

# 2024 AI 정보영재교육원

## 중등영재캠프

[과정 2] IoT 프로그래밍

7.20 ~ 7.21

❖ 본 실험 강의는 아래와 같은 환경에서 진행되었습니다. ❖

❖ 실험 환경

- ✓ NUC 11TNHi5
- ✓ Raspberry Pi 4
- ✓ Raspberry Pi Camera 2

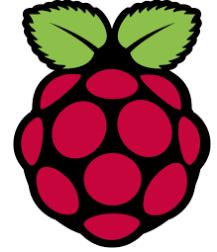
❖ 실험 OS

- ✓ Ubuntu 24.04 LTS
- ✓ Raspberry Pi OS Bookworm

# 1. Raspberry Pi & Linux 기초

- 강의 목표
  - Raspberry Pi 란?
  - 운영체제
  - Linux CLI 및 Vim 사용 방법
  - Linux 기본 명령어 학습
  - Python 이란?
  - 실험 환경 설정

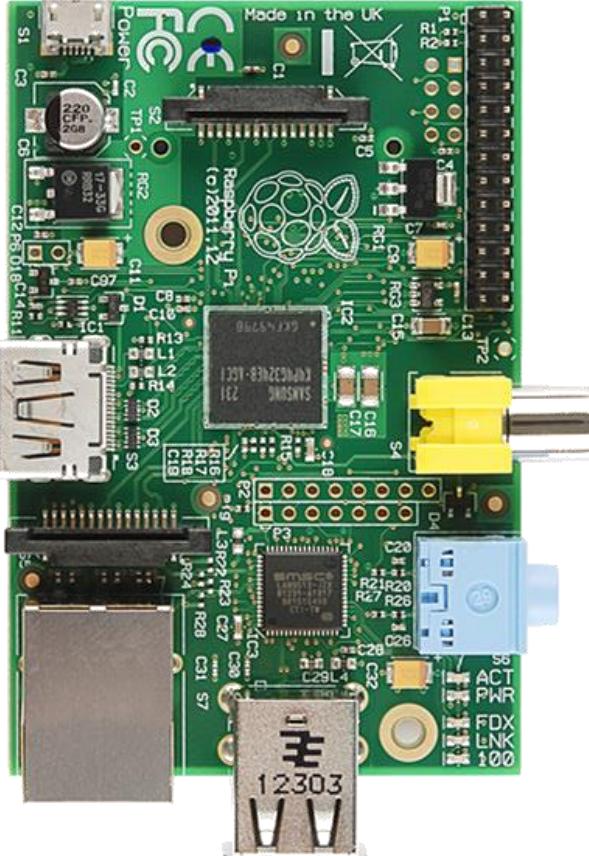
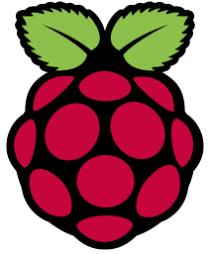
# Raspberry Pi란?



- 2012년 2월 영국 라즈베리파이 재단에서 교육적인 목적으로 제작 발표한 Smartphone 크기의 크기의 Single Board Computer
  - Single Board Computer : micro processor, memory, I/O 등이 포함된 단일 회로 보드
- 현재 Element14, RS Component, Egoman 사에 의해 제작
- 특징
  - 우수한 성능과 저렴한 가격 : Model A: \$25, Model B 및 2 : \$35
  - Linux를 사용한 편리한 개발 환경(Open platform)
  - 전 세계 수많은 사용자들이 참여하는 다양한 프로젝트/커뮤니티 활성화
  - Intel processor와 달리 Embedded platform에서 강세를 띠는 ARM processor를 사용
  - 많은 전력이 필요하지 않음
  - 기존 platform 위에 결합 가능한 Shield와 액세서리, 센서 등을 연결하여 구성 가능



# Raspberry Pi 초기 모델



Raspberry Pi 1

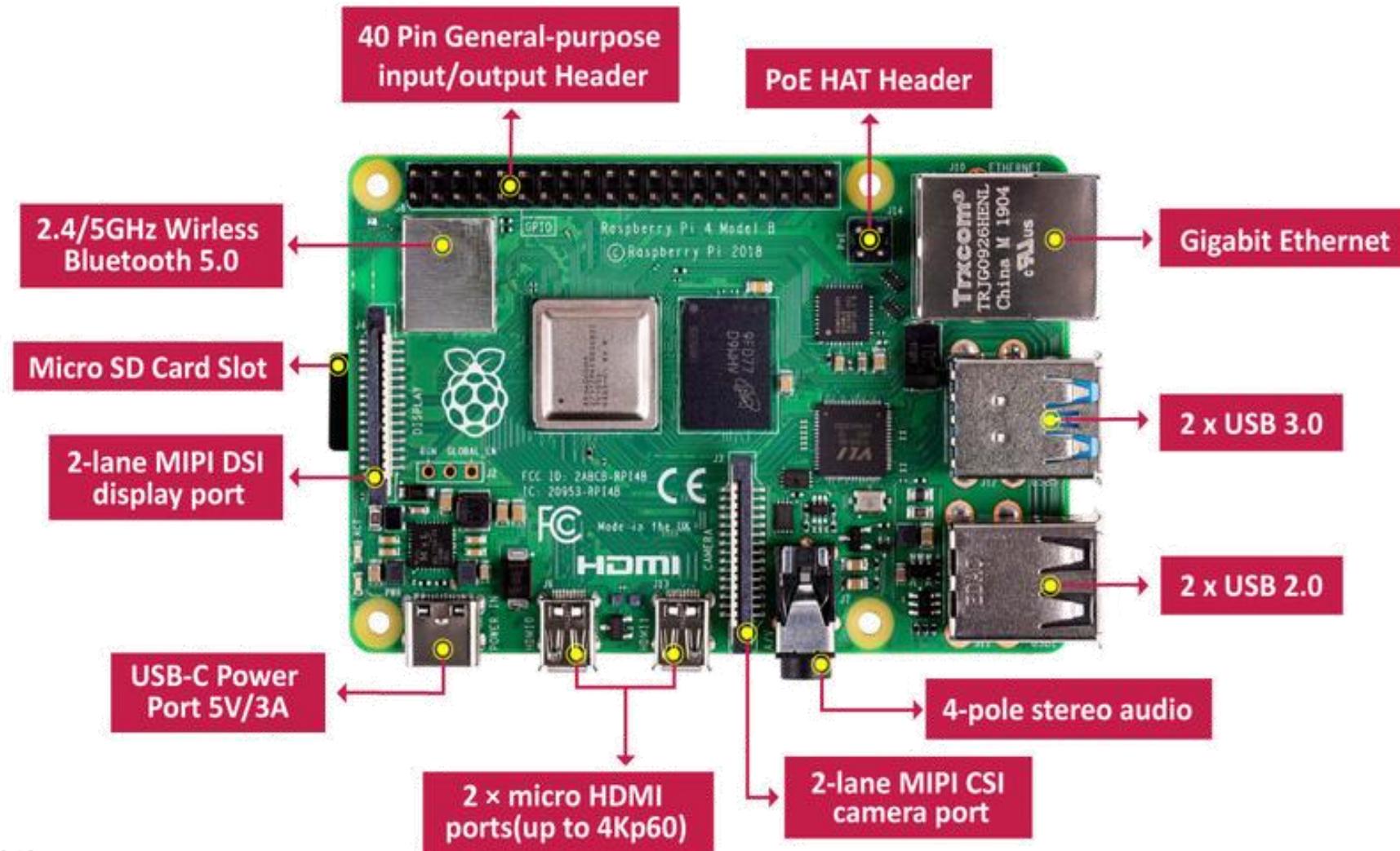
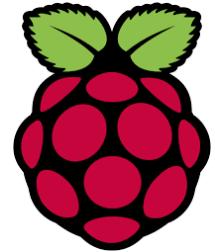


Raspberry Pi 2

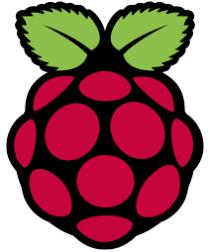


Raspberry Pi 3

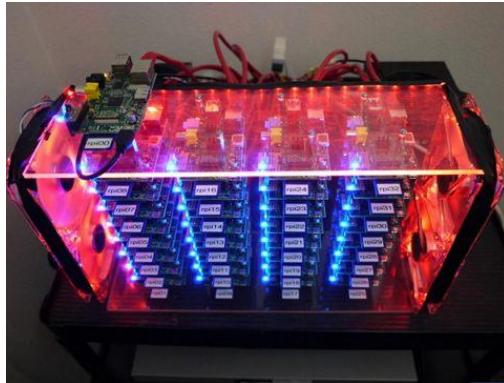
# Raspberry Pi 4 구조



# Raspberry Pi 응용



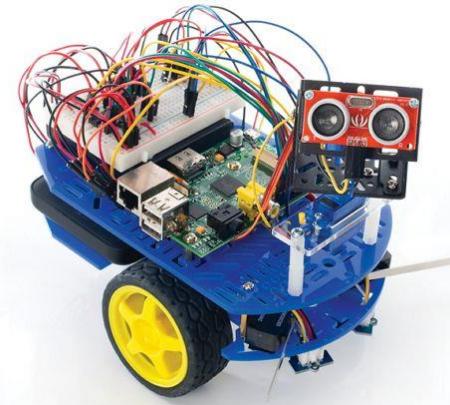
- The question isn't "**What can you do with a Raspberry Pi?**"
- The question is "**What can't you do with a Raspberry Pi?**"



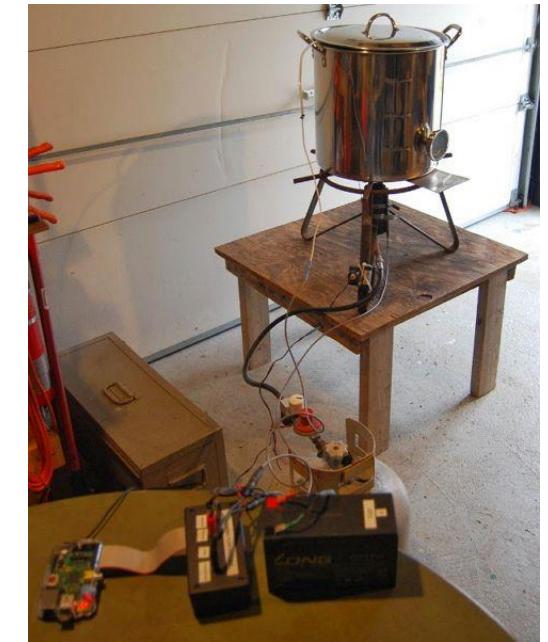
Raspberry Pi supercomputer



Raspberry Pi in the sky



Raspberry Pi Robot Project

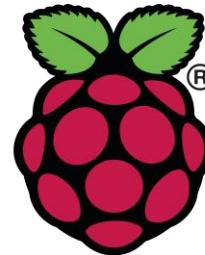


Raspberry Pi Home Brewing

# 운영체제란?

Raspberry Pi & NUC

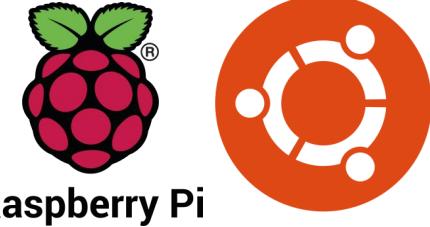
운영체제



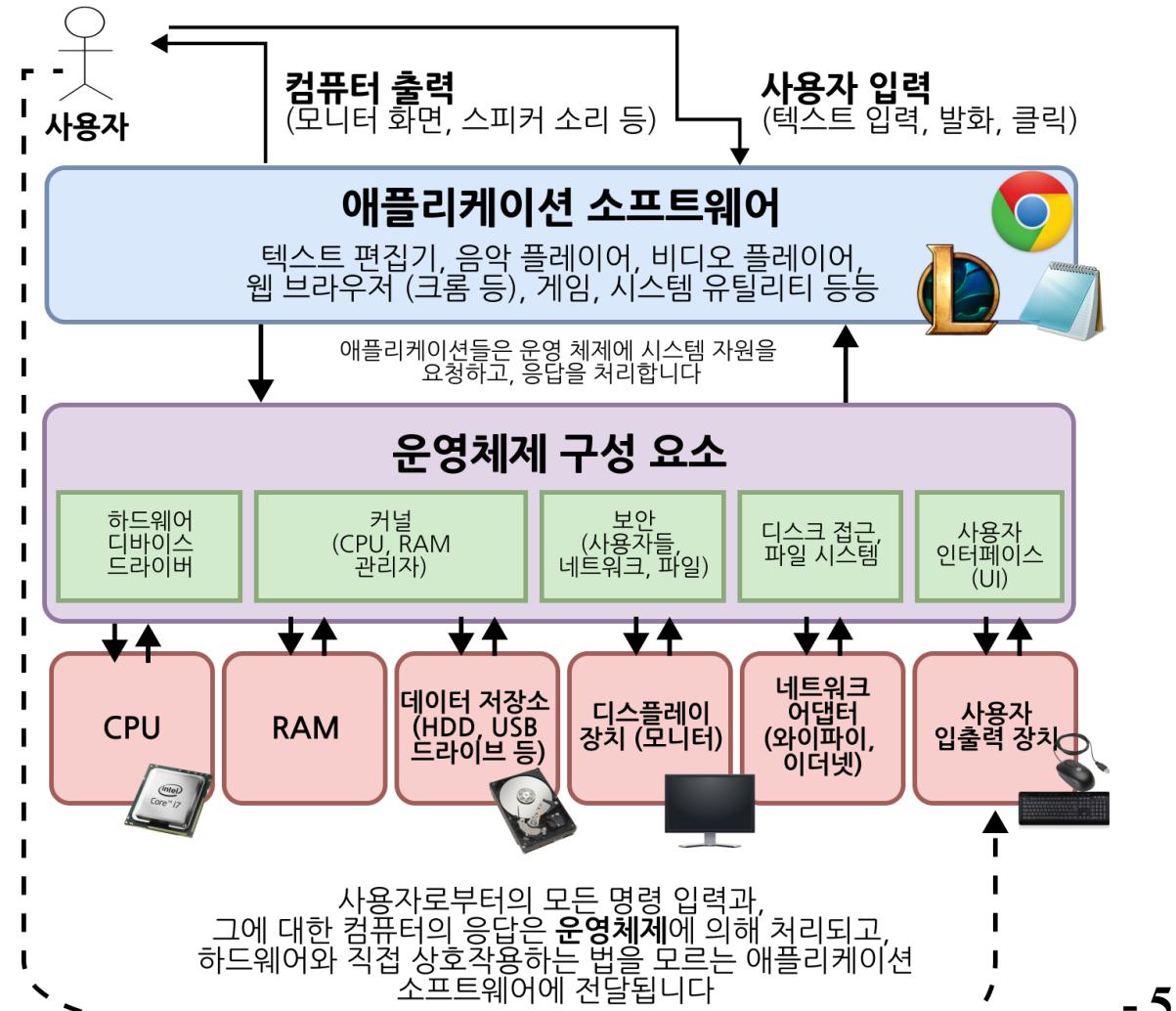
Raspberry Pi



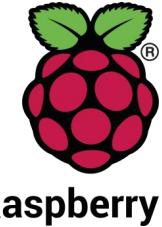
# 운영 체제(Operating System : OS)



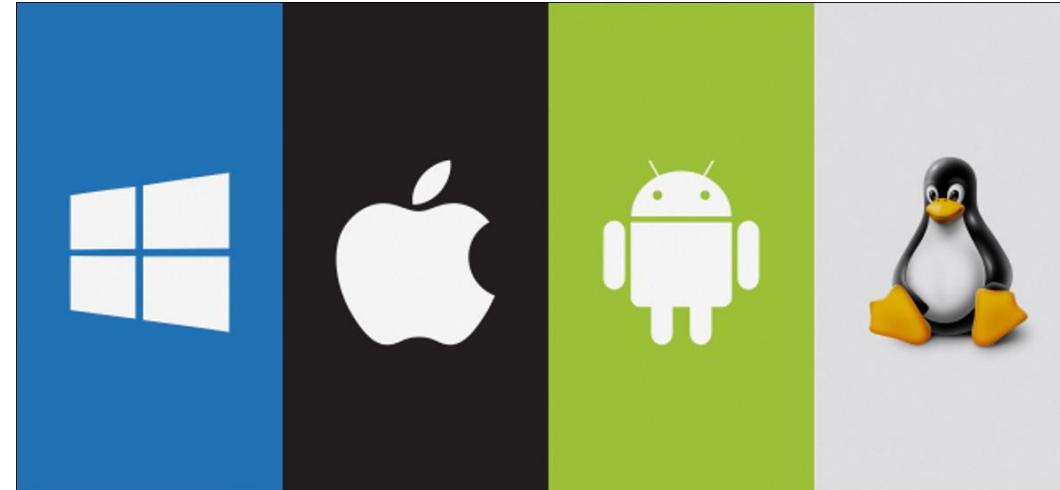
- 운영체제는 프로그래머가 컴퓨터 하드웨어를 모르더라도, 소프트웨어를 쉽게 만들 수 있도록 해줍니다.
- 운영체제의 발전 목적은 아래와 같습니다.
  - 컴퓨터 처리 능력 증대
  - 컴퓨터 응답 시간 단축
  - 사용 가능성 증대
  - 신뢰도 향상
- 운영체제는 여러 자원들을 제어하고, 상태를 감시합니다.



# 운영 체제(Operating System : OS)



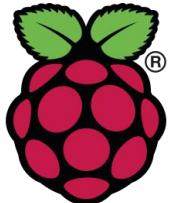
- 운영체제란, 컴퓨터 시스템의 하드웨어 자원과 소프트웨어 자원을 효율적으로 관리하여 사용자에게 편리함을 제공하는 프로그램
- 대표적으로 컴퓨터에는 Windows, Linux, UNIX 등의 운영체제가 있고, 모바일(핸드폰)의 대표적인 OS는 iOS와 안드로이드 OS



# 운영 체제(Operating System : OS)



- 실습에서 사용할 운영체제는 모두 Linux라는 운영체제의 한 종류로서 Raspberry Pi OS와 Ubuntu



Raspberry Pi

## Raspberry Pi OS

Raspberry Pi에는 자체적 운영체제로 라즈베리 파이의 저성능 CPU에 최적화가 잘 되어 있음



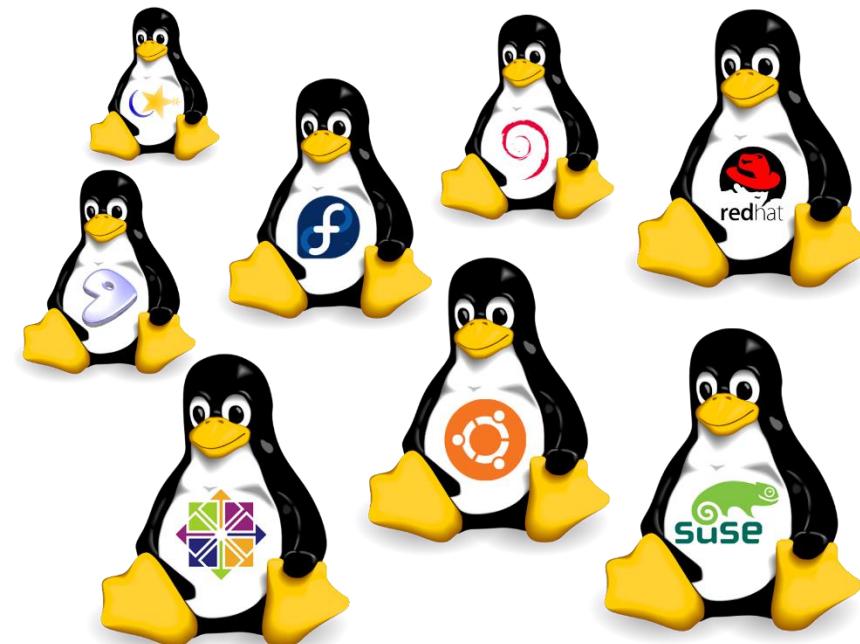
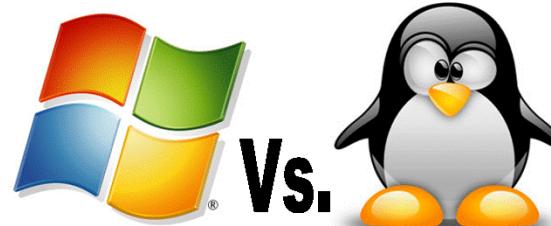
## Ubuntu

Linux를 기반으로 만들어졌으며, 개인용 데스크탑 환경에 최적화 되어 있으며 무료임

# Linux?

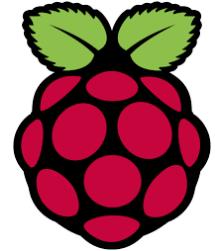


- 1991년 핀란드 헬싱키 대학에 다니던 Linus Benedict Torvalds 가 취미삼아 개발하던 Kernel을 인터넷에 공개 (Just for fun : Linux)
- Version 0.1 이라 불리는 Linux 운영체제 탄생
- 기본적으로 Unix 운영체제의 특징을 가지고 있으며, multi-user system
- 각각의 사용자는 자신의 계정(user account), password, home directory 라는 저장공간을 제공받으며, 계정별 접근 권한의 제약을 두어 기본적인 시스템 보안 제공
- 2000년대부터 Embedded 시장을 석권



많은 종류의 Linux

# Raspberry Pi 배포판



- 라즈비안 OS(Raspbian OS)
  - Raspberry Pi에서 가장 많이 사용하고 권장하는 Linux 배포판
  - Debian Linux 기반으로 경량 데스크탑 환경, 웹브라우저, 파이썬, 스크래치 등의 다양한 도구들을 제공
  - <http://www.raspberrypi.com>에서 공식 배포판을 다운로드

## Raspberry Pi OS

Your Raspberry Pi needs an operating system to work. This is it. Raspberry Pi OS (previously called Raspbian) is our official supported operating system.



### Raspberry Pi OS

Our recommended operating system for most users.

Compatible with:

[All Raspberry Pi models](#)

#### Raspberry Pi OS with desktop

Release date: March 15th 2024

System: 32-bit

Kernel version: 6.6

Debian version: 12 (bookworm)

Size: 1,158MB

[Show SHA256 file integrity hash](#)

[Release notes](#)

[Download](#)

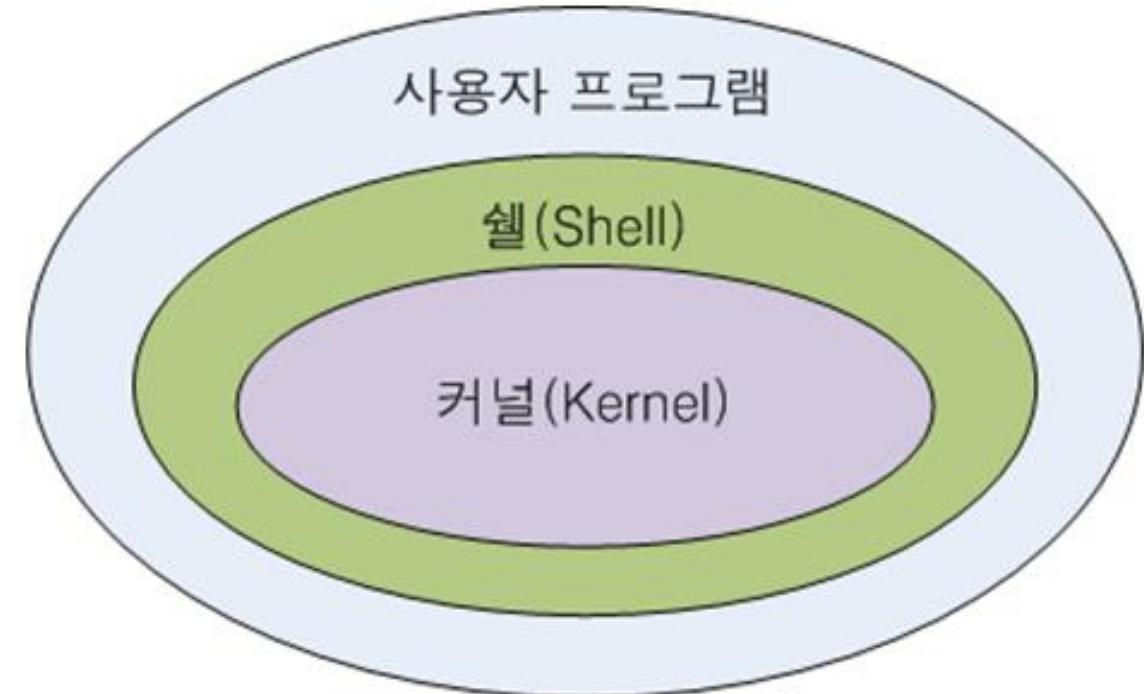
[Download torrent](#)

[Archive](#)



# 셸(Shell)

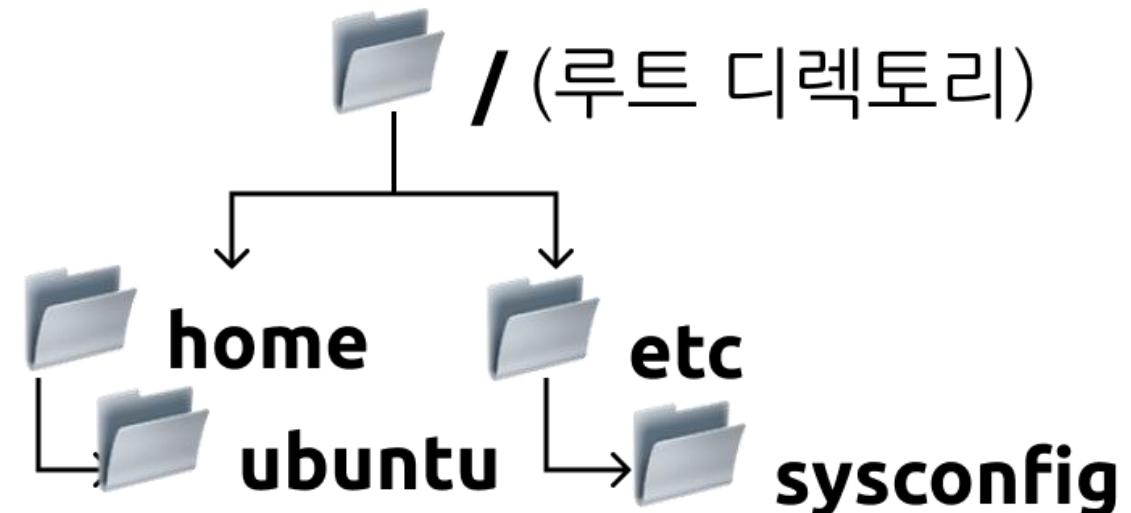
- 셸과 커널(Kernel)은 **운영체제의 한 부분**으로, 셸에서 상호작용하는 부분을 사용자 인터페이스 (User interface) 라고 합니다. 사용자 인터페이스는 크게 GUI (Graphic User Interface)와 CLI (Command Line Interface)로 나눌 수 있습니다.
- Shell Script** : 텍스트 형식으로 저장되는 프로그램으로, 컴퓨터는 shell script 파일에 저장되어 있는 명령어를 한 줄씩 차례로 읽어서 실행합니다.





# 파일 시스템(File System)

- 파일시스템(File System)은 운영체제가 저장되어 있는 파일과 폴더들을 효율적으로 관리하고 읽고 쓰기 위한 체계입니다.
- 파일(File)이란, 프로그램 또는 데이터 등과 같은 정보들의 집합입니다. 바이트의 연속적인 연결로, 디스크 등 기억 장소에 저장되어 있습니다. 디스크의 여러 파일들은 각자의 고유한 이름을 가짐으로써 구별됩니다.
- 디렉토리(Directory)란 파일들을 보다 효율적으로 관리하는 방법으로, 서로 연관된 파일들을 한곳에 저장할 수 있도록 합니다. 서류함처럼 관련된 파일들이 저장되는 장소를 제공합니다. 디렉토리도 파일의 일종입니다.



# CLI(Command Line Interface)란?



## CLI (Command Line Interface)

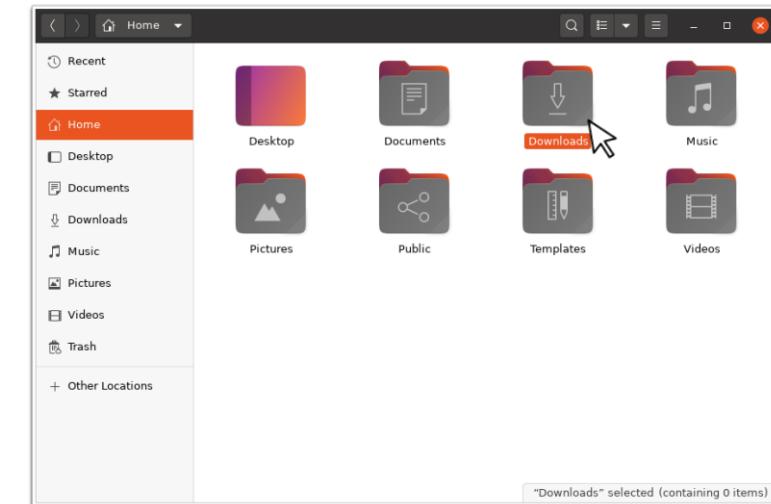
- 명령 줄 인터페이스는 텍스트로, 터미널(Terminal)을 통해 사용자와 컴퓨터가 상호작용하는 방식
- 사용자는 컴퓨터 키보드 등을 통해 텍스트를 입력하고, 컴퓨터 또한 문자열 형태로 결과를 보여줌

```
ubuntu@ubuntu-machine: ~
ubuntu@ubuntu-machine: ~ $ ls
Desktop Documnets Downloads Music Pictures Public Templates Videos
ubuntu@ubuntu-machine: ~ $ cd Downloads/
```

```
ubuntu@ubuntu-machine: ~/Downloads
ubuntu@ubuntu-machine: ~ $ ls
Desktop Documnets Downloads Music Pictures Public Templates Videos
ubuntu@ubuntu-machine: ~ $ cd Downloads/
ubuntu@ubuntu-machine: ~/Downloads $
```

## GUI (Graphic User Interface)

- 현재는 거의 모든 운영체제에서 GUI (Graphic User Interface)를 지원함
- 모니터에서 윈도우 창 등을 보며 대부분의 컴퓨터 작업을 진행할 수 있음





# Vim 소개

- **Vim(Vi improved)**은 Unix CLI 환경에서 사용할 수 있는 편집기
- 새로운 파일을 생성하고, 보고, 편집할 때 자주 이용함
- Vim에는 3가지 모드가 있음
  - Normal Mode: Vim을 실행하면 기본적으로 들어가는 모드입니다. 다른 모드에서 "esc" 키를 누르면 진입 가능
  - Insert Mode: 대표적으로 "i" 키를 누르면 입력 모드로 들어가, 일반적인 메모장처럼 글 작성 가능
  - Visual Mode: "v" 키를 누르면 들어갈 수 있음

```
ubuntu@ubuntu-machine: ~ $ vim test.txt
# test.txt 파일이 없다면 생성하고, 있다면 편집합니다.
```

# Vim 단축키



| Vim 명령어 단축키         |                 |                |              |               |             |              |            |            |               |            |            |            |            |        | 손에 잡히는 Vim 인사이트 |      |
|---------------------|-----------------|----------------|--------------|---------------|-------------|--------------|------------|------------|---------------|------------|------------|------------|------------|--------|-----------------|------|
| <b>Esc</b><br>명령 모드 | ~ 대소문자 전환       | ! 외부 명령        | @ 매크로 실행     | # 이전 검색       | \$ 행 끝으로 이동 | % 짹 글호 찾기    | ^ 첫 글자     | & 반복       | *             | 다음 검색      | ( 문장 시작    | ) 문장 끝     | ) 행 앞으로    | - 아래 행 | +               | 다음 행 |
| ‘ 표식으로 이동           | 1 ②             | 2              | 3            | 4             | 5           | 6            | 7          | 8          | 9             | 0          | - 이전 행     | -          | -          | =      | =               | =    |
| Q 실행 모드             | W 다음 단어 (의미상)   | E 단어 (의미상) 끝으로 | R 수정 모드      | T 행에서 후방 검색   | Y 행 단위 복사   | U 행 단위 실행 취소 | I 행 시작에 삽입 | O 행 위에 삽입  | P 커서 이전에 불여넣기 | { 문단 시작    | ]          | { 문단 끝     |            |        |                 |      |
| q· 매크로 기록           | w 다음 단어         | e 단어 끝으로       | r 한 문자 교체    | t· 행에서 전방 검색  | y 복사 ①③     | u 실행 취소      | i 커서 앞에 삽입 | o 행 아래에 삽입 | p 커서 이후에 불여넣기 | [          |            |            |            |        |                 |      |
| A 행 끝에 삽입           | S 행 입력 모드       | D 행 삭제         | F· 행에서 후방 검색 | G 파일 끝으로 이동   | H 화면 상단     | J 아래 행 합치기   | K 도움말      | L 화면 하단    | :             | 명령행 모드     | ;" 레지스터 지정 | ;" 표식으로 이동 | ;" 사용하지 않음 |        |                 |      |
| a 커서 뒤에 삽입          | s 단어 삭제 후 입력 모드 | d 삭제 ①③        | f· 행에서 전방 찾기 | g· 확장 ⑥ 명령    | h ←         | j ↓          | k ↑        | l →        | ;             | t/T/f/F 반복 |            |            |            |        |                 |      |
| Z· 종료 ④             | X 백스 페이스        | C 끝까지 비꾸기      | V 비주얼 라인 모드  | B 이전 단어 (의미상) | N 이전 (찾기)   | M 화면 가운데     | < 내어 쓰기    | > 들여 쓰기    | ?             | 찻기 (위로)    |            |            |            |        |                 |      |
| z· 확장 명령 ⑤          | x 글자 삭제         | c 바꾸기 ①③       | v 비주얼 모드     | b 이전 단어       | n 다음 (찾기)   | m 표식 설정      | ,          | ,          | ./            | 찾기 (아래로)   |            |            |            |        |                 |      |



# Vim의 단축키, 노말 모드에서의 명령어

## 입력 관련

- a** - 텍스트를 현재 커서 위치 뒤에 추가합니다. 입력 모드로 전환됩니다
- i** - 텍스트를 현재 커서 앞에 추가합니다. 입력 모드로 전환됩니다
- c** - 변경합니다. cw를 입력하면 커서부터 한 단어를, \$를 입력하면 커서부터 문장의 끝까지 변경합니다. 이외에도 여러 옵션이 있습니다
- d** - 삭제합니다. dd - 한 문장을 삭제합니다. d3 - 3문장을 삭제합니다
- y** - 복사합니다. yy - 한 문장을 복사합니다 y3 - 3문장을 복사합니다
- p** - 붙여넣습니다. d 또는 y 입력 후 캐싱한 내용을 붙여넣습니다

## 이동 관련

- \$** - 문장의 맨 끝으로 이동
- ^** - 문장의 맨 앞으로 이동
- gg** - 파일의 맨 앞으로 이동
- G** - 파일의 맨 끝으로 이동
- h, j, k, l** - 방향키

## 파일 저장, 찾기

- :w** - 파일을 저장합니다
- :wq** - 파일을 저장하고, vim을 끕니다
- /{어디있나}** - "어디있나"를 찾습니다

# Linux 기본 명령어(ls)



- ls : directory 안에 파일 목록 나열

```
ubuntu@ubuntu-machine: ~ $ ls                                # 현재 디렉토리의 파일 목록
ubuntu@ubuntu-machine: ~ $ ls /etc/sysconfig                # /etc/sysconfig/ 아래 디렉토리의 파일 목록
ubuntu@ubuntu-machine: ~ $ ls -a                            # 숨김 파일 목록 포함한 현재 디렉토리의 파일 목록
ubuntu@ubuntu-machine: ~ $ ls -l                            # 파일 정보까지 자세한 현재 디렉토리의 파일 목록
ubuntu@ubuntu-machine: ~ $ ls *.exe                         # 확장자가 exe인 현재 디렉토리의 파일 목록
ubuntu@ubuntu-machine: ~ $ ls -l /etc/sysconfig/a*        # /etc/sysconfig/ 안에 a로 시작하는 파일, 디렉토리 목록
```

# Linux 기본 명령어(cd)



- cd : directory 이동, 순회

```
ubuntu@ubuntu-machine: ~ $ cd                                     # 사용자의 홈 디렉토리로 이동
ubuntu@ubuntu-machine: ~ $ cd ..                                    # 상위 디렉토리로 이동
ubuntu@ubuntu-machine: ~ $ cd /etc/sysconfig                   # /etc/sysconfig 디렉토리로 이동
ubuntu@ubuntu-machine: ~ $ cd ../../etc/sysconfig
    # 상대 경로로, 상위 디렉토리로 간 뒤, 상위 디렉토리 아래의 /etc/config로 이동
    # ".." 은 현재 디렉토리 바로 위를 가리킴
ubuntu@ubuntu-machine: ~ $ pwd                                     # 지금 있는 디렉토리(위치)를 출력함
/home/ubuntu
```

# Linux 기본 명령어(mkdir, rmdir)



- mkdir, rmdir : directory 생성, 삭제

```
ubuntu@ubuntu-machine: ~ $ mkdir abc
```

abc라는 디렉토리 생성

```
ubuntu@ubuntu-machine: ~ $ mkdir -p /ab/cde
```

ab 디렉토리 아래 cde 디렉토리 생성, ab가 없으면 ab도 생성

```
ubuntu@ubuntu-machine: ~ $ rmdir abc
```

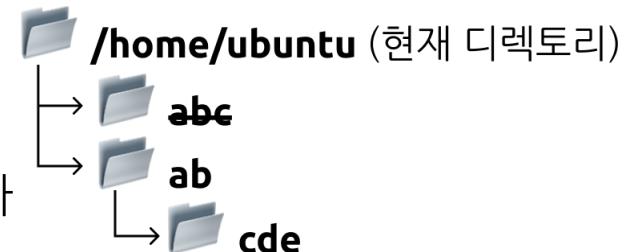
# 비어 있는 디렉토리 abc 삭제

디렉토리(폴더)를 생성하고, 삭제한 뒤의 모습입니다.

하지만, rmdir로 ab를 삭제하는 명령을 입력해도,

rmdir: failed to remove 'ab': Directory not empty

위와 같은 오류가 납니다. ab 아래에 cde라는 디렉토리가 있어 비어 있지 않기 때문입니다



# Linux 기본 명령어(touch, cp, mv)



- touch : 파일이 없다면 새로 생성

```
ubuntu@ubuntu-machine: ~ $ touch test.txt  
# 빈 test.txt 파일 생성, 이미 있을 시 수정시간만 변경
```

- cp, mv : 파일, 디렉토리 복사 및 이동

```
ubuntu@ubuntu-machine: ~ $ cp test.txt test2.txt  
test.txt 파일을 test2.txt로 복사
```

```
ubuntu@ubuntu-machine: ~ $ cp -r ab ab2  
ab 디렉토리를 ab2 디렉토리로 하위 파일 포함 복사
```

```
ubuntu@ubuntu-machine: ~ $ mv test.txt ./ab  
test.txt를 ab 디렉토리로 이동
```

```
ubuntu@ubuntu-machine: ~ $ mv test.txt test2.txt ./ab  
test.txt 와 test2.txt를 ab 디렉토리로 이동
```

```
ubuntu@ubuntu-machine: ~ $ mv test3.txt practice.txt  
test3.txt 파일을 practice.txt로 변경
```

# Linux 기본 명령어(rm)



- rm : 파일, 디렉토리 삭제

```
ubuntu@ubuntu-machine: ~ $ rm test.txt
```

test.txt 파일을 삭제, 기본적으로 아래 -i 옵션이 적용됨

```
ubuntu@ubuntu-machine: ~ $ rm -i test.txt
```

test.txt 파일을 삭제하기 전에 한번 더 확인

```
ubuntu@ubuntu-machine: ~ $ rm -f test.txt
```

test.txt를 파일을 묻지 않고 삭제

```
ubuntu@ubuntu-machine: ~ $ rm -r ab
```

디렉토리 ab 삭제

```
ubuntu@ubuntu-machine: ~ $ rm -rf abc
```

디렉토리 abc 묻지 않고 삭제

# Linux 기본 명령어(cat, head, tail)



- cat, head, tail : 파일 미리보기

```
ubuntu@ubuntu-machine: ~ $ cat test.txt
```

test.txt 파일을 내용 미리 보기

```
ubuntu@ubuntu-machine: ~ $ head test.txt
```

test.txt의 첫 10줄을 보여줍니다.

head -5 와 같은 형태로 몇 줄을 보일지 알 수 있습니다.

```
ubuntu@ubuntu-machine: ~ $ tail test.txt
```

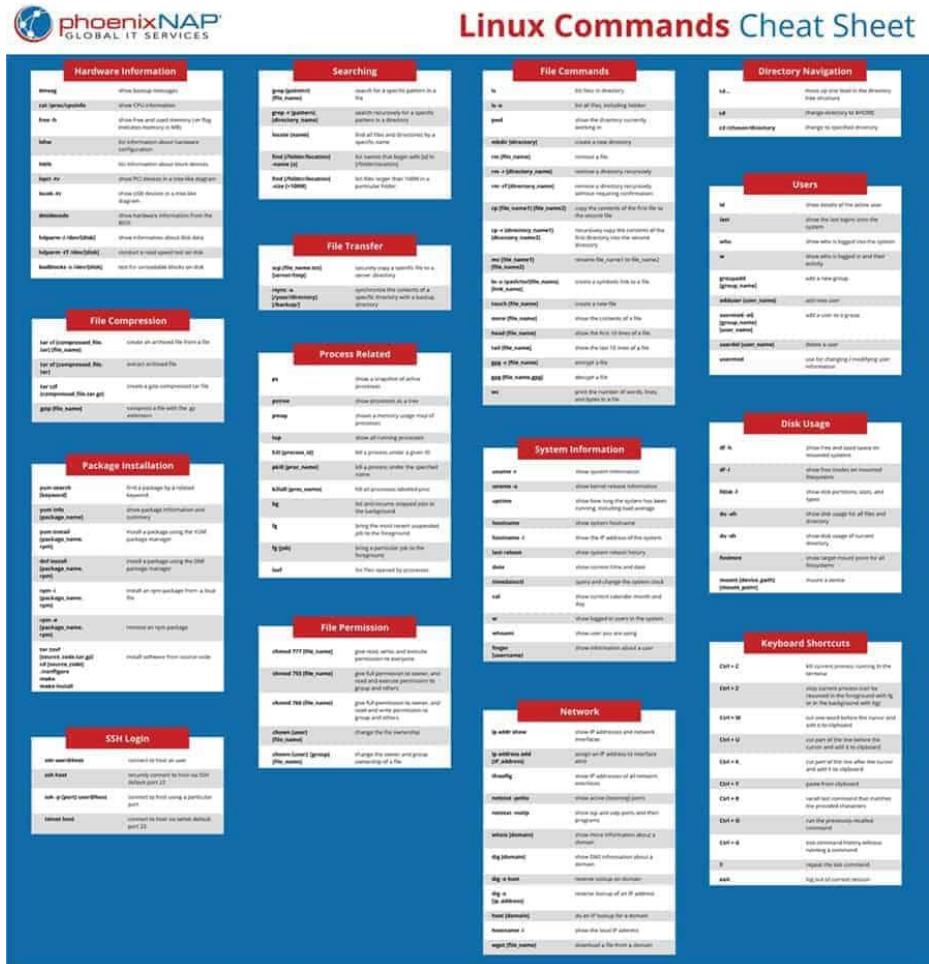
test.txt의 끝 10줄을 보여줍니다.

tail -5 와 같은 형태로 몇 줄을 보일지 알 수 있습니다.

# 리눅스 명령어 찾는 방법 / 설명 보기



- 리눅스에는 많은 명령어들이 있음
  - Linux command line cheat sheet 같은 자료를 참조
- 명령어에 대한 설명을 보고 싶으면 다음 커맨드를 사용
  - man 명령어
  - help 명령어





# Python?

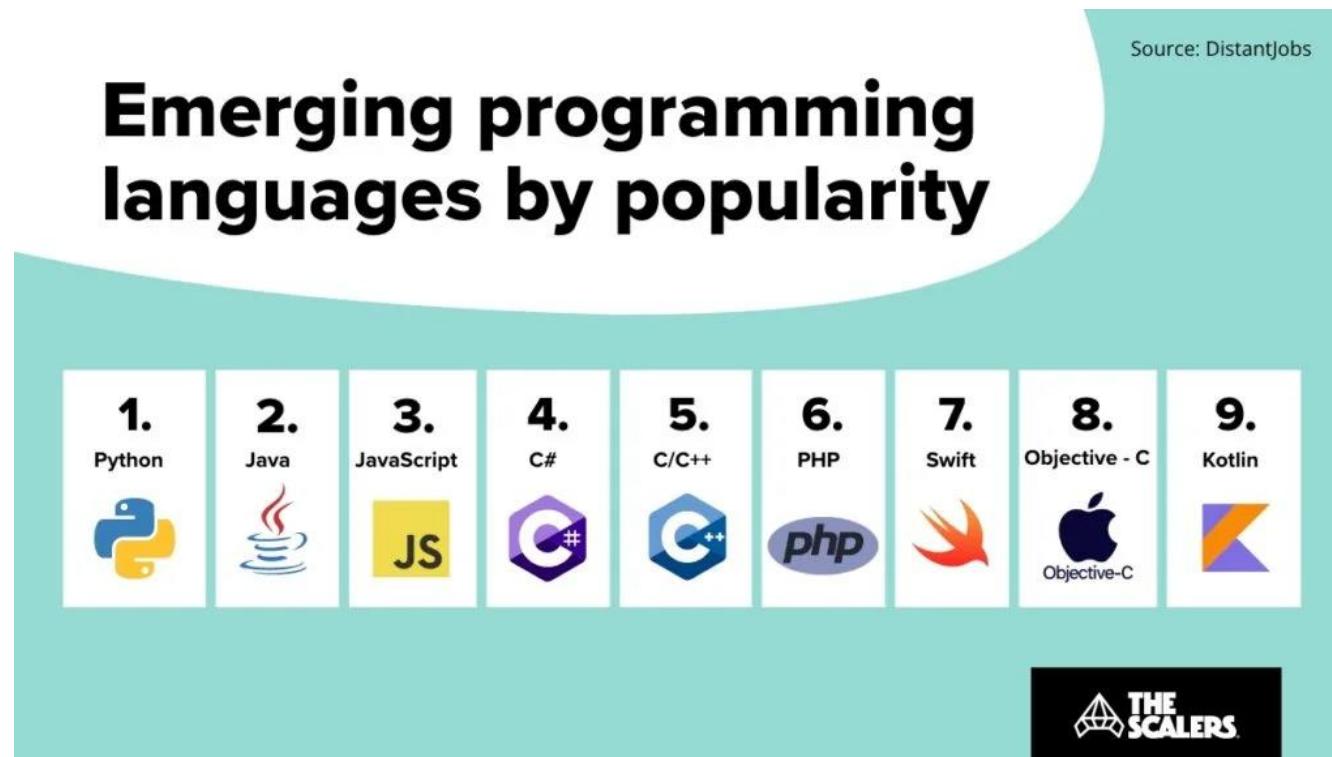
- 1991년 Guido Van Rossum 이 만든 프로그래밍 언어
  - 1989년 본격적으로 개발 착수
  - 2005년 Google에 합류, python 관련 project 진행
- 특징
  - C, JAVA 등과 같은 고급 언어이면서도 interpreter 언어
  - 프로그램 전체를 미리 작성할 필요 없이 프로그램 코드를 작성하면서 동시에 실행 가능
  - 문법적으로 매우 쉽고 직관적이어서 초보자도 쉽게 익힐 수 있음
  - 코드가 간결하며 이해하기 쉬움
  - 함수형 프로그래밍 방식도 지원하고 객체 지향 클래스도 지원
  - Machine learning, Graphics, Web programming 등 여러 분야에서 선호하는 언어

```
def add5(x):  
    return x+5  
  
def dotwrite(ast):  
    nodename = getNodeName()  
    label=symbol.sym_name.get(int(ast[0]),ast[0])  
    print '%s [%s=%s]' % (nodename, label),  
    if isinstance(ast[1], str):  
        if ast[1].strip():  
            print '= %s';' % ast[1]  
        else:  
            print '';  
    else:  
        print '';  
    children = []  
    for n, child in enumerate(ast[1:]):  
        children.append(dotwrite(child))  
    print ', %s -> {' % nodename  
    for n in range(len(children)):  
        print '%s' % children[n],
```

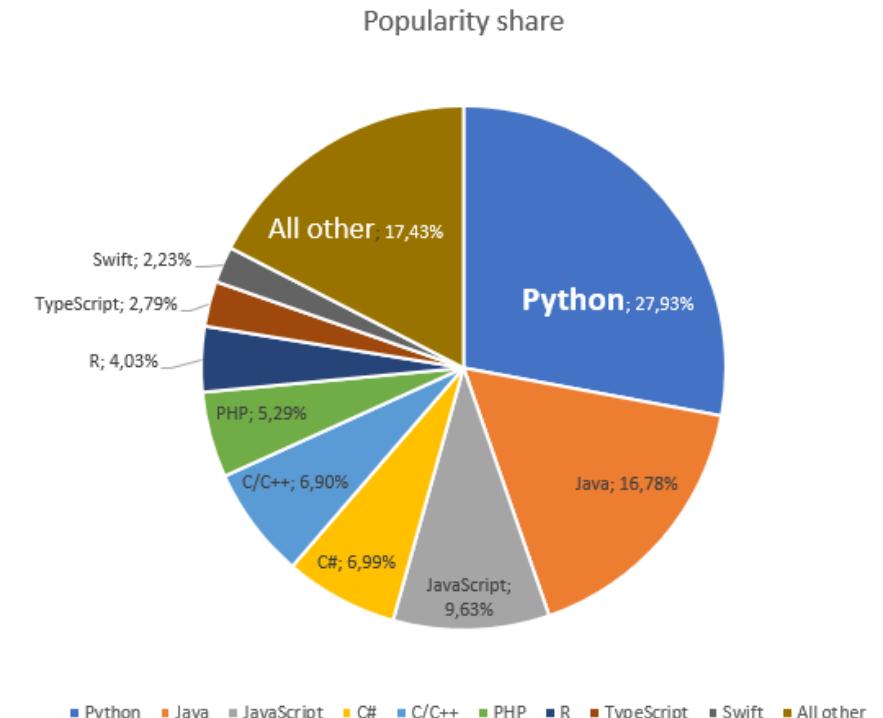


# Python?

- Top programming languages for your business



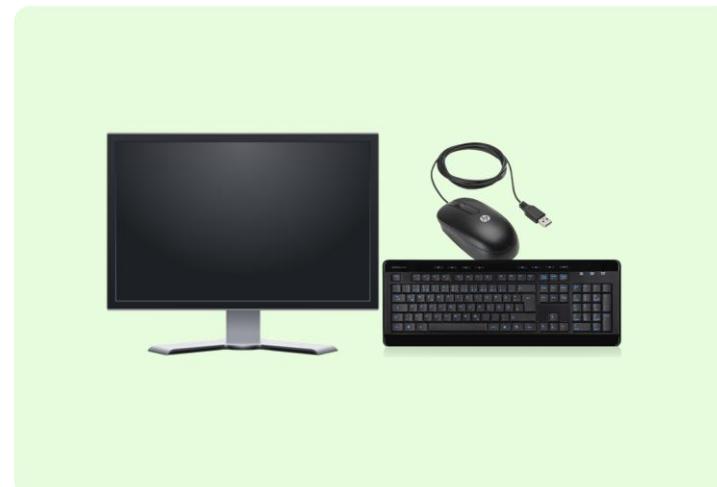
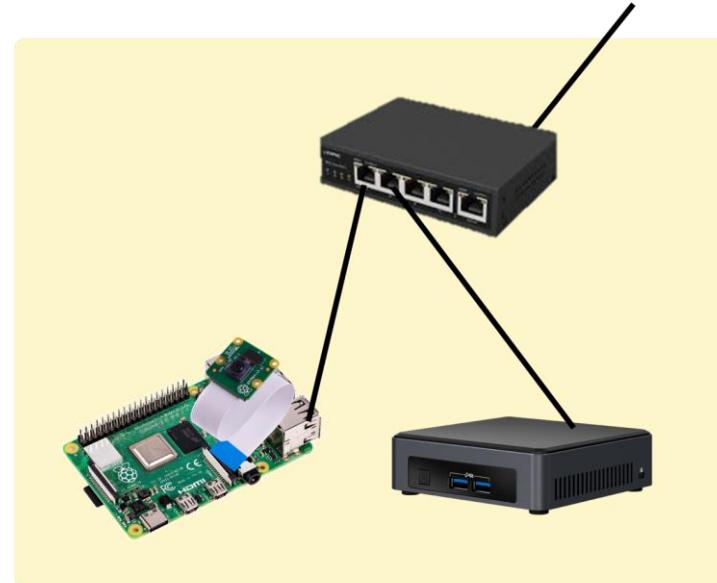
- 출처 : <https://distantjobs.com/>



- 출처 : <https://hitech-us.com/>

# 실험 환경

-  실험실의 인터넷 회선, 스위치와 Raspberry Pi, NUC는 연결되어 있음
- 스위치를 사용해 Raspberry Pi와 NUC 가 서로 연결될 수 있고 인터넷에도 연결 가능
-  Raspberry Pi, NUC에 연결할 모니터, 키보드, 마우스가 준비되어 있음



# 실험 환경 설정

Pi에서 실행



- Raspberry Pi 기본 설정

```
pi@raspberrypi: ~ $ sudo apt update  
pi@raspberrypi: ~ $ sudo apt upgrade  
pi@raspberrypi: ~ $ sudo apt install vim
```

- Repository 다운(강의 자료, python code)

```
pi@raspberrypi: ~ $ git clone https://github.com/minjunj/genius_camp.git
```

# 실험 환경 설정

NUC에서 실행



- NUC 기본 설정

```
ubuntu@ubuntu-machine: ~ $ sudo apt update
ubuntu@ubuntu-machine: ~ $ sudo apt upgrade
ubuntu@ubuntu-machine: ~ $ sudo apt install vim
ubuntu@ubuntu-machine: ~ $ sudo apt update
```

- Repository 다운(강의 자료, python code)

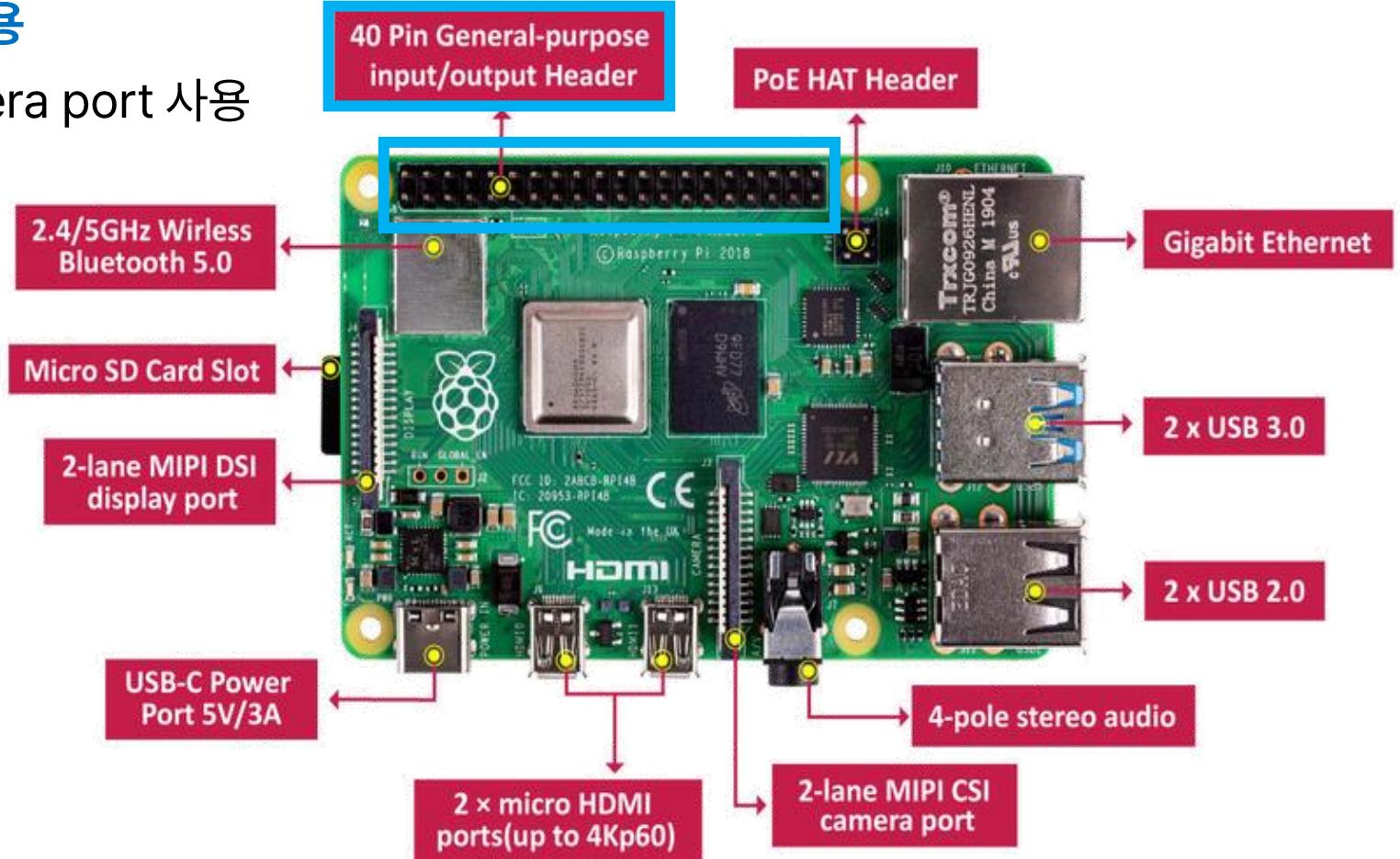
```
ubuntu@ubuntu-machine: ~ $ git clone https://github.com/minjunj/genius_camp.git
```

## 2. Raspberry Pi로 만드는 IoT

- 강의 목표
  - IoT(Internet of Things) 실습 소개
  - Raspberry Pi 의 GPIO pin
  - LED 출력 테스트
  - PWM을 이용한 LED 밝기 제어
  - RGB LED 제어
  - LED/Push Button 입출력 테스트
  - 초음파 센서를 이용한 거리 측정

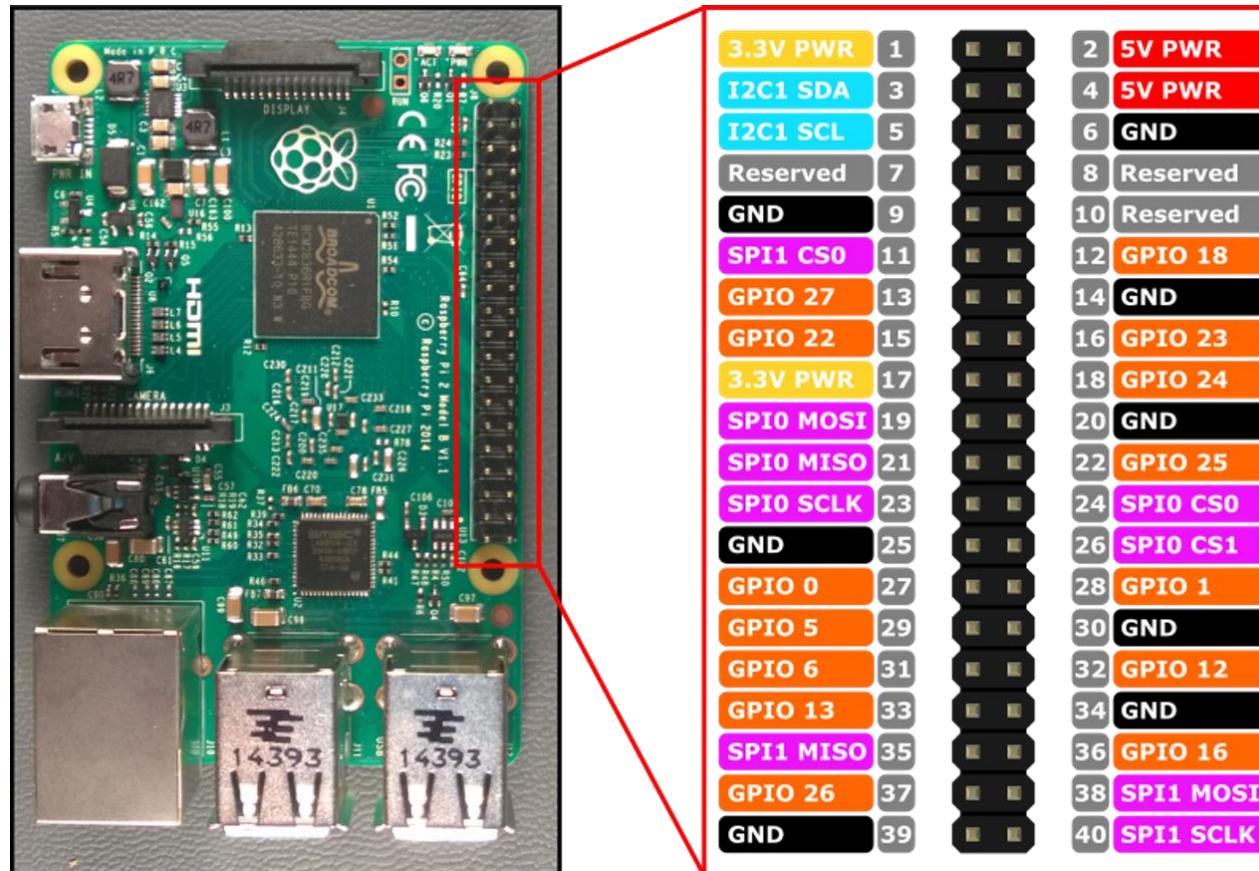
# Raspberry Pi로 만드는 IoT

- IoT 교육 : 40 Pin GPIO 사용
- Pi camera 교육 : CSI camera port 사용



# Raspberry Pi로 만드는 IoT

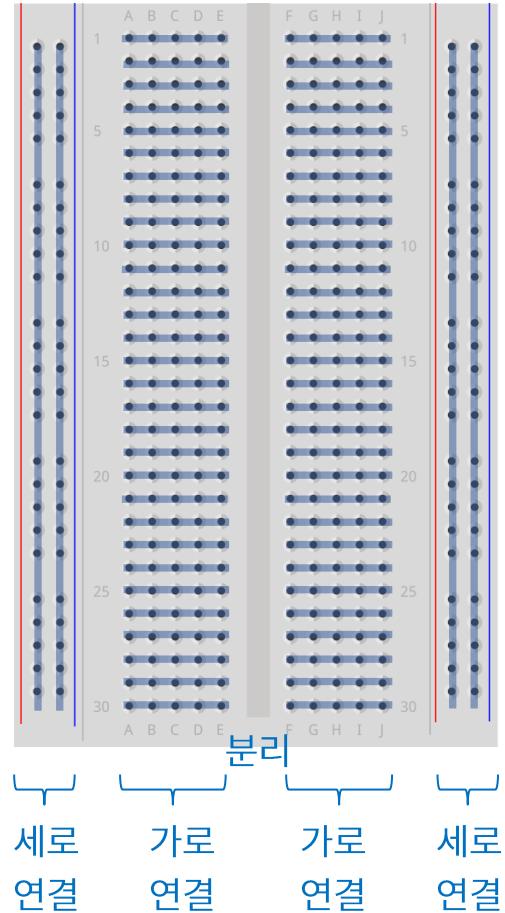
- 각종 센서, 모터, 카메라 등 수많은 장치들을 직접 연결할 수 있음
- Pi에서 제공되는 GPIO(General Purpose Input/Output) 포트를 이용



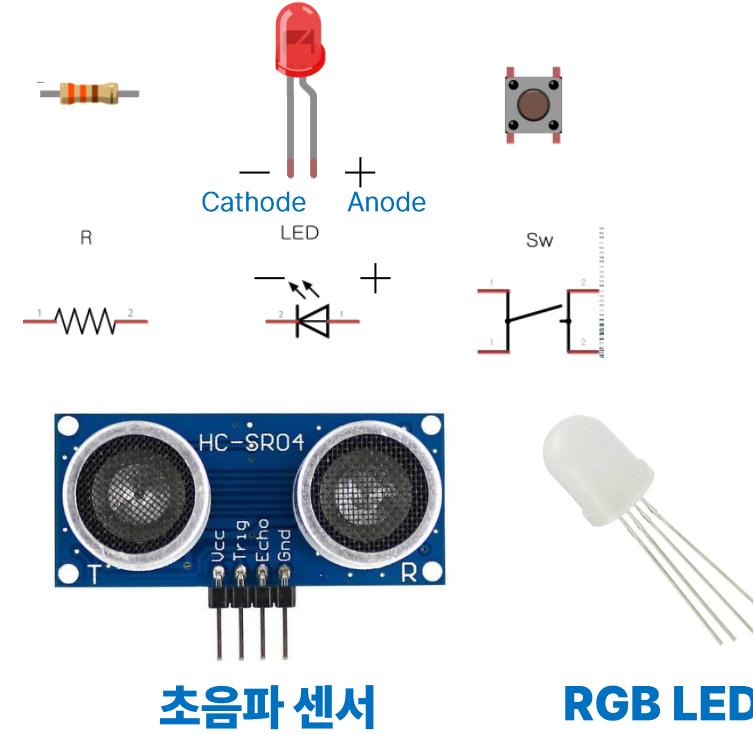
# Raspberry Pi로 만드는 IoT

## Bread Board

- 기본 실습 부품들



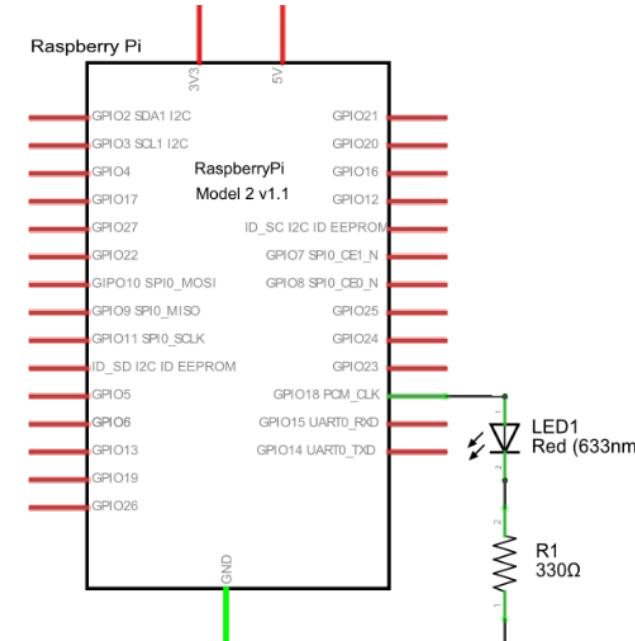
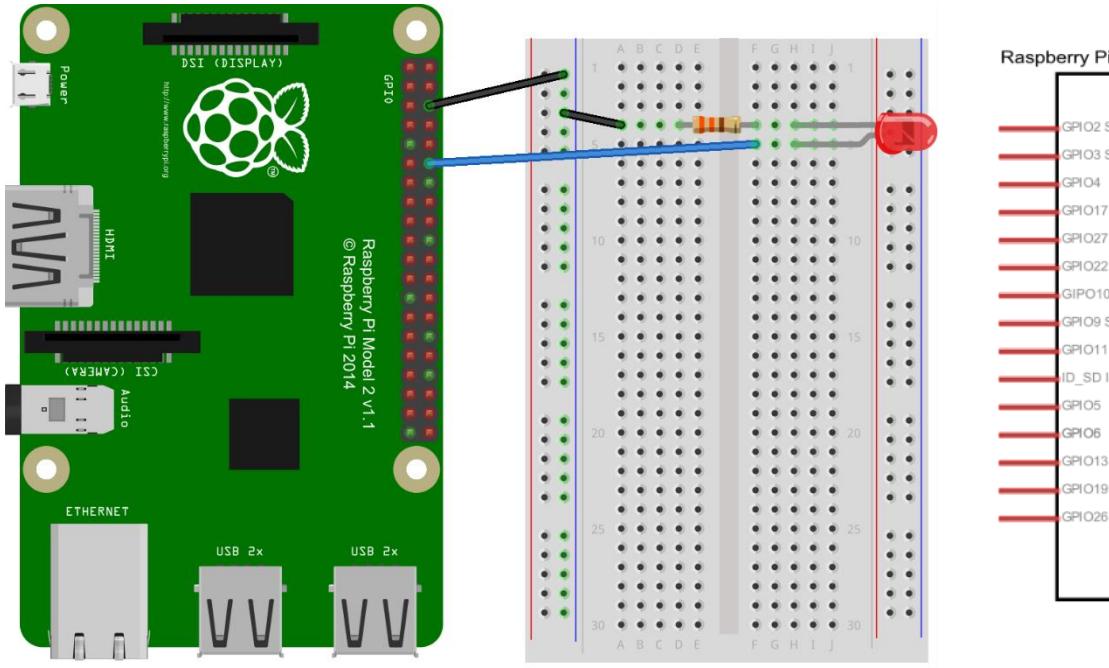
## Resistor LED push button



# Python을 이용한 LED 출력 테스트

Pi에서 실행

- 회로 구성 후 [blink.py](#) 스크립트 작성 후 실행



```
#blink.py

import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)

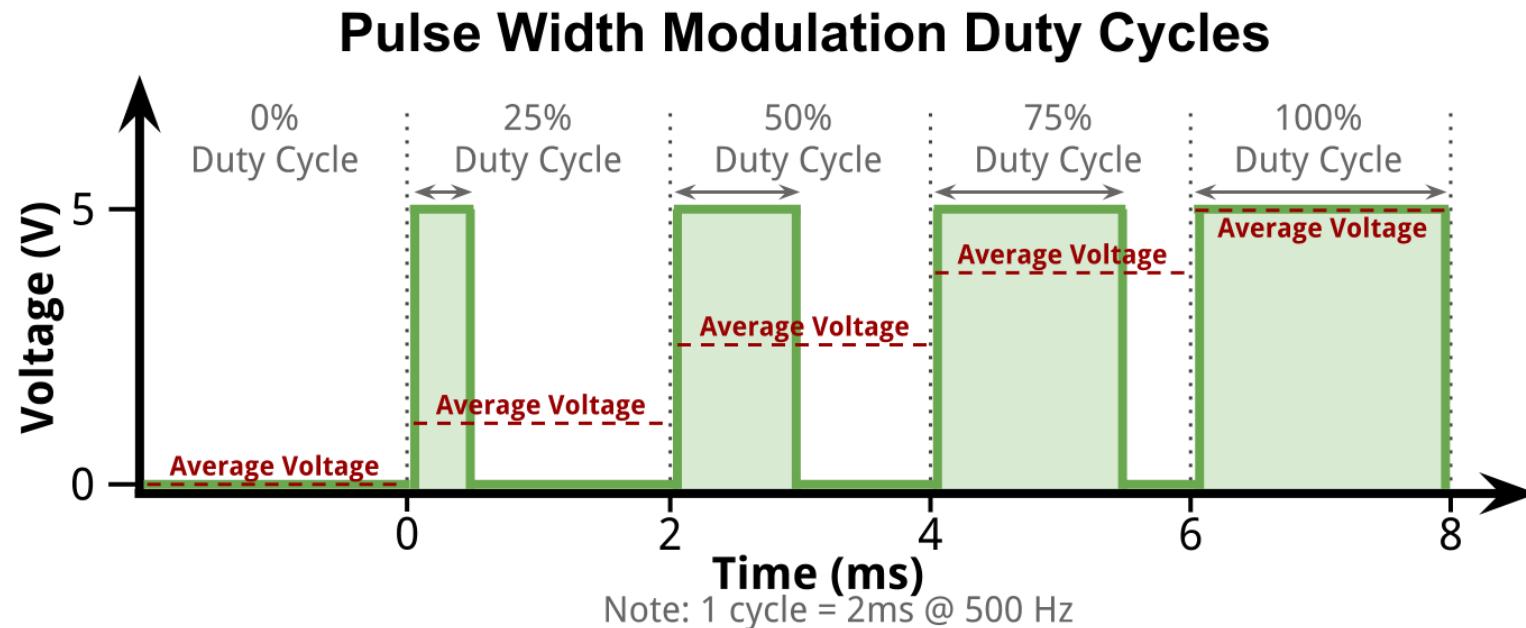
try:
    while (True):
        GPIO.output(18, True)
        time.sleep(1)
        GPIO.output(18, False)
        time.sleep(1)

except KeyboardInterrupt:
    GPIO.output(18, GPIO.LOW)
    GPIO.cleanup()
```



# PWM을 이용한 LED 밝기 제어

- Pulse Width Modulation(PWM) : 펄스 폭 변조
- 초당 pulse 전체수(Hz로 표시되는 주파수)를 유지한 채 pulse의 길이를 변화시키는 방법
- Digital을 Analog 방법으로 Motor 속도 또는 LED 밝기 제어 가능



# PWM을 이용한 LED 밝기 제어

- 위의 회로를 유지한 채 `led_brightness.py` 스크립트 작성 후 실행

```
#led_brightness.py

import RPi.GPIO as GPIO

LED = 18
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED, GPIO.OUT)

pwm_led = GPIO.PWM(LED, 500)
pwm_led.start(100)

try:
    while (True):
        duty_s = input("Enter Brightness (0 to 100):")
        duty = int(duty_s)
        pwm_led.ChangeDutyCycle(duty)

except KeyboardInterrupt:
    pwm_led.stop( )
    GPIO.cleanup( )
```

```
pi@raspberrypi: ~ $ python3 genius_camp/iot/led_brightness.py
```



# PWM을 이용한 LED 밝기 제어

Pi에서 실행



- 위의 회로를 유지한 채 [led\\_dimming.py](#) 스크립트 작성 후 실행

```
#led_dimming.py

import RPi.GPIO as GPIO
import time

LED = 18
GPIO.setmode(GPIO.BCM)
GPIO.setup(LED, GPIO.OUT)

LED = GPIO.PWM(LED, 100)
LED.start(0)
delay = 0.1
```

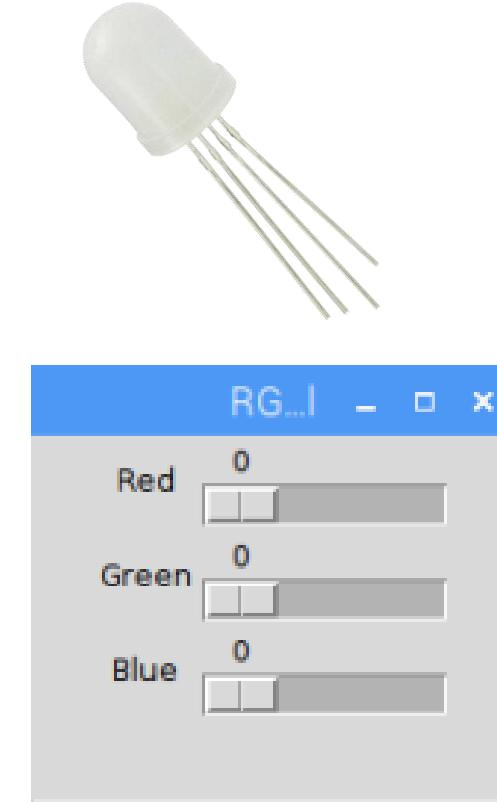
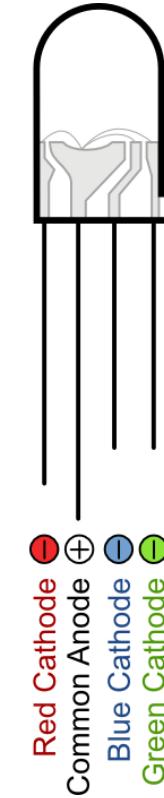
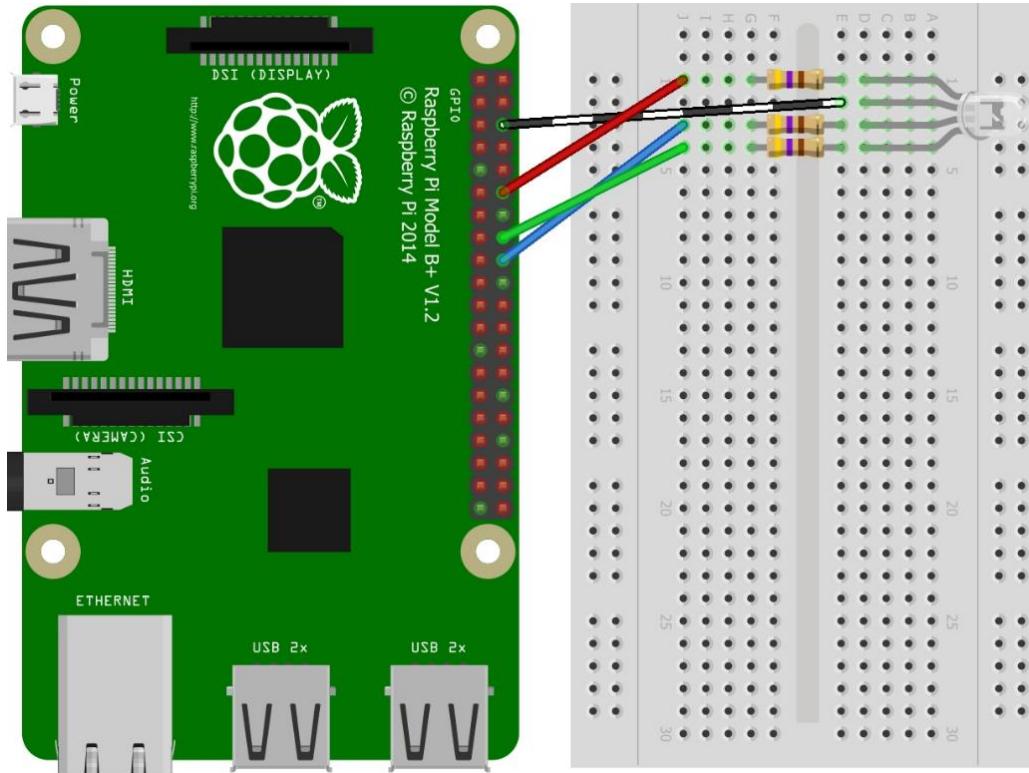
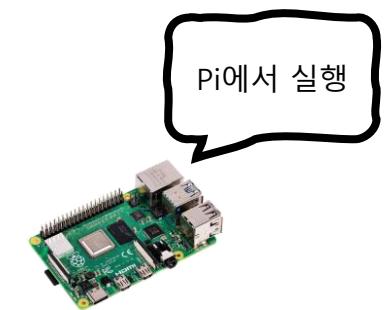
```
try:
    while (True):
        for i in range(0, 101):
            LED.ChangeDutyCycle(i)
            time.sleep(delay)
        for i in range(100, -1, -1):
            LED.ChangeDutyCycle(i)
            time.sleep(delay)

except KeyboardInterrupt:
    LED.stop()
    GPIO.cleanup()
```

```
pi@raspberrypi: ~ $ python3 genius_camp/iot/led_dimming.py
```

# RGB LED 제어

- PWM 출력으로 GPIO 18(Red), 23(Green), 24(Blue) pin 사용(mixing\_colors.py)



```
pi@raspberrypi: ~ $ python3 genius_camp/iot/mixing_colors.py
```

Scroll-bar를 움직이며 LED의 변화 확인

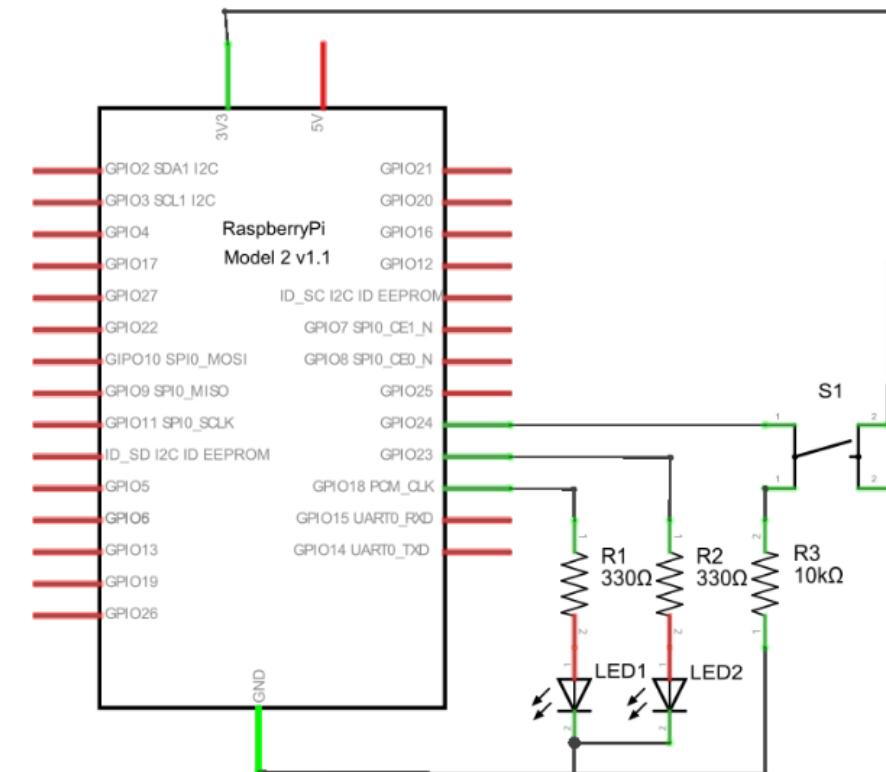
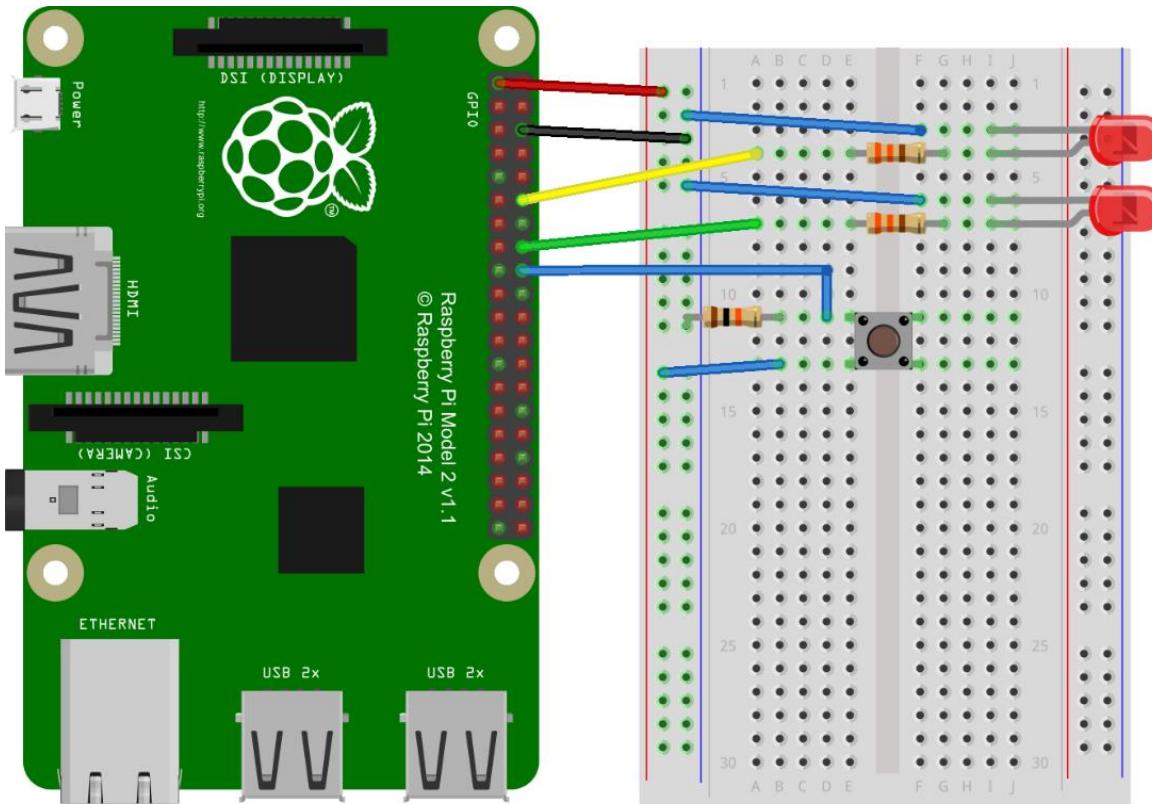
# LED/Push Button 입력 출력 테스트

Pi에서 실행



- Python Bottle Web Framework 설치

```
pi@raspberrypi: ~ $ sudo rm /usr/lib/python3.11/EXTERNALLY-MANAGED  
pi@raspberrypi: ~ $ sudo pip install bottle
```



# LED/Push Button 입출력 테스트

Pi에서 실행



- 실행 및 실행 결과

```
pi@raspberrypi: ~ $ python3 genius_camp/iot/ledbtn_web.py
```

- 결과 확인

- Web browser 실행
- <http://localhost:8080> 으로 접속
- 회로의 push button, LED0, LED1 버튼을 클릭하여 결과 확인

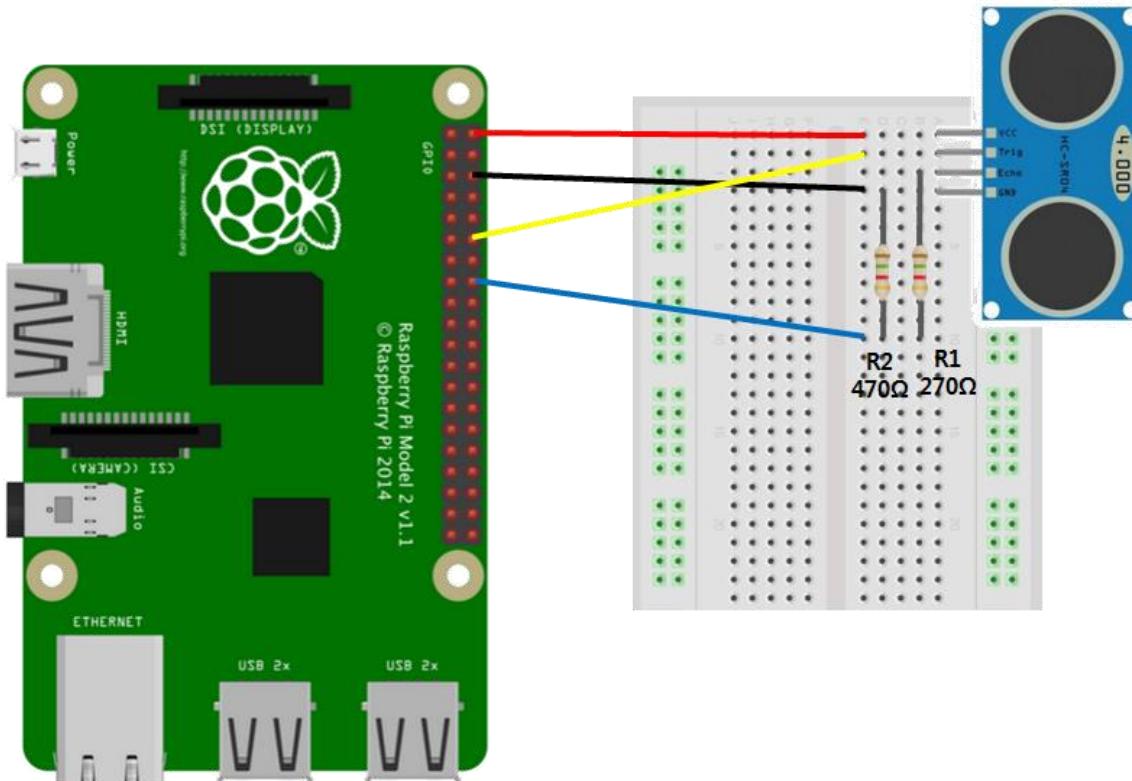


# 초음파 센서를 이용한 거리 측정

Pi에서 실행



- 사람 귀에 들리지 않는 주파수(20Khz 이상)의 소리인 초음파가 가지고 있는 특성을 이용한 센서



초음파 센서(HC-SR04)

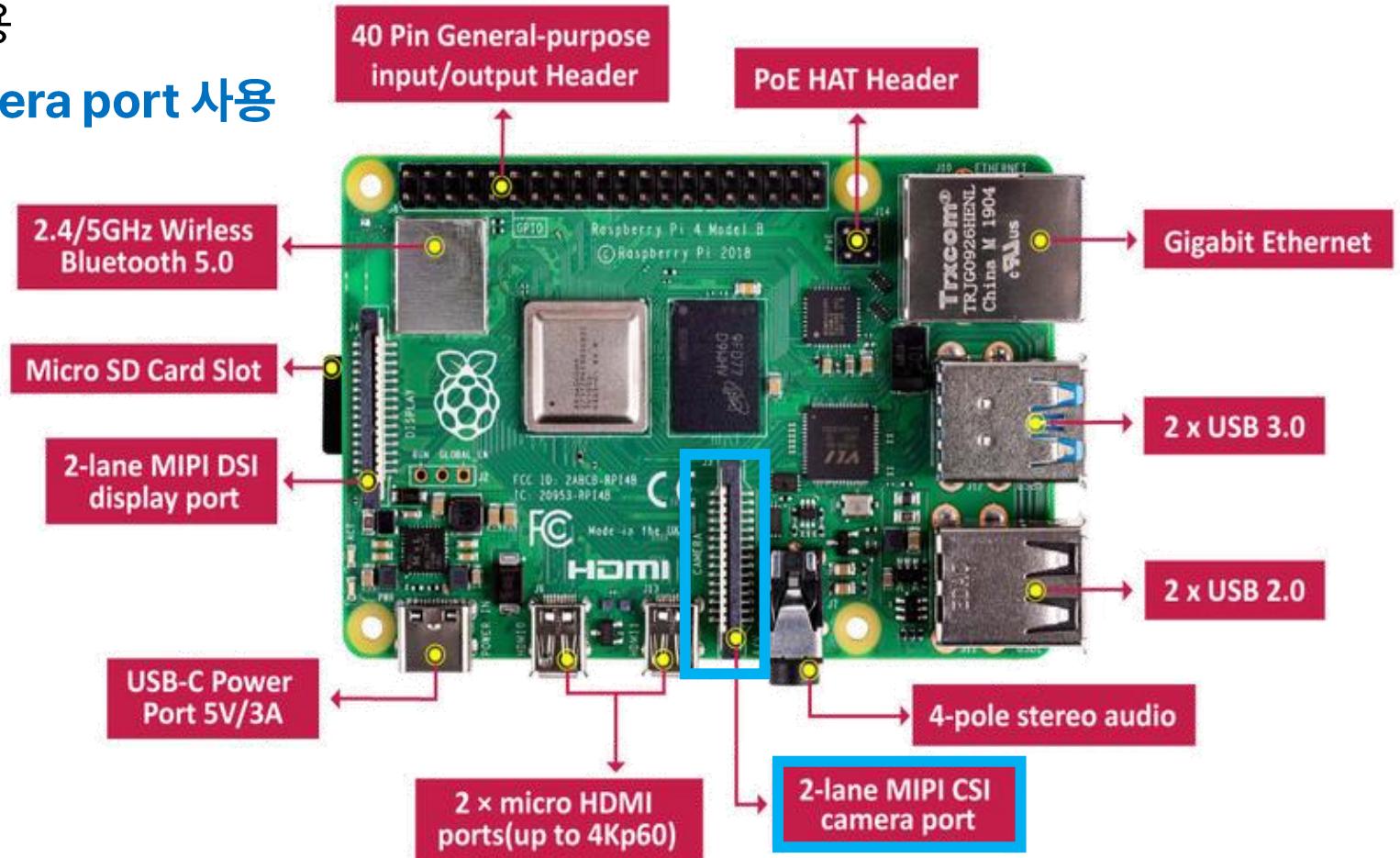
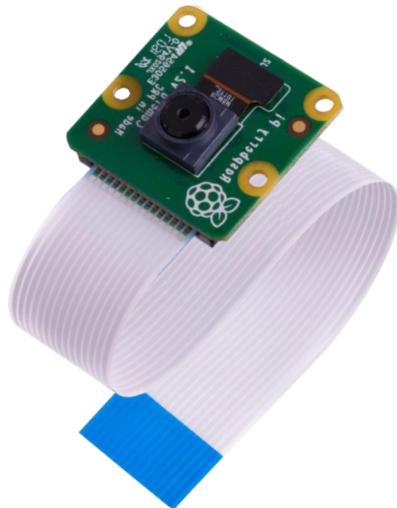
```
pi@raspberrypi: ~ $ python3 genius_camp/iot/ranger.py
```

# 3. Pi Camera를 이용한 영상 처리

- 강의 목표
  - Pi Camera 연결
  - Pi Camera 테스트 및 python을 이용한 영상 녹화
  - NATS 설정 및 테스트
  - NATS를 이용한 영상 전송
  - NATS를 이용한 객체 인식

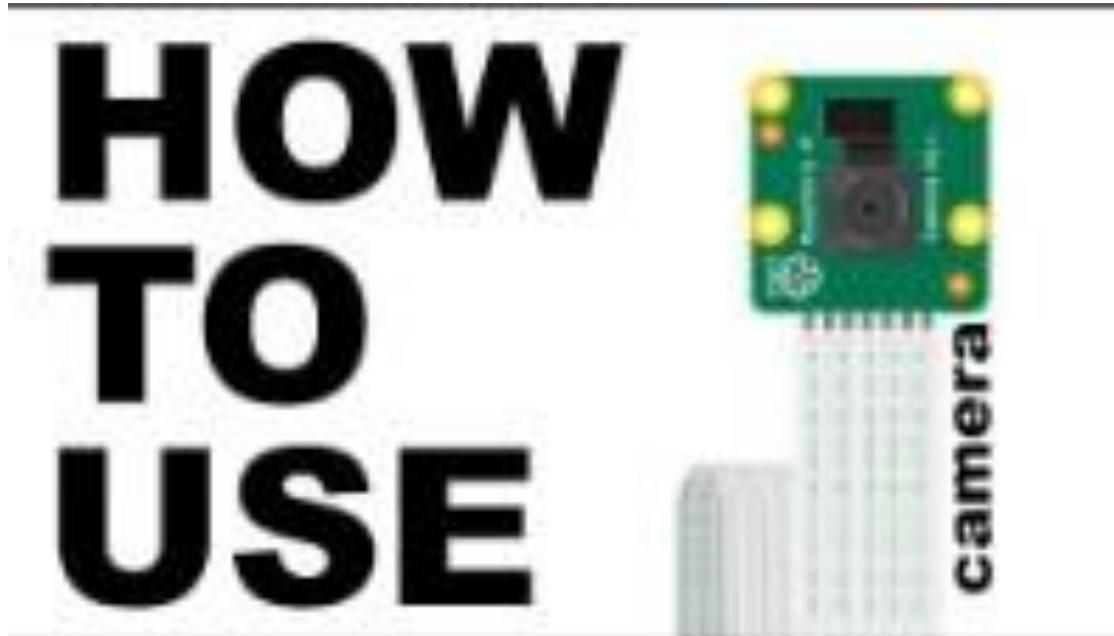
# Pi Camera를 이용한 영상 처리

- IoT 교육 : 40 Pin GPIO 사용
- Pi camera 교육 : CSI camera port 사용



# Pi Camera 연결

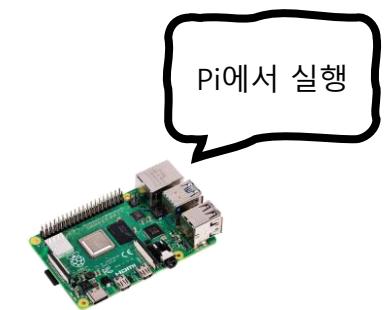
- Raspberry Pi의 전원을 끈 상태에서, 카메라 모듈을 연결



영상 설명 : <https://youtu.be/VzYGDq0D1mw>

# Pi Camera 테스트

- 재부팅 후, [camera.py](#) 스크립트 작성 후 실행



```
#camera.py

from picamera2 import Picamera2, Preview
import time

picam2 = Picamera2()
camera_config = picam2.create_preview_configuration()
picam2.configure(camera_config)
picam2.start_preview(Preview.QTGL)

picam2.start()
time.sleep(2)
picam2.capture_file("test.jpg")
```

```
pi@raspberrypi: ~ $ python3 camera.py
```

# Pi Camera 테스트

- python으로 Raspberry Pi Camera의 영상 녹화하기 🎥
  - record\_video.py 스크립트 작성 후 실행



Pi에서 실행

```
# record_video.py

from picamera2.encoders import H264Encoder
from picamera2 import Picamera2
import time

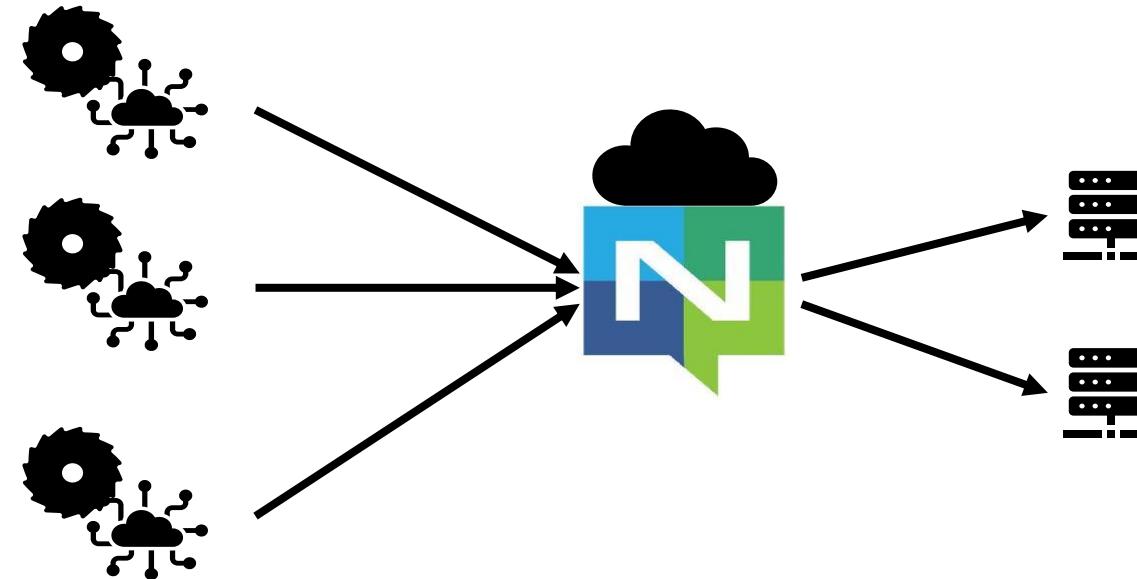
picam2 = Picamera2()
video_config = picam2.create_video_configuration()
picam2.configure(video_config)

encoder = H264Encoder(bitrate=10000000)
output = "my_video.h264"

picam2.start_recording(encoder, output)
time.sleep(5)
picam2.stop_recording()
```

# NATS?

- NATS : 메시지 지향 미들웨어(message oriented middleware)
  - 소프트웨어 애플리케이션과 서비스 간의 데이터를 교환할 수 있는 인프라를 제공
  - NATS는 메시지 형태로 데이터를 교환할 수 있으며, 이를 "메시지 지향 미들웨어" 또는 "분산 시스템"으로 정의
  - 다양한 시스템 전반에서 실시간 데이터를 저장하고 보낼 수 있음



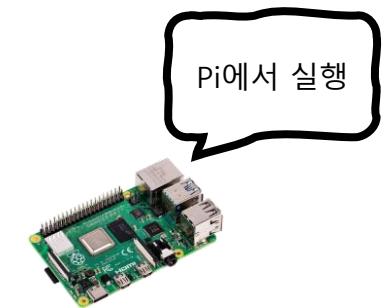
- 출처 : <https://docs.nats.io/>

# NATS vs Apache Kafka



- 시스템 여러 구성 요소 간의 통신을 가능하게 하는 Message System의 대표적인 open source
  - 안정적인 서비스 제공
  - 실시간 데이터 처리 및 안전한 데이터 전송 보장
  - 대량의 데이터를 효율적으로 처리
- NATS
  - Subject-based messaging 을 사용하여 Kafka 보다 간단(Message Queue 방식)
  - Microservice에 적합
- Kafka
  - Event Streaming platform
  - 실시간 대량의 데이터를 처리할 수 있는 big-data application에 적합

# Raspberry Pi에서 NATS 설정하기



- NATS 라이브러리 설치

```
pi@raspberrypi: ~ $ pip3 install nats-py
```

- NATS 테스트

```
pi@raspberrypi: ~ $ python3 genius_camp/pi/nats-test.py
```

- 아래와 같은 응답이 온다면 정상 작동

```
Received: Msg(_client=<nats client v2.4.0>, subject='foo', reply='', data=b'Hello from Python!',  
headers=None, _metadata=None, _ackd=False, _sid=1)
```

- NUC에서 작동 확인을 위해 아래의 파일을 실행

```
pi@raspberrypi: ~ $ python3 genius_camp/pi/nats-main.py
```

# NUC에서 NATS 설정하기

NUC에서 실행



- NUC에 라이브러리 설치

```
ubuntu@ubuntu-machine: ~ $ sudo apt install python3-pip
ubuntu@ubuntu-machine: ~ $ sudo rm /usr/lib/python3.11/EXTERNALLY-MANAGED
ubuntu@ubuntu-machine: ~ $ pip install nats-py
```

```
ubuntu@ubuntu-machine: ~ $ sudo pip install pillow --upgrade
ubuntu@ubuntu-machine: ~ $ sudo apt-get install python3-tk
ubuntu@ubuntu-machine: ~ $ sudo apt install libatlas-base-dev
```

```
ubuntu@ubuntu-machine: ~ $ sudo pip3 install opencv-python
ubuntu@ubuntu-machine: ~ $ sudo pip3 install -U numpy
```

# NATS Test

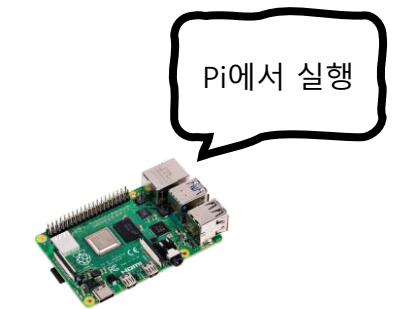


- NUC에서 NATS 테스트를 위한 파일 실행

```
ubuntu@ubuntu-machine: ~ $ python3 genius_camp/nuc/nats-test.py
```

- 반드시 Pi에서 실행되어 있는 상태에서 진행
- 확인 후 지금까지 실행한 .py 는 ctrl+c 를 이용하여 정지

# Raspberry Pi에서 영상 전송



- Raspberry Pi에서 실행

```
pi@raspberrypi: ~ $ python3 genius_camp/pi/result.py --subject {원하는 이름}
```

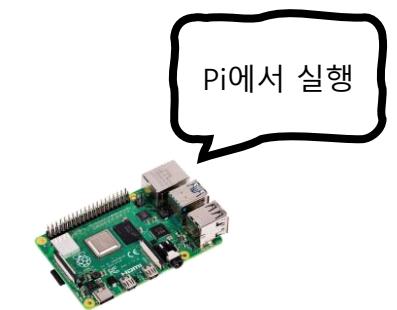
- NUC에서 실행

```
ubuntu@ubuntu-machine: ~ $ python3 genius_camp/nuc/result.py --subject {원하는 이름}
```

- Raspberry Pi에서 먼저 실행시키지 않으면 NUC에서 동작하지 않음



# Raspberry Pi 카메라 객체 인식



- Raspberry Pi에서 실행

```
pi@raspberrypi: ~ $ python3 genius_camp/pi/result.py --subject {원하는 이름}
```

- NUC에서 실행

```
ubuntu@ubuntu-machine: ~ $ cd genius_camp/nuc/object-detect
```

```
ubuntu@ubuntu-machine: ~ $ python3 video_object_detection.py --subject {원하는 이름}
```

