



[Introduction]

한국폴리텍대학교 성남캠퍼스

INDEX

1 ROS 개요

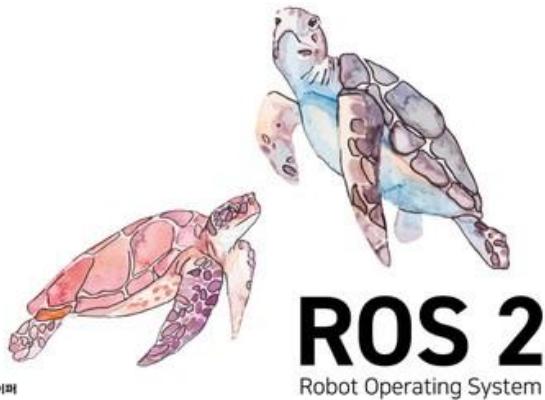
❖ 강의 참고자료

- 주 참고 교재는 다음과 같음
 - ROS2로 시작하는 로봇 프로그래밍
 - 출판사 : 루비페이퍼
 - 저 자 : 표윤석
- 관련 웹사이트
 - 오픈 카페로 온라인 세미나 및
참고자료를 바탕으로 책 발간

<https://cafe.naver.com/openrt/24070>

ROS 2로 시작하는
로봇 프로그래밍

표윤석, 임태준 지음



- ❖ ROS – Robot Operating System

<https://www.ros.org/>

ROS - Robot Operating System

The Robot Operating System (ROS) is a set of software libraries and tools that help you build robot applications. From drivers to state-of-the-art algorithms, and with powerful developer tools, ROS has what you need for your next robotics project. And it's all open source.



ROS 개요

❖ ROS – Platform?

하드웨어 모듈 + 운영체제 + 응용프로그램(서비스) + 사용자



Personal Computer



Personal Phone

❖ ROS – Platform?

- 컴퓨터를 위한 생태계 존재
- 각 분야 별 협업을 통한 시장 형성 및 동반 성장



Windows 10
Windows 8



❖ ROS – Platform?

- 플랫폼화로 인한 효과
 - ✓ 하드웨어 인터페이스 통합
 - ✓ 하드웨어 추상화, 규격화, 모듈화
 - ✓ 저가, 고성능화
 - ✓ 하드웨어, 운영체제, 응용프로그램 분야로 분리
 - ✓ 사용자 수요에 맞는 서비스에 집중
 - ✓ 사용자 증가로 시장 형성 및 생태계 선순환 구조 형성

- ❖ ROS – Middleware, Meta OS, Tool?



❖ ROS – Platformization

- 하드웨어 플랫폼과 소프트웨어 플랫폼 간의 인터페이스 확립
- 모듈형 하드웨어 플랫폼 확산
- 하드웨어에 대한 지식이 부족해도 응용프로그램 작성 가능
- 소프트웨어인력들이 로보틱스 분야로 진입 가능성 확대
- 유저에게 제공할 서비스에 집중
- 실 수요가 있는 서비스 제공으로 유저계층 형성 및 피드백 확보로 로봇관련 시장 생태계 형성

❖ ROS

- ROS의 진정한 목적 : 로보틱스 소프트웨어 개발을 전 세계 레벨에서 공동 작업이 가능하도록 생태계를 구축하는 것
- ROS = Robot Operating System

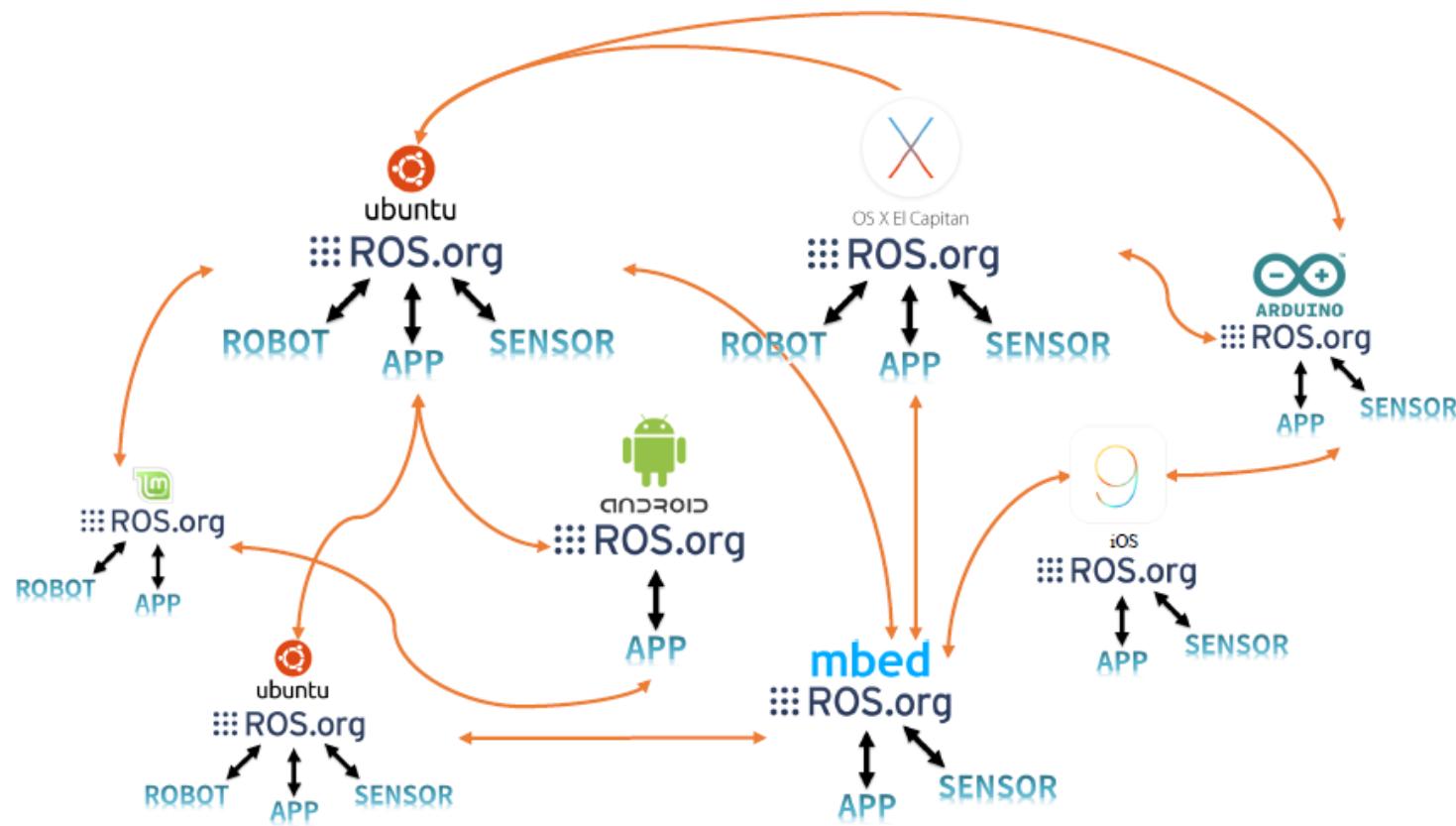
<https://wiki.ros.org/>



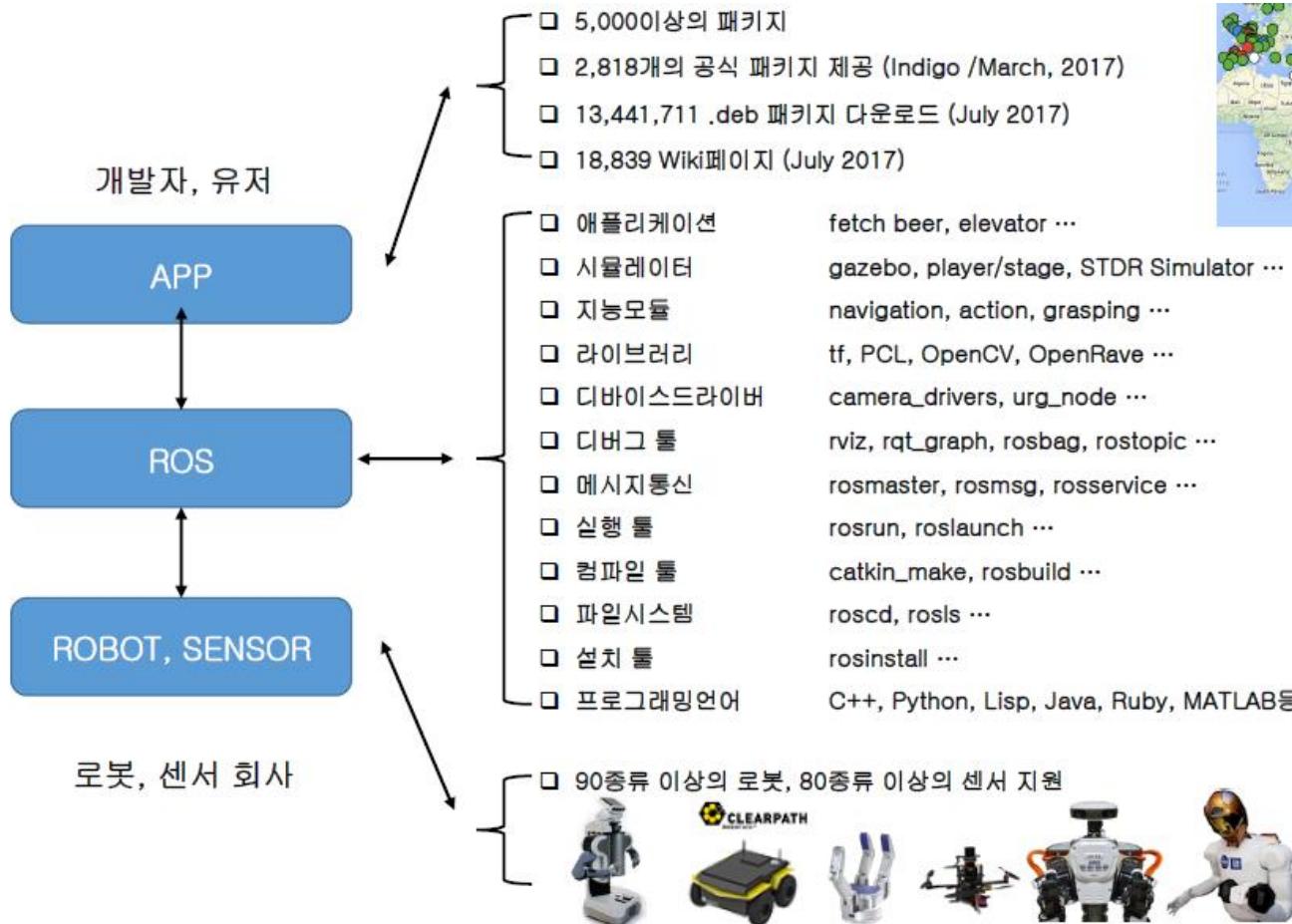
❖ ROS – Meta Operating System

- 메타운영체제(Meta-Operating System), 정확히 정의된 용어는 아니지만, 어플리케이션과 분산 컴퓨팅 자원 간의 가상화 레이어로 분산 컴퓨팅 자원을 활용하여, 스케줄링 및 로드, 감시, 에러처리 등을 실행하는 시스템
- 즉, 윈도우, 리눅스, 안드로이드와 같은 전통적인 운영체제가 아니며, 기존의 전통적인 운영체제를 지원하고 있으며, 기존 운영체제의 프로세스 관리시스템, 파일시스템, 유저 인터페이스, 프로그램 유ти(컴파일러, 스레드 모델 등) 등을 사용
- 추가적으로 다수의 이기종 하드웨어 간 데이터 송수신, 스케줄링, 에러처리 등 로봇 응용소프트웨어 개발을 위한 필수 기능들을 라이브러리형태로 제공
- 로봇 소프트웨어 프레임워크를 기반으로 다양한 목적의 응용프로그램을 개발, 관리, 제공하고 있으며 유저들이 개발한 패키지를 유통할 수 있는 생태계(ecosystem) 보유

❖ ROS – Tool

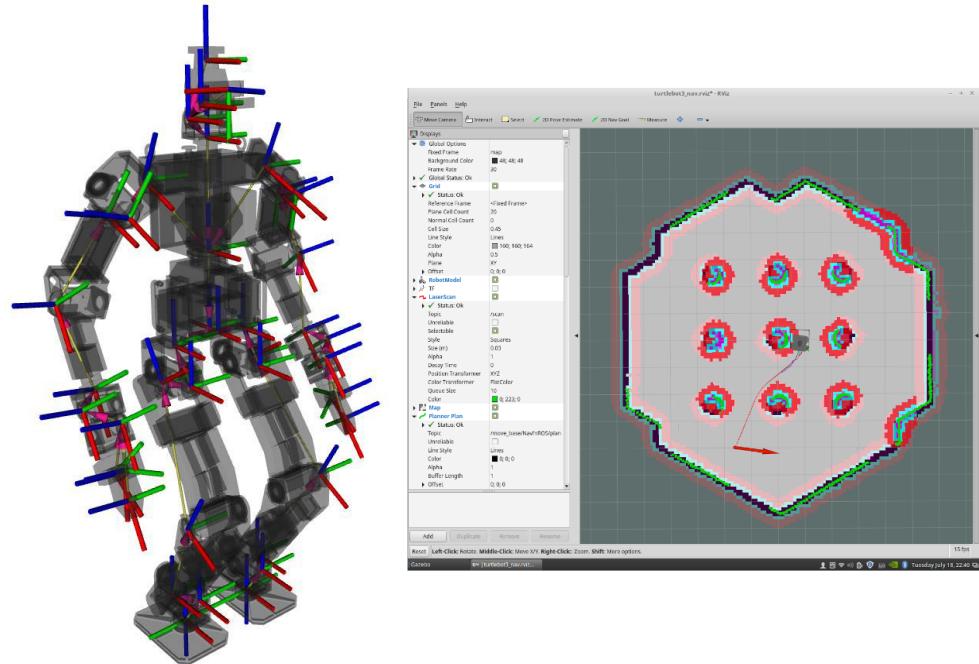


❖ ROS – Tool



❖ ROS – 특징

- 노드 간 데이터 통신 제공
 - ✓ 통상적 미들웨어로 지칭되는 메시지 전달 인터페이스 지원
 - ✓ 카메라, IMU, 레이저 등의 센서와 경로 및 지도 등의 내비게이션 데이터 등에 대한 표준 메시지 정의
- 로봇 기하학 라이브러리
- 로봇 기술 언어(XML)
- 진단 시스템
- 센싱/인식 라이브러리 제공
- 내비게이션(SLAM)
- 매니퓰레이션(MoveIt!)



❖ ROS – 통신

- 메시지 파싱
 - ✓ 로봇 개발 시 빈번히 사용되는 통신 시스템 제공
 - ✓ 캡슐화 및 코드 재사용을 촉진하는 노드들 간의 메시지 전달 인터페이스
- 메시지의 기록 및 재생
 - ✓ 노드 간 송/수신 되는 데이터 인 메시지를 저장하고 필요 시에 재사용 가능
 - ✓ 저장된 메시지를 기반으로 반복적인 실험 가능, 알고리즘 개발에 용이함
- 메시지 사용으로 인한 다양한 프로그래밍 언어 사용 가능
 - ✓ 노드 간의 데이터교환이 메시지를 사용하기 때문에 각 노드는 서로 다른 언어로 작성 가능
 - ✓ 클라이언트라이브러리: roscpp, rospy, roslib, rosjava, roslua, rosccs, roseus, PhaROS, rosR
- 분산 매개 변수 시스템
 - ✓ 시스템에서 사용되는 변수를 글로벌 키 값으로 작성하여 공유 및 수정하여 실시간으로 반영

❖ ROS – 개발도구

- Command-Line Tools
 - ✓ GUI 없이 ROS에서 제공되는 명령어로만 로봇 제어 및 거의 모든 ROS 기능 수행
- RViz
 - ✓ 3D 시각화 도구
 - ✓ 레이저, 카메라 등 센서데이터 시각화
 - ✓ 로봇 외형과 계획된 동작을 표현
- RQT
 - 그래픽 인터페이스 개발을 위한 Qt기반 프레임워크 제공
 - 노드와 그 사이의 연결정보 표시(rqt_graph)
 - 인코더, 전압, 또는 시간에 따라 변하는 값 Plotting(rqt_plot)
 - 데이터를 메시지 형태로 기록하고 재생(rqt_bag)
- Gazebo
 - 물리엔진 탑재, 로봇, 센서, 환경모델 등을 지원, 3차원 시뮬레이터
 - ROS와 높은 호환성

2 ROS 2

❖ ROS 2 vs ROS 1

- ROS 1은 2007년 Willow Garage사가 개인 서비스 로봇인 PR2개발에 필요한 미들웨어 형태의 로봇 개발 프레임워크를 다양한 개발 툴과 함께 오픈 소스로 공개 한 것으로 시작
- 지금은 대학, 연구 기관, 산업계, 로봇 자작 취미활동까지 폭넓게 이용
- 따라서 개발 환경은 PR2의 초기 컨셉을 그대로 이어 다음과 같은 제한 사항이 있음
 - ✓ 단일 로봇
 - ✓ 워크스테이션급 컴퓨터
 - ✓ Linux 환경
 - ✓ 실시간 제어 지원하지 않음
 - ✓ 안정된 네트워크 환경이 요구됨
 - ✓ 주로 대학이나 연구소와 같은 아카데믹 연구 용도

❖ ROS 2 vs ROS 1

- 상업용 로봇 개발 환경과는 큰 차이가 있고, 요구 사항이 늘어남
 - ✓ NASA가 국제 우주 정거장에서 사용한 Robonaut에는 ROS 1가 채용되고 있지만, 실시간 제어를 지원하지 않아 ROS 1을 수정하여 사용
- 새로운 로봇 개발 환경 및 요구되는 기능은 다음과 같음
 - ✓ 복수의 로봇에 적용
 - ✓ 임베디드 시스템에서의 ROS 사용
 - ✓ 실시간 제어
 - ✓ 불안정한 네트워크 환경에서도 동작 할 수 있는 유연함
 - ✓ 멀티 플랫폼 (Linux, Windows, macOS)
 - ✓ 최신 기술 지원 (Zeroconf, Protocol Buffers, ZeroMQ, WebSockets, DDS 등)
 - ✓ 상업용 제품 지원

❖ ROS 2 vs ROS 1

- ROS 1에서 이러한 새롭게 요구되는 기능을 제공하려면 대규모 API의 변경이 필요
 - ✓ 기존의 ROS 1과의 호환성을 유지하면서 수 많은 새로운 기능을 추가하는 것은 쉽지 않음
 - ✓ 기존의 ROS 1을 문제 없이 이용하고 있는 사용자는 API의 큰 변화는 호환성 문제 야기
- ROS에게 요구되던 기능을 도입한 버전을 ROS 2라고, ROS 1에서 분리하여 개발
- 기존의 ROS1 사용자는 필요하다면 그대로 ROS 1을 이용할 수 있음
- ROS 1과 ROS 2 사이에서 서로 메시지 통신이 가능한 브리지 프로그램 (ros1_bridge) 제공되므로 두 버전 모두를 함께 사용하는 것도 가능

❖ Why ROS 2?



Why ROS 2?





ROS 2 (Robot Operating System 2) is an open source software development kit for robotics applications. The purpose of ROS 2 is to offer a standard software platform to developers across industries that will carry them from research and prototyping through to deployment and production. ROS 2 builds on the success of ROS 1, which is used today in myriad robotics applications around the world.

- » **Shorten time to market**
ROS 2 provides the robotics tools, libraries, and capabilities that you need to develop your applications, allowing you to spend your time on the work that is important for your business. Because it is open source, you have the flexibility to decide where and how to use ROS 2, as well as the freedom to customize it for your needs.
- » **Designed for production**
Drawing on a decade of experience in establishing ROS 1 as the de facto global standard for robotics R&D, ROS 2 was built from the ground up to be industry-grade and used in production, including high reliability and safety critical systems. Design choices, development practices, and project governance for ROS 2 are based on requirements from industry stakeholders.
- » **Multi-platform**
ROS 2 is supported and tested on Linux, Windows, and macOS, allowing seamless development and deployment of on-robot autonomy, back-end management, and user interfaces. The tiered support model allows for ports to new platforms, such as real-time and embedded OSs, to be introduced and promoted as they gain interest and investment.
- » **Multi-domain**
Like ROS 1 before it, ROS 2 is ready for use across a wide array of robotics applications, from indoor to outdoor, home to automotive, underwater to space, and consumer to industrial.

ROS www.openrobotics.org www.ros2.org



Why ROS 2?











» **No vendor lock-in**
ROS 2 is built on an abstraction layer that insulates the robotics libraries and applications from the communication technologies. Below the abstraction are multiple implementations of the communications code, including both open source and proprietary solutions. Above the abstraction, core libraries and user applications are portable.

» **Built on open standards**
The default communications method in ROS 2 uses industry standards like IDL, DDS, and DDS+RTPS, which are already widely deployed in a variety of industrial applications, from factories to aerospace.

» **Permissive open source license**
ROS 2 code is licensed under Apache 2.0 License, with ported ROS 1 code under the 3-clause (or "new") BSD License. Both licenses allow permissive use of the software, without implications on the user's intellectual property.

» **Global community**
Over 10+ years the ROS project has produced a vast ecosystem of software for robotics by nurturing a global community of hundreds of thousands of developers and users who contribute to and improve that software. ROS 2 is developed by and for that community, who will be its stewards into the future.

» **Industry support**
As demonstrated by the membership of the ROS 2 Technical Steering Committee, industry support for ROS 2 is strong. Companies large and small from around the world are committing their resources to making open source contributions to ROS 2, in addition to developing products on top.

» **Interoperability with ROS 1**
ROS 2 includes a bridge to ROS 1 that handles bidirectional communication between the two systems. If you have an existing ROS 1 application, you can start experimenting with ROS 2 via the bridge and port your application incrementally according to your requirements and available resources.

ROS www.openrobotics.org www.ros2.org

- Shorten time to market
- Designed for production
- Multi-platform
- Multi-domain
- No vendor lock-in
- Built on open standards
- Permissive open-source license
- Global community
- Industry support
- Interoperability with ROS 1

❖ ROS 2 – 10가지 중요 이슈

(1) 시장 출시 시간 단축

- ROS 2는 로봇 응용 프로그램을 개발하는 데 필요한 도구, 라이브러리 및 기능을 제공하므로 본연의 중요한 로봇 개발 작업에 더 많은 시간을 할애할 수 있다는 것이 가장 큰 장점
- 특정 로봇 개발을 위해 처음부터 프레임워크를 만들고 통신 방법을 선정하고 디버깅 툴과 시각화 툴을 다시 만드는 비효율적인 방법에서 벗어날 수 있게 함
- ROS 2는 상용 소프트웨어가 아닌 ROS 커뮤니티에서 개발해오고 있는 오픈소스이기 때문에 ROS 2를 사용할 부분과 사용 방법을 유연하게 결정할 수 있을 뿐만 아니라 필요에 따라 자유롭게 수정할 수 있음

❖ ROS 2 – 10가지 중요 이슈

(2) 생산을 위한 설계

- ROS는 로보틱스 R&D의 사실 상의 글로벌 표준으로서 ROS 1의 10년의 경험을 바탕으로, ROS 2는 처음부터 산업용 수준으로 개발되고 높은 신뢰성과 안전을 중시
- ROS 1의 아카데믹 성격과는 달리 ROS 2는 프로토 타이핑 개발부터 실제 생산에 이르기 위해 ROS 2의 설계, 개발 및 프로젝트 관리는 업계 이해 관계자들로부터 얻은 실질적인 요구 사항을 기반으로 하고 있음

(3) 멀티 플랫폼

- ROS 2는 Linux, Windows, macOS에서 개발, 지원, 테스트 진행하고 있기에 자율성, 백엔드 관리 및 사용자 인터페이스의 원활한 개발 및 배포가 가능
- 계층형 지원 모델을 사용하고 있기에 실시간 운영체제(Real-time OS) 및 임베디드 OS(Embedded OS)와 같은 새로운 플랫폼으로의 포팅에 대한 관심과 투자를 통해 도입 및 홍보 가능

❖ ROS 2 – 10가지 중요 이슈

(4) 다중 도메인

- 이전의 ROS 1과 마찬가지로 ROS 2는 실내에서 실외, 가정에서 자동차, 수중에서 우주, 소비자에서 산업에 이르기까지 다양한 로봇 응용 분야에서 사용할 수 있음

(5) 벤더 선택 가능

- ROS 2는 로봇공학 라이브러리와 응용 프로그램을 통신 기능으로부터 분리하여 추상화
- 추상화된 부분에는 오픈소스 솔루션 방식과 독점 솔루션을 포함한 여러 가지 방법론을 제공하며 그 이외의 핵심 라이브러리 및 사용자 애플리케이션은 사용자가 원하는 형태로 개발, 수정하여 추가 가능

(6) 공개 표준 기반

- ROS 2의 기본 통신 방법은 IDL, DDS 및 DDS-I RTPS과 같이 제조 산업에서 항공 산업까지도 사용되고 있는 산업 표준을 사용

❖ ROS 2 – 10가지 중요 이슈

(7) 자유 재량 허용 범위가 넓은 오픈소스 라이센스 채택

- ROS 2 코드는 Apache 2.0 라이센스를 기본 라이센스로 사용하여 지적 재산권에 영향을 주지 않으면서 자유 재량으로 넓은 범위로 사용 가능

(8) 글로벌 커뮤니티

- ROS 커뮤니티는 10년 이상 ROS 프로젝트는 소프트웨어에 기여하고 개선하는 수십만 명의 개발자와 사용자로 구성된 글로벌 커뮤니티를 육성
- 로봇 공학을 위한 방대한 소프트웨어 에코 시스템을 구축하였고 ROS 2는 커뮤니티를 위해, 커뮤니티에 의해 개발

❖ ROS 2 – 10가지 중요 이슈

(9) 산업 지원

- ROS 2 기술 운영위원회(ROS 2 Technical Steering Committee)의 멤버쉽에서 알 수 있듯이 ROS 2에 대한 업계 지원은 강력함
- 전 세계의 크고 작은 회사는 제품을 개발할 뿐만 아니라 ROS 2에 오픈소스 기여를 하기 위해 자원을 투입하고 있음

(10) ROS 1과의 상호운용성 확보

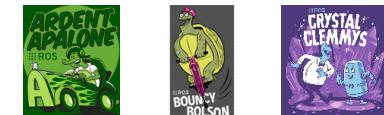
- ROS 2에는 두 시스템 간의 양방향 통신을 처리하는 ROS 1에 대한 브리지가 포함
- ROS 1 애플리케이션이 있는 경우 브리지를 통해 ROS 2 테스트를 시작하고 요구 사항 및 사용 가능한 자원에 따라 점차적으로 애플리케이션을 포팅할 수 있음

❖ ROS 2 history

- 2024.05.23 - ROS 2 Jazzy Jalisco (LTS, 5 years support)
- 2023.05.23 - ROS 2 Iron Irwini
- 2022.05.23 - ROS 2 Humble Hawksbill (LTS, 5 years support)
- 2021.05.23 - ROS 2 Galactic Geochelone
- 2020.06.05 - ROS 2 Foxy Fitzroy release (LTS, 3 years support)
- 2019.11.22 - ROS 2 Eloquent Elusor release
- 2019.05.31 - ROS 2 Dashing Diademata release (First LTS, 2 years support)
- 2018.12.14 - ROS 2 Crystal Clemmys release
- 2018.07.02 - ROS 2 Bouncy Bolson release
- 2017.12.08 - ROS 2 Ardent Apalone release (1st version)

- 2017.09.13 - ROS2 Beta3 release (code name R2B3)
- 2017.07.05 - ROS2 Beta2 release (code name R2B2)
- 2016.12.19 - ROS2 Beta1 release (code name Asphalt)

- 2016.10.04 - ROS2 Alpha8 release (code name Hook.and.Loop)
- 2016.07.14 - ROS2 Alpha7 release (code name Glue Gun)
- 2016.06.02 - ROS2 Alpha6 release (code name Fastener)
- 2016.04.06 - ROS2 Alpha5 release (code name Epoxy)
- 2016.02.17 - ROS2 Alpha4 release (code name Duct tape)
- 2015.12.18 - ROS2 Alpha3 release (code name Cement)
- 2015.11.03 - ROS2 Alpha2 release (code name Baling wire)
- 2015.08.31 - ROS2 Alpha1 release (code name Anchor)



- 2020년 마지막 릴리즈 후 EOL을 9개월 앞둔 ROS 1?
- 매년 새로운 버전을 릴리즈하고 있는 새로운 ROS 2?

The logo for ROS 2 features a stylized icon composed of four vertical dots of increasing size from left to right, followed by the text "ROS 2" in a large, bold, sans-serif font. A small superscript "TM" is positioned at the top right of the "2".

3 실습환경

❖ 실습환경

1. 운영체계

- Ubuntu 22.04.4 LTS(Jammy Jellyfish)
- <https://learn.microsoft.com/ko-kr/windows/wsl/install>

2. ROS

- ROS 2 Humble
- <https://docs.ros.org/en/humble/index.html>

3. IDE

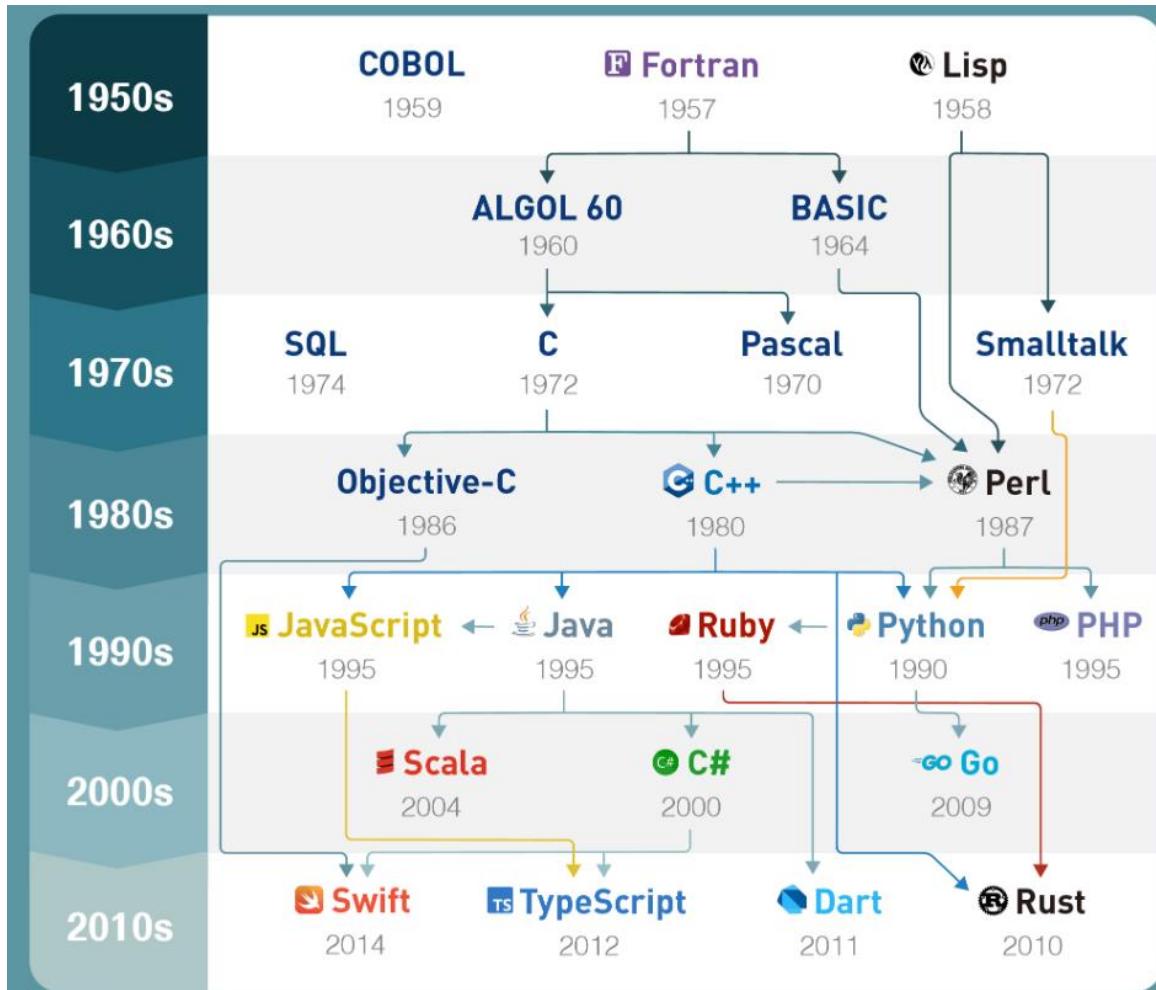
- Visual Studio Code

4. Programming Language

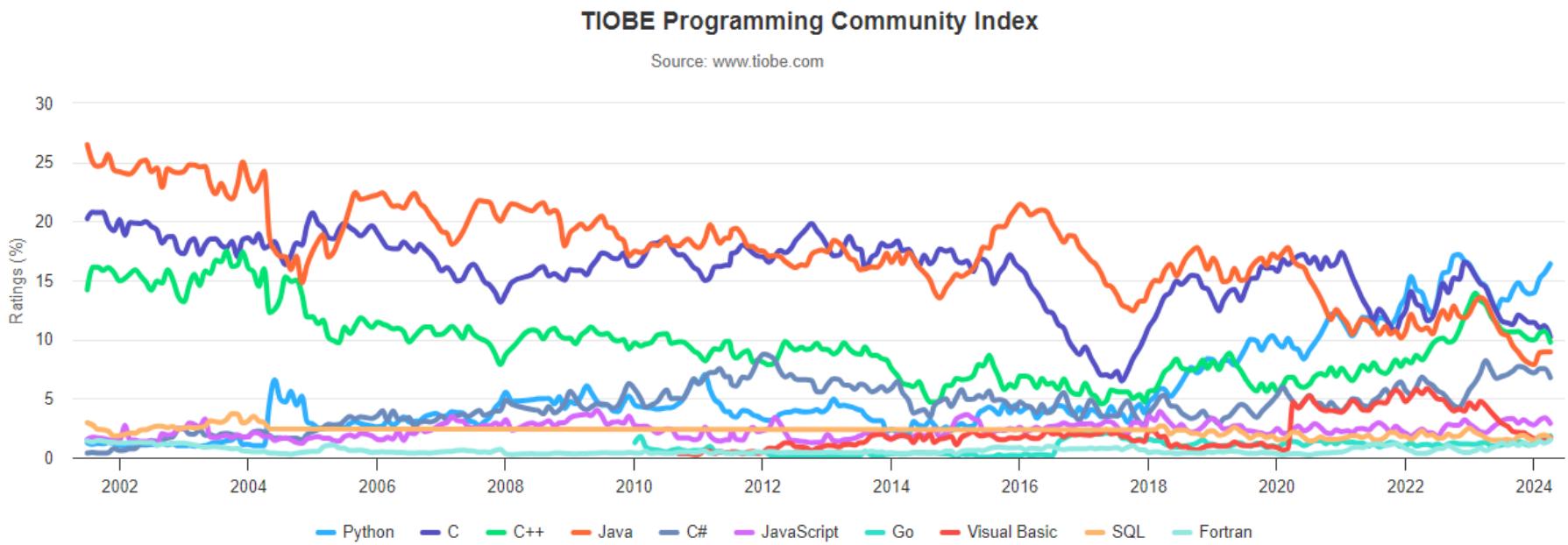
- Python, C++

4 Python basic

History of programming language



History of programming language

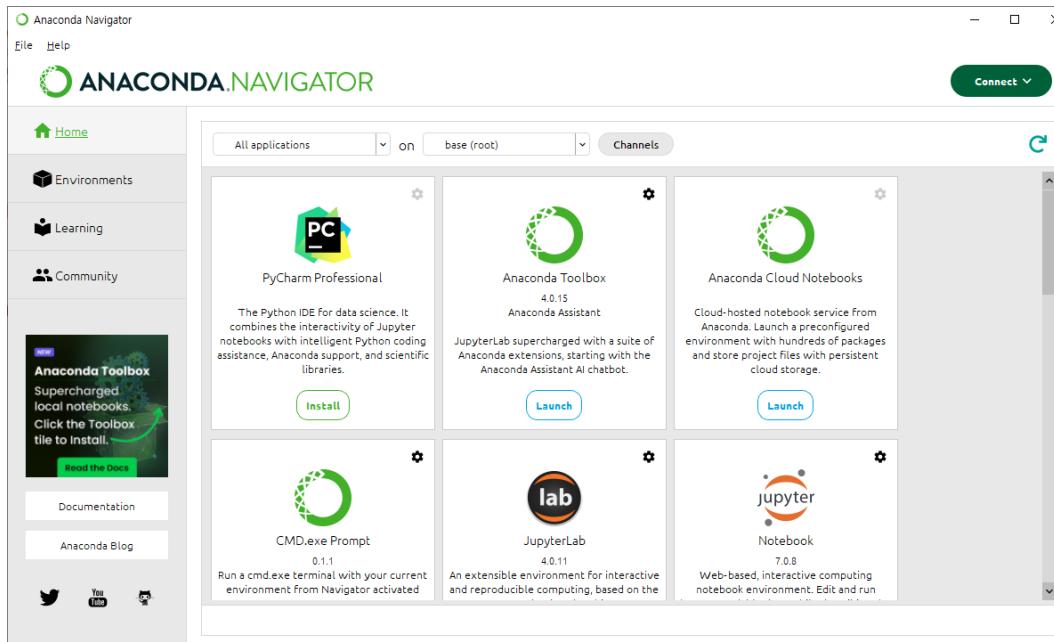


SRC : <https://www.tiobe.com/tiobe-index/>

Python with ANACONDA



- ❖ The world's most popular open-source Python distribution platform
- ❖ Anaconda Distribution contains conda, which is a package and environment manager, which helps users to manage a collection of over 8,000 open-source data science and machine learning packages.



■ Python with ANACONDA

homepage : <https://www.anaconda.com/>



Products Solutions Resources Partners Company

Free Download

Sign Up

Sign In



Windows

Python 3.12

64-Bit Graphical Installer (912.3M)



Provide email to download Distribution

Don't miss out! Get access to: Cloud Notebooks, Anaconda Assistant, easy application deployment, learning resources, and updates from Anaconda.

Email Address:

I agree to receive communication from Anaconda regarding relevant content, products, and services. I understand that I can revoke this consent [here](#) at any time.

By continuing, I agree to Anaconda's [Privacy Policy](#) and [Terms of Service](#).

Submit >

Skip registration



Install for : Just Me (recommended)

Advanced Options : Add Anaconda to my_PATH environment variable check

IDE Setting with Visual Studio Code

homepage : <https://code.visualstudio.com/>

The screenshot shows the Visual Studio Code homepage. At the top right is a red-bordered "Download" button. Below it is a large Windows logo with a blue arrow pointing towards it. On the left is the extension marketplace sidebar with icons for file, search, keymap, notebook, slide show, Pylance, Python, and Python Debugger. The main area shows a list of installed extensions under the "INSTALLED" heading, including Jupyter, Jupyter Cell Tags, Jupyter Keymap, Jupyter Notebook, Jupyter Slide Show, Pylance, Python, and Python Debugger. Each entry includes a Microsoft logo and a small circular badge with the number 8.

Windows
Windows 10, 11

User Installer x64 Arm64
System Installer x64 Arm64
.zip x64 Arm64
CLI x64 Arm64

Select Interpreter

Selected Interpreter: ~\anaconda3\python.exe

- + Create Virtual Environment...
- Enter interpreter path...
- Python 3.11.4 64-bit ~\AppData\Local\Programs\Python\Python311\python.exe Recommended
- Python 3.12.4 ('base') ~\anaconda3\python.exe Conda Global
- Python 3.11.4 64-bit ~\AppData\Local\Programs\Python\Python311\python.exe Global
- Python 3.10.11 64-bit (system) C:\msys64\mingw64\bin\python.exe

■ Python with Jupyter Notebooks



A screenshot of a Jupyter Notebook interface within the Visual Studio Code editor. The title bar shows "Test > test.ipynb". The menu bar includes "Code", "Markdown", "Run All", "Restart", "Clear All Outputs", "Variables", and "base (Python 3.12.4)". The code cell contains the following Python code:

```
a=123
b="hello"
print("a={}, b={}".format(a,b))
print(f"a={a}, b={b}")
```

The output cell shows the results of the execution:

```
[7] ...
... a=123, b=hello
a=123, b=hello
```

The "Python" tab is selected at the bottom right.

- A kernel is a “computational engine” that executes the code contained in a notebook document.
- A cell is a container for text to be displayed in the notebook or code to be executed by the notebook’s kernel.
- A code cell contains code to be executed in the kernel. When the code is run, the notebook displays the output below the code cell that generated it.
- A Markdown cell contains text formatted using Markdown and displays its output in-place when the Markdown cell is run.
- You can use Jupyter Notebooks files in VS Code with “~~~.ipynb” file extension.

■ Python Basics – Variables

❖ Print()

```
print("1 hello")
print("2 hello", "hi")
print("3 hello"+ "hi")
print("4 hello", end="")
print('5 hello')
print('6 hello', "hi")
print("7 'hello' ")
print('8 "hello" ')
```

❖ Input()

```
input()
input("Input value = ")
a = input("Input 1st value = ")
b = input("Input 2nd value = ")
print(a+b)
```

```
a=123
b="hello"
print("a={}, b={}".format(a,b))
print(f'a={a}, b={b}')
```

Python Basics – Variables

❖ Variable Types

```
a=10  
b=20  
c=a+b  
print(c)
```

30

```
e=3.14  
f=10  
print(e+f)
```

13.14

```
a=10  
b=20  
c=float(a)+float(b)  
print(c)
```

30.0

```
d='10'  
print(c+d)
```

TypeError

```
d='10'  
print(c+int(d))
```

40

```
d='10'  
print(str(c)+d)
```

3010

```
a_bool = True  
b_bool = False  
a_int = 1  
b_int = 0  
print(a_bool)  
print(b_bool)  
print(type(a_bool))  
print(type(b_bool))  
print(type(a_int))  
print(type(b_int))
```

■ Python Basics – Data Types

❖ List (use '[]')

```
a_list = [1,2,3,4,5]  
print(a_list)  
print(a_list[0])  
print(a_list[1])
```

```
[1,2,3,4,5]  
1  
2
```

```
print(a_list[:2])  
print(a_list[2:])
```

```
[1,2]  
[3,4,5]
```

```
b_list = []  
b_list.append(1)  
b_list.append(2)  
b_list.append(3)  
print(b_list)
```

```
[1,2,3]
```

```
c_list = [ 1, 3.14, 'hello', [1,2,3] ]  
print(c_list)  
print(c_list[1:3])
```

```
[1, 3.14, 'hello', [1,2,3] ]  
[3.14, 'hello' ]
```

```
d_list = [ 1, 2, 3, 4, 5 ]  
print(d_list)  
d_list[0] = 5  
print(d_list)
```

```
[1, 2, 3, 4, 5 ]  
[5, 2, 3, 4, 5 ]
```

■ Python Basics – Data Types

❖ Tuple

- Tuple is similar to list, but values are immutable.(use '()')

```
a_tuple = (1,2,3,4,5)
```

```
print(a_tuple)
```

```
print(a_tuple[0])
```

```
a_tuple[0] = 5
```

```
(1,2,3,4,5)
```

```
1
```

```
TypeError
```

❖ Set

- Set removes duplicated data and doesn't sort it in order.(use '{ }')

```
a_set = set( [1,2,3,4] )
```

```
print(a_set)
```

```
b_set = set( [1,1,2,2,3,4] )
```

```
print(b_set)
```

```
{1,2,3,4}
```

```
c_set = set("python40s")
```

```
print(c_set)
```

```
{'o', 'h', 't', 'n', 'y', '4', 's', '0', 'p'}
```

■ Python Basics – Data Types

❖ Dictionary

- Dictionary consists of keys and values.
- Dictionary is expressed in the following form: { key1:value1, key2:value2, key3:value3 }

```
a_dic = {'a':1, 'b':2, 'c':3}
```

```
print(a_dic)
```

```
print(a_dic['a'])
```

```
print(a_dic['c'])
```

```
{'a':1, 'b':2, 'c':3}
```

```
1
```

```
3
```

```
b_dic = {1:'a', 'b':[1,2,3], 'c':3}
```

```
print(b_dic[1])
```

```
print(b_dic['b'])
```

```
print(b_dic['c'])
```

```
a
```

```
[1, 2, 3 ]
```

```
3
```

```
b_dic['d'] = 4
```

```
print(b_dic)
```

```
{ 1:'a', 'b':[1,2,3], 'c':3, 'd':4 }
```

■ Python Basics – Operator

❖ Arithmetic

```
print("Add : ", 10+20)
print("Sub : ", 10-20)
print("Mul : ", 10*20)
print("Div : ", 10/20)
```

Add : 30
Sub : -10
Mul : 200
Div : 0.5

```
print( 10**2 )
print( 10**3 )
print( 10**4 )
```

100
1000
10000

```
print( 40//6 )
print( 40%6 )
```

6
4

❖ Logic

```
print(0 or 0)
print(0 or 1)
print(1 or 0)
print(1 or 1)
print(False or False)
print(False or True)
```

0
1
1
1
False
True

```
print(0 and 0)
print(0 and 1)
print(1 and 0)
print(1 and 1)
print(False and True)
print(True and True)
```

0
0
0
1
False
True

```
print( not 0 )
print( not 1 )
```

True
False

```
print( not False )
print( not True )
```

True
False

▀ Python Basics – Operator

❖ Comparison

```
print( 10 == 10 )  
print( 10 >= 10 )  
print( 10 <= 10 )  
print( 10 < 5 )  
print( 10 > 5 )  
print( 10 != 10 )
```

```
True  
True  
True  
False  
True  
False
```

```
a_list = [ 'a', 2, 'hello', 3 ]  
print( 'a' in a_list )  
print( 1 in a_list )  
print( 'hello' in a_list )  
print( '3' in a_list )
```

```
True  
False  
True  
False
```

```
a_str = "hello python"  
print( "python" in a_str )  
print( "py" in a_str )  
print( "40" in a_str )
```

```
True  
True  
False
```

Python Basics – Condition

❖ If

```
A = 1  
B = 2  
  
if A == B :  
    print( "Equal" )  
  
else :  
  
    print( "Unequal" )
```

Unequal

```
a_str = "hello python"  
  
if a_str == "hello python" :  
    print("string is equal.")  
  
if 'hi' not in a_str :  
  
    print("hi is not included.")
```

string is equal.
hi is not included.

```
A = 1  
B = 2  
  
if A > B :  
    print( "A is bigger" )  
  
elif A<B :  
  
    print( "B is bigger" )  
  
else :  
  
    print( "Equal" )
```

B is bigger

```
a_list = ["hello", 1, 2, 'python']  
  
if "hello" in a_list :  
    print("string is included.")  
  
if 2 not in a_list :  
  
    print("it's not working.")
```

string is included.

■ Python Basics – Loop

❖ For

```
for i in range(7) :  
    print(i)
```

```
0  
1  
2  
3  
4  
5  
6
```

```
a_list = [1,2,3,4,5, 'hi', 'good']  
  
for i in a_list :  
  
    print(i)
```

```
1  
2  
3  
4  
5  
hi  
good
```

```
for i in range(5, 10) :  
    print(i)
```

```
5  
6  
7  
8  
9
```

```
word_list = ['kim', 'dae', 'sung']  
num_list = [ 100, 200, 300 ]  
  
for i, k in enumerate(word_list) :  
    print(k, end=' ')  
  
    print(num_list[i])  
  
for i, k in enumerate(word_list) :  
    print(word_list[i], end=' ')  
  
    print(num_list[i])
```

```
kim 100  
dae 200  
sung 300  
kim 100  
dae 200  
sung 300
```

```
for i in range(10, 5, -1) :  
    print(i)
```

```
10  
9  
8  
7  
6
```

■ Python Basics – Loop

❖ For

```
word_list = ['kim', 'dae', 'sung']
num_list = [ 100, 200, 300 ]
for i in range(len(word_list)):
    print(word_list[i], end=' ')
    print(num_list[i])
```

```
kim 100
dae 200
sung 300
```

```
test_list = [ i for i in range(5) ]
print(test_list)
```

```
[0, 1, 2, 3, 4]
[0, 1, 2, 3, 4]
```

```
test2_list = []
for i in range(len(test_list)):
    test2_list.append(i)
print(test2_list)
```

```
test_list = [ i*5 for i in range(5) ]
print(test_list)
```

```
[0, 5, 10, 15, 20]
[0, 0, 0, 0, 0]
```

```
test2_list = [ 0 for i in range(5)]
print(test2_list)
```

 Python Basics – Loop

❖ While

```
a = 0  
while a<5 :  
    print(a)  
    a = a+1
```

```
0  
1  
2  
3  
4
```

```
a = 0  
while True :  
    print(a)  
    a = a+1  
  
    if a>=5 :  
        break
```

```
0  
1  
2  
3  
4
```

Python Basics – Error and Exception

❖ Try / Except

```
try :  
    dlksjdslkjsfd  
except:  
    print('Error!!')
```

Error!!

```
try :  
    dlksjdslkjsfd  
except:  
    pass  
print('Skip Error')
```

Skip Error

```
try :  
    dlksjdslkjsfd  
except Exception as e :  
    print('Error : ', e)
```

Error : name 'dlksjdslkjsfd' is not defined

Python Basics – Function

❖ Using functions

```
def func():  
    print("This is func print.")  
  
func()
```

This is func print.

```
def func_add(a,b) :  
    return a+b  
  
c = func_add(1,2)  
print(c)
```

3

```
def func_add_mul(a,b) :
```

add = a+b

mul = a*b

return add, mul

```
c, d = func_add_mul(3,4)
```

```
print(c, d)
```

7 12

```
def func_add_mul(a,b) :
```

add = a+b

mul = a*b

return add, mul

```
_, d = func_add_mul(3,4)
```

```
print(d)
```

12

Python Basics – Class

❖ Declaring and using Class

```
class Greet():

    def hello(self):
        print("hello")

    def hi(self):
        print("hi")
```

```
human1 = Greet()
human2 = Greet()
human1.hello()
human1.hi()
human2.hello()
human2.hi()
```

```
hello
hi
hello
hi
```

```
class student():

    def __init__(self, name, age, like):
        self.name = name
        self.age = age
        self.like = like

    def stu_info(self):
        print(f'{self.name} / {self.age} / {self.like}')

A = student("kim", 17, 'Boxing')
B = student("dae", 27, 'Game')
A.stu_info()
B.stu_info()
```

```
kim / 17 / Boxing
dae / 27 / Game
```

▀ Python Basics – Class

❖ Declaring and using Class

```
class mom():
    def characteristic(self):
        print("Tall")
        print("Smart")
class daughter(mom):
    def characteristic(self):
        super().characteristic()
        print("Strong")
```

```
M = mom()
D = daughter()
print("[ Mother ]")
M.characteristic()
print("[ Daughter ]")
D.characteristic()
```

[Mother]
Tall
Smart
[Daughter]
Tall
Smart
Strong

```
class mom():
    def __init__(self):
        print("Tall")
        print("Smart")
class daughter(mom):
    def __init__(self):
        super().__init__()
        print("Strong")
```

```
print("[ Mother ]")
M = mom()
print("[ Daughter ]")
D = daughter()
```

[Mother]
Tall
Smart
[Daughter]
Tall
Smart
Strong

Python Basics – Comments

❖ Using Comments

```
# Comment1  
print("hello") # Comment2
```

hello

```
"""  
Comment line 1  
Comment line 2  
Comment line 3  
"""
```

```
a_str = """  
line 1  
line 2  
line 3  
"""  
print(a_str)
```

line 1
line 2
line 3

- Drag the code areas and press [Ctrl + /] : Codes ↔ Comments

▀ Python Basics – import

- ❖ Using import to call library or module

```
import random  
print(random.randint(1,100))
```

63

```
import random as rd  
print(rd.randint(1,100))
```

59

```
from random import randint  
print(randint(1,100))
```

30

```
from random import *  
print(randint(1,100))
```

2

■ Python Projects – 1) Number Guessing Game

- ❖ Make Python code that can work as following:

```
Input number 1~99 : eer
Error : invalid literal for int() with base 10: 'eer'
Input number 1~99 : 50
Up
Input number 1~99 : 80
Down
Input number 1~99 : 60
Down
Input number 1~99 : 55
Down
Input number 1~99 : 53
Up
Input number 1~99 : 54
Congratulations, you got it right in 6 tries.
```

■ Python Projects – 2) Qr Codes

❖ Using qrcode Library

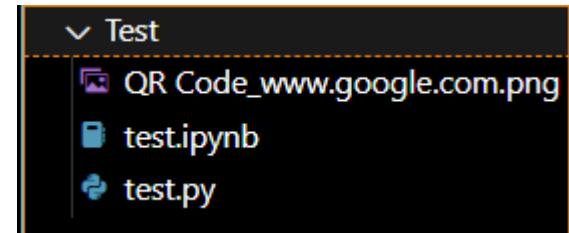
```
pip install qrcode
```

Type these commands on terminal window

```
import qrcode

qr_data = 'www.google.com'
qr_img = qrcode.make(qr_data)

save_path = 'QR Code_'+qr_data+'.png'
qr_img.save(save_path)
```



Python Projects – 3) Qr Codes

◆ Make Qr Codes from *.txt file

```
import qrcode

file_path = r'QrCodes.txt'

with open(file_path, 'rt', encoding='UTF8') as f :
    read_lines = f.readlines()
    cnt=0
    for line in read_lines :
        cnt = cnt+1
        line = line.strip()
        print(line)
        qr_data = line
        qr_img = qrcode.make(qr_data)

        save_path = 'QR_Code'+ str(cnt) + '.png'
        qr_img.save(save_path)
```

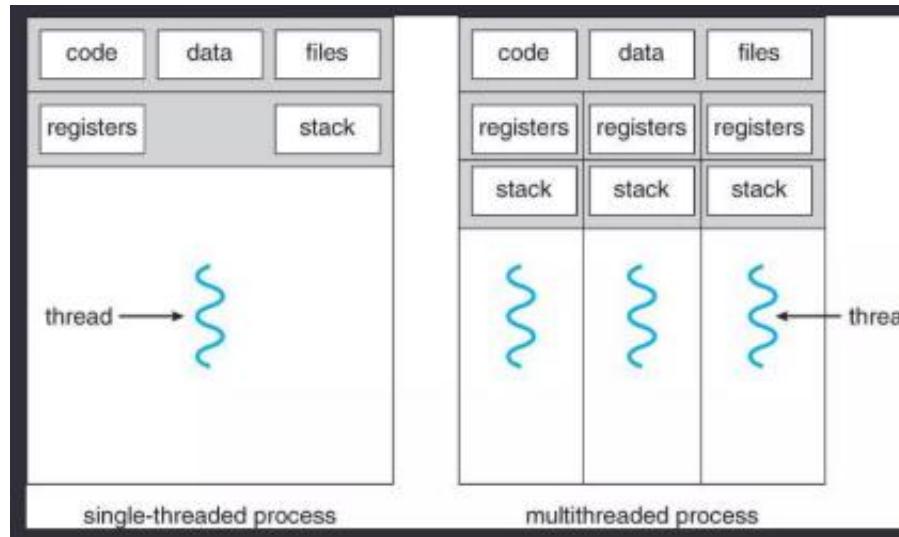
```
QrCodes.txt
1 www.google.com
2 https://docs.python.org/3/tutorial/index.html
3 12345/678910/11
```

```
Test
QR_Code_1.png
QR_Code_2.png
QR_Code_3.png
QrCodes.txt
test.ipynb
test.py
```

■ Python Projects – 4) Thread programming

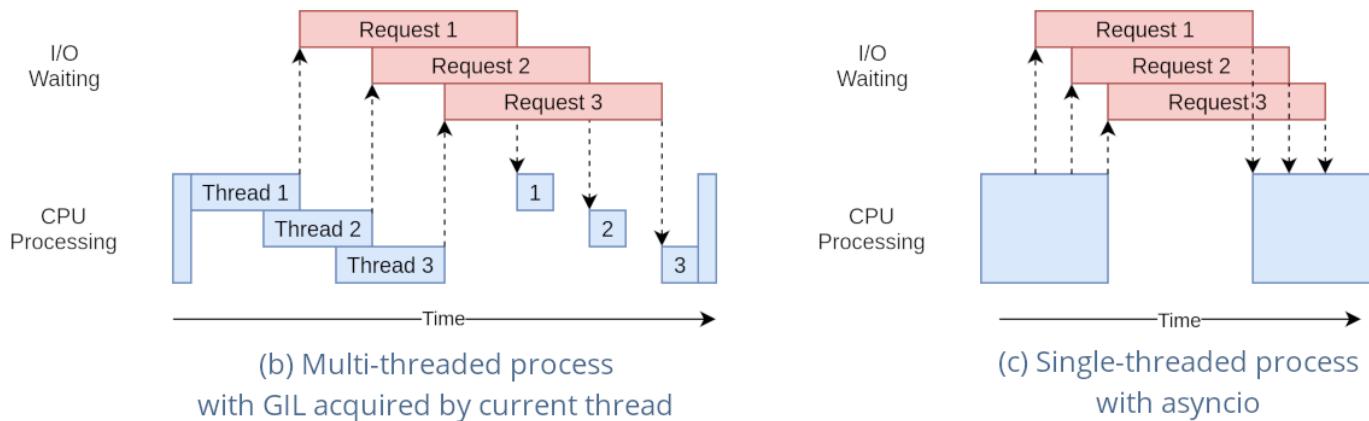
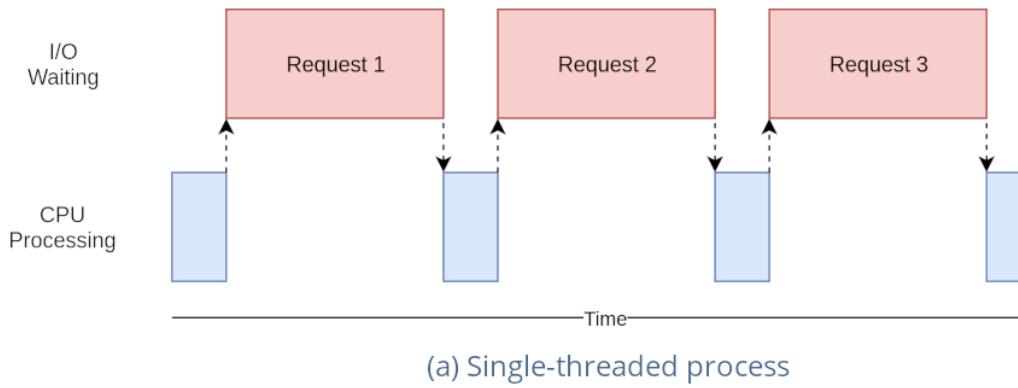
❖ What is thread?

- A thread is a flow of execution through the process code, with its own program counter, system registers and stack.
- A thread is also called a lightweight process. Threads provide a way to improve application performance through parallelism.
- Each thread belongs to exactly one process and no thread can exist outside a process.



Python Projects – 4) Thread programming

❖ What is thread?



■ Python Projects – 4) Thread programming

- ❖ Thread function only runs when the main code is executed.

```
import threading  
import time  
  
def thread_1():  
    while True:  
        print('Thread 1 is working')  
        time.sleep(0.7)  
  
t1 = threading.Thread(target=thread_1)  
t1.daemon = True  
t1.start()  
  
while True:  
    print('Main loop')  
    time.sleep(1)
```

■ Python Projects – 4) Thread programming

❖ Multi threads code stop by event

```
import threading  
import time  
  
def thread_1(stop_thread_1):  
    print('Thread_1 Start')  
    while True:  
        print('Thread_1 working')  
        time.sleep(0.5)  
        if stop_thread_1.is_set():  
            print('Thread_1 is terminated')  
            break  
  
def thread_2(stop_thread_2):  
    print('Thread_2 Start')  
    while True:  
        print('Thread_2 working')  
        time.sleep(0.7)  
        if stop_thread_2.is_set():  
            print('Thread_2 is terminated')  
            break
```

```
stop_thread_1 = threading.Event()  
stop_thread_2 = threading.Event()  
t1 = threading.Thread(target=thread_1, args=(stop_thread_1,))  
t1.start()  
t2 = threading.Thread(target=thread_2, args=(stop_thread_2,))  
t2.start()  
  
cnt=0  
while True:  
    cnt = cnt+1  
    print('Main code working')  
    print(cnt)  
    time.sleep(1)  
    if cnt>10 :  
        stop_thread_1.set()  
        t1.join()  
    if cnt>15 :  
        stop_thread_2.set()  
        t2.join()
```

Python Projects – 5) Auto mouse

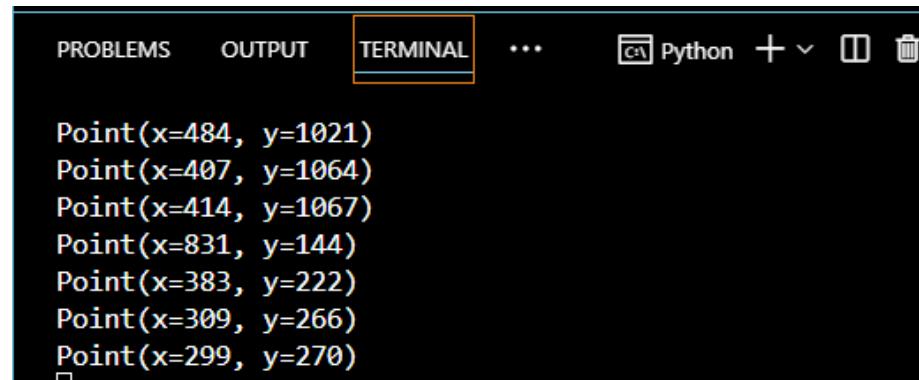
- ❖ Output the current mouse position

```
pip install pyautogui
```

```
pip install pyperclip
```

Type these commands on terminal window

```
import pyautogui  
import time  
  
while True:  
    print(pyautogui.position())  
    time.sleep(0.3)
```



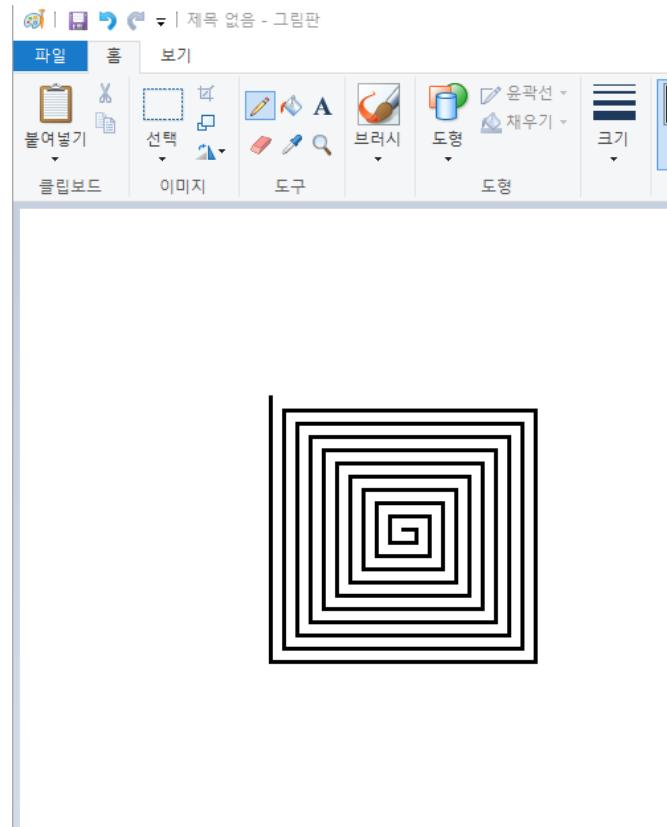
The terminal window shows the following output:

```
PROBLEMS OUTPUT TERMINAL ... ⌂ Python + ⌄ ⌁ ⌂  
Point(x=484, y=1021)  
Point(x=407, y=1064)  
Point(x=414, y=1067)  
Point(x=831, y=144)  
Point(x=383, y=222)  
Point(x=309, y=266)  
Point(x=299, y=270)  
□
```

■ Python Projects – 5) Auto mouse

- ❖ Make Python code that can work as following:

Make a python code that draw the line automatically.



■ Python Projects – 6) Excel file R/W

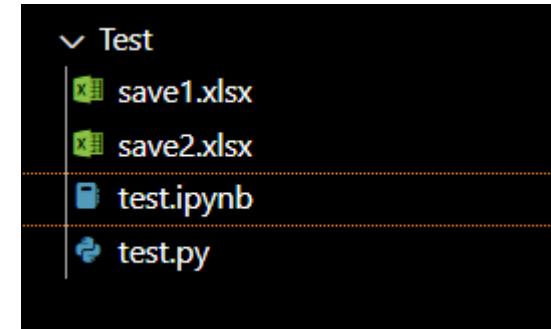
- ❖ Write data as 'data1.xlsx' . Read data from saved file and change Row & Col

```
import pandas as pd
from openpyxl import load_workbook

d1 = ["kim", "dae", "sung"]
d2 = ["1977", "2000", "2024"]
d3 = ["1", "23", "47"]
d4 = ['22','33','55']
data = [d1,d2,d3,d4]
df1 = pd.DataFrame(data)
print(df1)
df1.to_excel(r'save1.xlsx', index=False, header=False)

load_wb = load_workbook(r'save1.xlsx')
load_ws = load_wb.active

Rld_data = [[0 for j in range(load_ws.max_row)] for i in range(load_ws.max_column)]
for r in range(1, load_ws.max_column+1):
    for c in range(1, load_ws.max_row+1):
        Rld_data[r-1][c-1] = load_ws.cell(c,r).value
df2 = pd.DataFrame(Rld_data)
print(df2)
df1.to_excel(r'save2.xlsx', index=False, header=False)
```



	0	1	2
0	kim	dae	sung
1	1977	2000	2024
2	1	23	47
3	22	33	55
	0	1	2
0	kim	1977	1
1	dae	2000	23
2	sung	2024	47
			3

■ Python Projects – 7) GUI

- ❖ Make a simple gui using tkinter library

```
import tkinter  
import tkinter.font  
  
win = tkinter.Tk()  
win.title("Test title")  
win.geometry("400x200")  
win.resizable(False,False)  
  
font = tkinter.font.Font(size=40)  
label = tkinter.Label(win, text='hello',  
font=font)  
label.pack()  
  
win.mainloop()
```



■ Python Projects – 7) GUI

- ❖ Make a simple gui using tkinter library

```
import tkinter  
import tkinter.font  
  
win = tkinter.Tk()  
win.title("Test title")  
win.geometry("400x200")  
win.resizable(False,False)  
  
font = tkinter.font.Font(size=40)  
label = tkinter.Label(win, text=" ", font=font)  
label.pack()  
  
cnt = 0  
def run_1sec():  
    global cnt  
    cnt = cnt+1  
    label.config(text=str(cnt))  
    win.after(1000, run_1sec)  
  
run_1sec()  
win.mainloop()
```





Q & A
