0
working:0

working:0
working:0
working:0
working:0
working:1
working:1
working:1
working:1
working:1
working:2
working:2
```

# 17. tempfile

파일을 임시로 만들어서 사용할 때 유용한 모듈이 바로 `tempfile`이다. `tempfile.mkstemp()`는 중복되지 않는 임시 파일의 이름을 무작위로 만들어서 리턴한다.

```
import tempfile  
filename = tempfile.mkstemp()  
filename  
  
print(filename)
```

(3, 'C:\\\\Users\\\\saysa\\\\AppData\\\\Local\\\\Temp\\\\tmp9coppdd2')

`tempfile.TemporaryFile()`은 임시 저장 공간으로 사용할 파일 객체를 리턴한다. 이 파일은 기본적으로 바이너리 쓰기 모드(`wb`)를 갖는다. `f.close()`가 호출되면 이 파일은 자동으로 삭제된다.

```
python.py > ...  
1 import tempfile  
2 f = tempfile.TemporaryFile()  
3 f.close()  
4  
5 print(f.close())
```

<tempfile.\_TemporaryFileWrapper object at 0x00000218612D3D10>  
PS C:\\Users\\\\saysa\\\\Desktop\\\\vscode> & C:/Users/saysa/AppData/Local/Temp/python.py  
None

# 17. traceback

traceback은 프로그램 실행 중 발생한 오류를 추적하고자 할 때 사용하는 모듈이다.

다음과 같은 코드를 작성하여 실행해 보자.

```
python.py > ..
1 # traceback_test.py
2 def a():
3     return 1/0
4
5 def b():
6     a()
7
8 def main():
9     try:
10         b()
11     except:
12         print("오류가 발생했습니다.")
13
14 main()
```

프로그램 실행 결과는 다음과 같다.

오류가 발생했습니다.

main() 함수가 시작되면 b() 함수를 호출하고 b() 함수에서 다시 a() 함수를 호출하여 1을 0으로 나누므로 오류가 발생하여 "오류가 발생했습니다."라는 메시지를 출력했다.

이때 이 코드에서 오류가 발생한 위치와 원인을 정확히 판단할 수 있도록 코드를 업그레이드하려면 어떻게 해야 할까?

오류가 발생한 위치에 다음과 같이 `traceback` 모듈을 적용해 보자.

```
python.py > ...
1  # traceback_test.py
2  import traceback
3
4  def a():
5      return 1/0
6
7  def b():
8      a()
9
10 def main():
11     try:
12         b()
13     except:
14         print("오류가 발생했습니다.")
15         print(traceback.format_exc())
16
17 main()
```

오류가 발생한 위치에 `print(traceback.format_exc())` 문장을 추가했다. `traceback` 모듈의 `format_exc()`는 오류 추적 결과를 문자열로 리턴하는 함수이다. 이렇게 코드를 수정하고 다시 프로그램을 실행하면 다음과 같이 출력될 것이다.

```
Traceback (most recent call last):
File "c:\Users\saysa\Desktop\vscode\python.py", line 12, in main
    b()
File "c:\Users\saysa\Desktop\vscode\python.py", line 8, in b
    a()
File "c:\Users\saysa\Desktop\vscode\python.py", line 5, in a
    return 1/0
           ~^~
ZeroDivisionError: division by zero
```

오류 추적을 통해 `main()` 함수에서 `b()` 함수를 호출하고 `b()` 함수에서 다시 `a()` 함수를 호출하여 `1 / 0`을 실행하려 하므로 `0`으로 나눌 수 없다는 `ZeroDivisionError`가 발생했다는 것을 로그를 통해 정확하게 확인할 수 있다.

# 18. json

json은 JSON 데이터를 쉽게 처리하고자 사용하는 모듈이다.

다음은 개인정보를 JSON 형태의 데이터로 만든 myinfo.json 파일이다.

```
[파일명: myinfo.json]
1 {
2     "name": "홍길동",
3     "birth": "0525",
4     "age": 30
5 }
```

인터넷으로 얻은 이 파일을 읽어 파이썬에서 처리할 수 있도록 딕셔너리 자료형으로 만들려면 어떻게 해야 할까?

JSON 파일을 읽어 딕셔너리로 변환하려면 다음처럼 json 모듈을 사용하면 된다.

```
>>> import json
>>> with open('myinfo.json') as f:
...     data = json.load(f)
...
>>> type(data)
<class 'dict'>
>>> data
{'name': '홍길동', 'birth': '0525', 'age': 30}
```

JSON 파일을 읽을 때는 이 예처럼 `json.load(파일_객체)`를 사용한다. 이렇게 `load()` 함수는 읽은 데이터를 딕셔너리 자료형으로 리턴한다. 이와 반대로 딕셔너리 자료형을 JSON 파일로 생성할 때는 다음처럼 `json.dump(딕셔너리, 파일_객체)`를 사용한다.

```
python.py > ...
1 import json
2 data = {'name': '홍길동', 'birth': '0525', 'age': 30}
3 with open('myinfo.json', 'w') as f:
4     json.dump(data, f)
```

이번에는 파이썬 자료형을 JSON 문자열로 만드는 방법에 대해서 알아보자.

```
python.py > ...
1 import json
2 d = {"name": "홍길동", "birth": "0525", "age": 30}
3 json_data = json.dumps(d)
4 json_data
```

'{"name": "\ud64d\uae38\ub3d9", "birth": "0525", "age": 30}'

딕셔너리 자료형을 JSON 문자열로 만들려면 `json.dumps()` 함수를 사용하면 된다. 그런데 딕셔너리를 JSON 데이터로 변경하면 ‘홍길동’과 같은 한글 문자열이 코드 형태로 표시된다. 왜냐하면 `dump()`, `dumps()` 함수는 기본적으로 데이터를 아스키 형태로 저장하기 때문이다. 유니코드 문자열을 아스키 형태로 저장하다 보니 한글 문자열이 마치 깨진 것처럼 보인다.

그러나 JSON 문자열을 딕셔너리로 다시 역변환하여 사용하는 데는 전혀 문제가 없다. JSON 문자열을 딕셔너리로 변환할 때는 다음처럼 `json.loads()` 함수를 사용한다.

```
>>> json.loads(json_data)
{'name': '홍길동', 'birth': '0525', 'age': 30}
```

# 19. urllib

urllib은 URL을 읽고 분석할 때 사용하는 모듈이다.

브라우저로 위키독스의 특정 페이지를 읽으려면 다음과 같이 요청하면 된다.

```
❷ python.py > ...
1   https://wikidocs.net/페이지_번호[예: https://wikidocs.net/12]
```

그러면 오프라인으로도 읽을 수 있도록 페이지 번호를 입력 받아 위키독스의 특정 페이지를 `wikidocs_페이지_번호.html` 파일로 저장하는 함수는 어떻게 만들어야 할까?

URL을 호출하여 원하는 리소스를 얻으려면 urllib 모듈을 사용해야 한다.

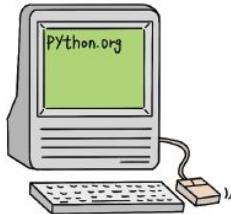
```
❷ python.py > ...
1  # urllib_test.py
2  import urllib.request
3
4  def get_wikidocs(page):
5      resource = 'https://wikidocs.net/{}'.format(page)
6      with urllib.request.urlopen(resource) as s:
7          with open('wikidocs_{}.html' % page, 'wb') as f:
8              f.write(s.read())
```

`get_wikidocs(page)`는 위키독스의 페이지 번호를 입력 받아 해당 페이지의 리소스 내용을 파일로 저장하는 함수이다. 이 코드에서 보듯이 `urllib.request.urlopen(resource)`로 `s` 객체를 생성하고 `s.read()`로 리소스 내용 전체를 읽어 이를 저장할 수 있다. 예를 들어 `get_wikidocs(12)`라고 호출하면 `https://wikidocs.net/12` 웹 페이지를 `wikidocs_12.html`라는 파일로 저장한다.

# 20. webbrowser

webbrowser는 파이썬 프로그램에서 시스템 브라우저를 호출할 때 사용하는 모듈이다.

개발 중 궁금한 내용이 있어 파이썬 문서를 참고하려 한다. 이를 위해 <https://python.org> 사이트를 새로운 웹 브라우저로 열려면 코드를 어떻게 작성해야 할까?



```
python.py
1 # webbrowser_test.py
2 import webbrowser
3
4 webbrowser.open_new('http://python.org')
```

파이썬으로 웹 페이지를 새 창으로 열려면 webbrowser 모듈의 open\_new() 함수를 사용해야 한다.

이미 열린 브라우저로 원하는 사이트를 열고 싶다면 다음처럼 open\_new() 대신 open()을 사용하면 된다.

```
python.py
1 # webbrowser_test.py
2 import webbrowser
3
4 webbrowser.open(['http://python.org'])
```

참고로 이 주소는 파이썬의 공식 다운로드 사이트이다.