

PYTHON PROGRAMMING

LECTURE 0: PYTHON OVERVIEW

goorm

KAIST AI
Graduate School of AI



Origin of Python

- 1989년 크리스마스에 연구실이 닫혀 있음
→ 심심해서 프로그래밍 언어를 만듦
- 영국 코미디 그룹 몬티 파이썬에서 이름을 따옴



Guido Van Rossum

Features of Python

- **플랫폼 독립적인 인터프리터 언어**
 - 코드와 인터프리터만 있다면 어디서든 실행 가능!
- **완전 객체 지향 언어**
 - 모든 것은 객체다
- **동적 타이핑 언어**
 - 코드를 실행하던 중에 타이핑
 - 덕 타이핑 (Duck typing)
 - 이것은 껍뻍거리므로 오리이다

Why Python? High Productivity!

- 쉬운 문법 & 다양한 기능
 - 높은 생산성

Hello World! in ...

```
1 class HelloWorldApp {  
2     public static void main(String[] args) {  
3         System.out.println("Hello World!");  
4     }  
5 }
```

JAVA

```
1 print ("Hello World!")
```

Python

Why Python? High Productivity!

- 쉬운 문법 & 다양한 기능
 - 높은 생산성

Sum 1 to 10 in ...

```
1  for (i = 1; i < 11; i++) {  
2  |      System.out.println (i)  
3  |  }
```

JAVA

```
1  for i in range(1, 11):  
2  |      print (i)
```

Python

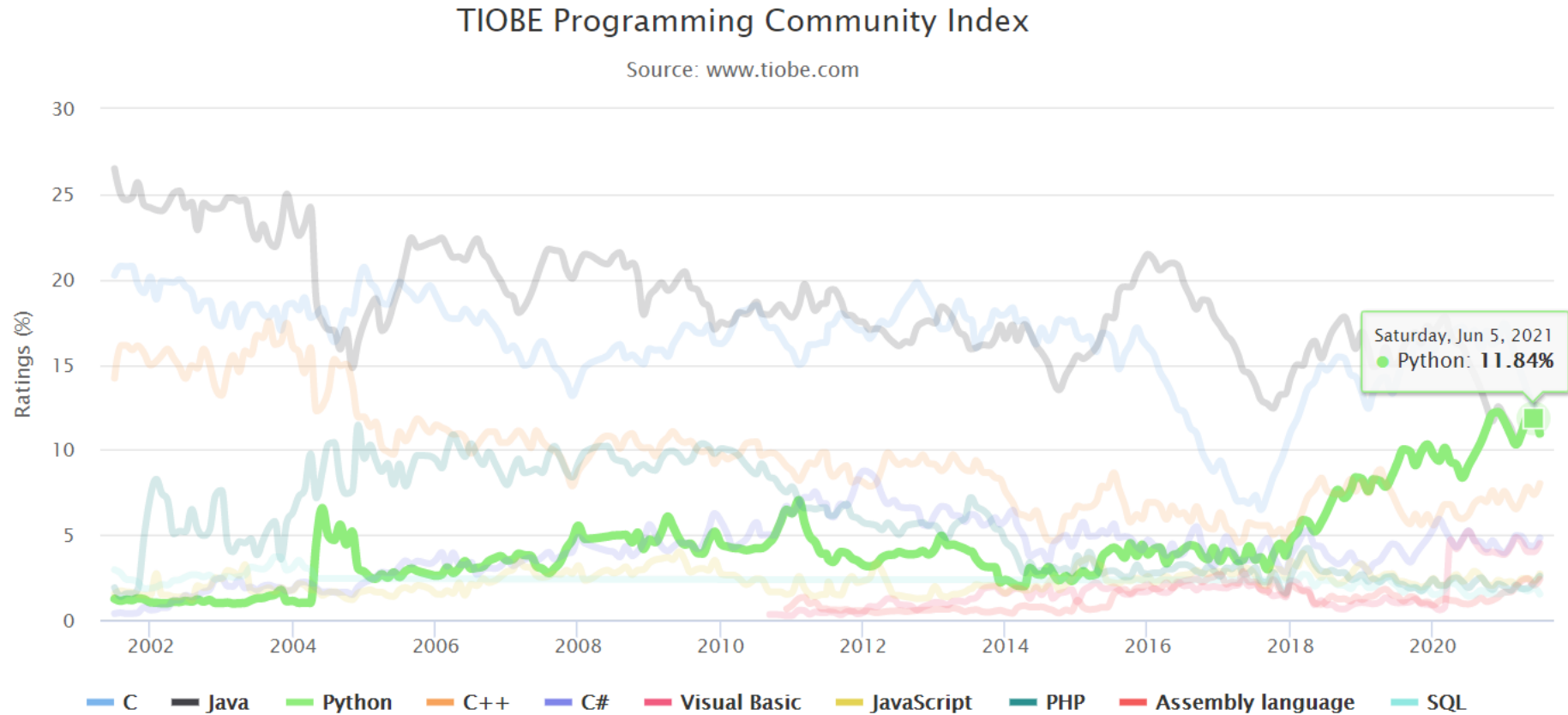
Why Python? Various Libraries!

- 다양한 라이브러리
 - 쉬운 라이브러리 설치 및 관리
 - 수많은 사람들이 다양한 라이브러리에 기여 및 공개



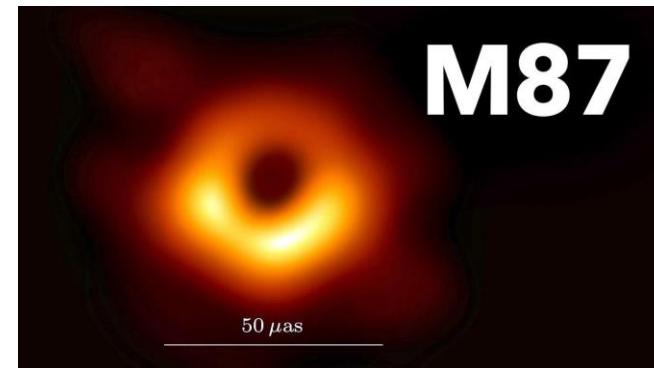
Why Python? Widely Used!

- 널리 쓰임
 - 인터넷에서 관련 문서 및 도움을 받기 유리함



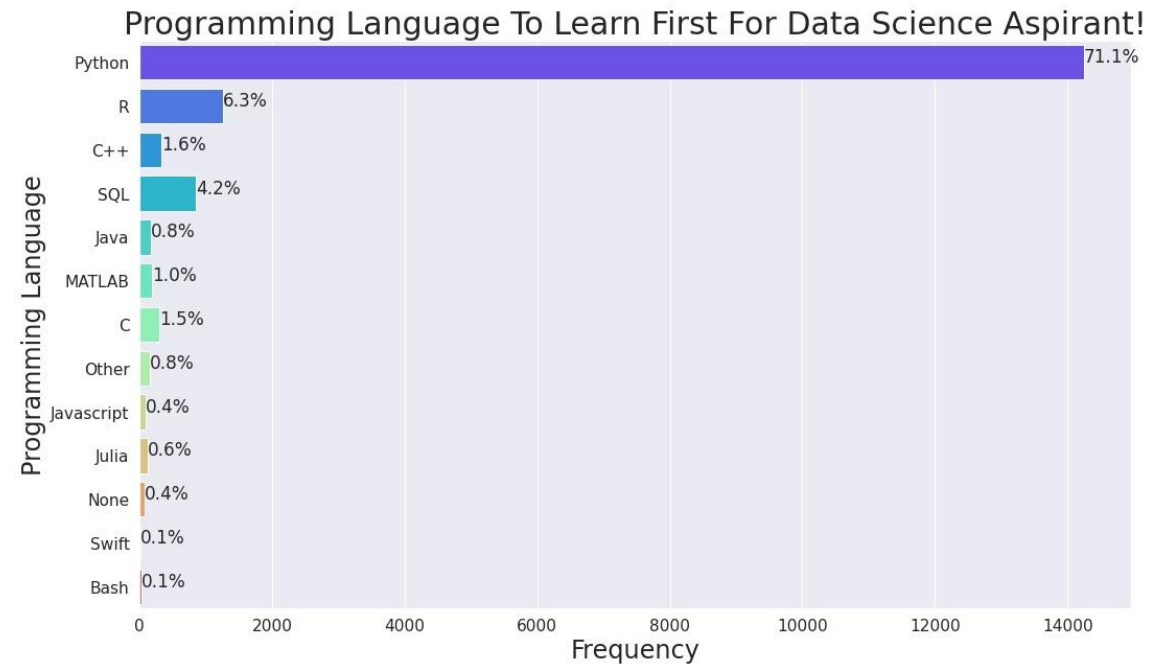
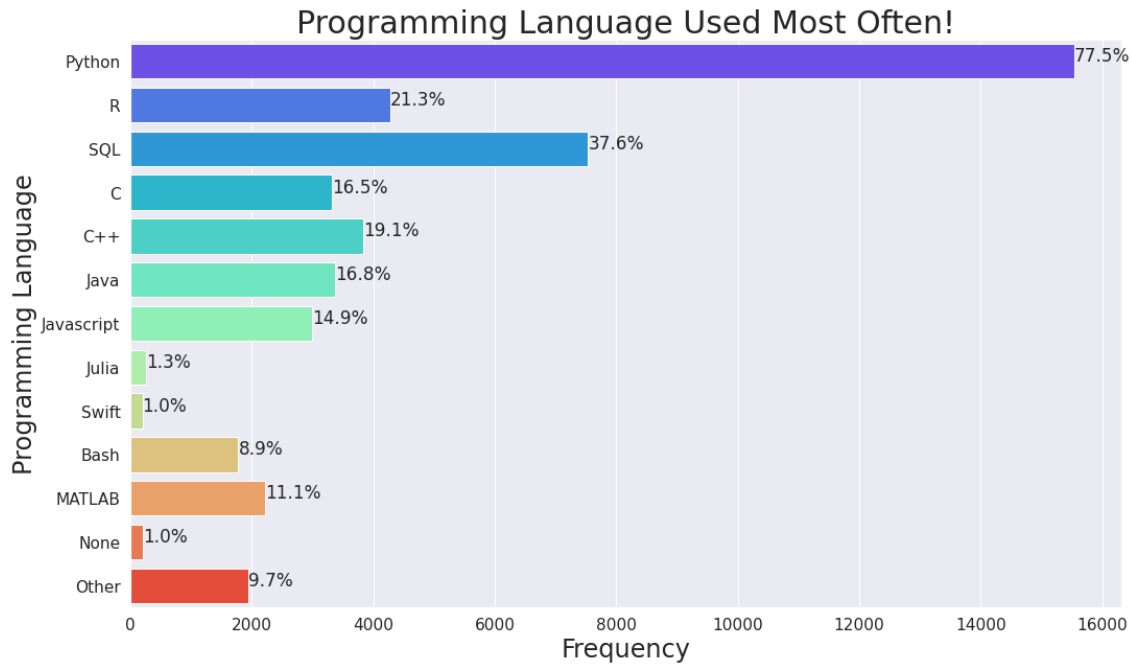
Why Python? Widely Used!

- 널리 쓰임
 - 분야를 가리지 않고 다양한 목적으로 사용



Why Python? Widely Used!

- 널리 쓰임
 - 데이터 분석 및 가공에서 두각을 드러냄



Thus, Why Python?

- 쉽고 다양한 문법
- 설치 및 관리가 쉬운 수많은 라이브러리
- 구글링으로 대부분의 문제를 해결가능

→ 매우 높은 생산성

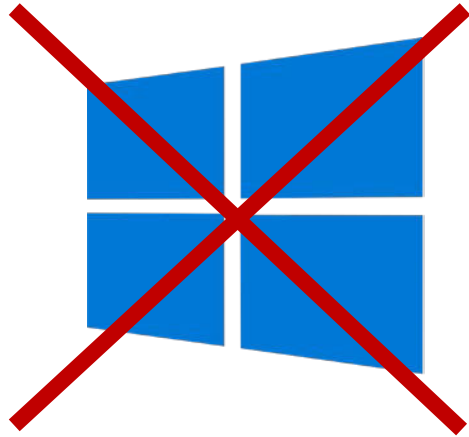
Life is short, You need Python

PYTHON PROGRAMMING
LECTURE 1: ENVIRONMENT

Development Environment

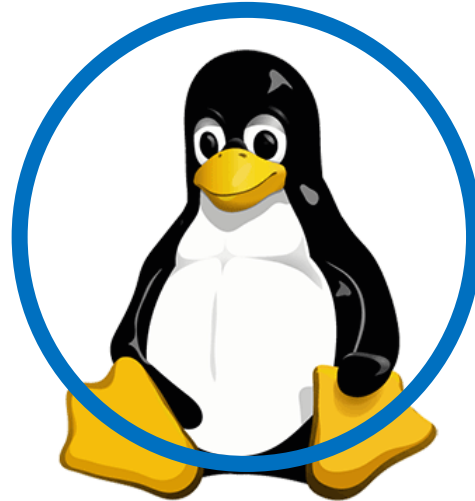
- ✓ 운영체제
- ✓ Python 인터프리터
 - python 3.9가 최신
 - 요즘엔 대부분 3.X를 쓰기 때문에 대부분 상관없음
- ✓ 코드 편집기 (Editor, IDE)
- ✓ 패키지 관리자

Operating System



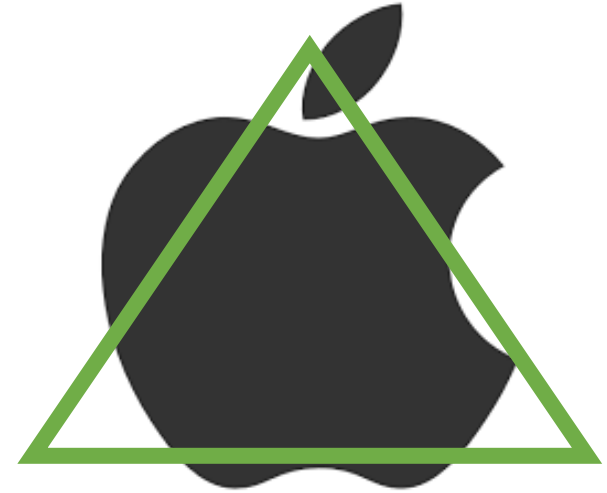
Window

친숙함, 워드, 한컴 사용가능
특정 라이브러리 설치 어려움



Linux

서버 호환 쉬움
라이브러리 설치 용이, 무료
대부분의 사용자가 친숙하지 않은 편



MAC

서버 사용 용이
비쌈

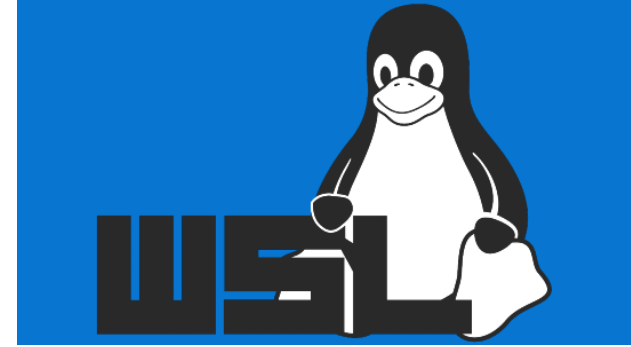
저는 윈도우 밖에 없는데요.....?

Alternative Options for Window Users



VirtualBox

가상 환경 사용



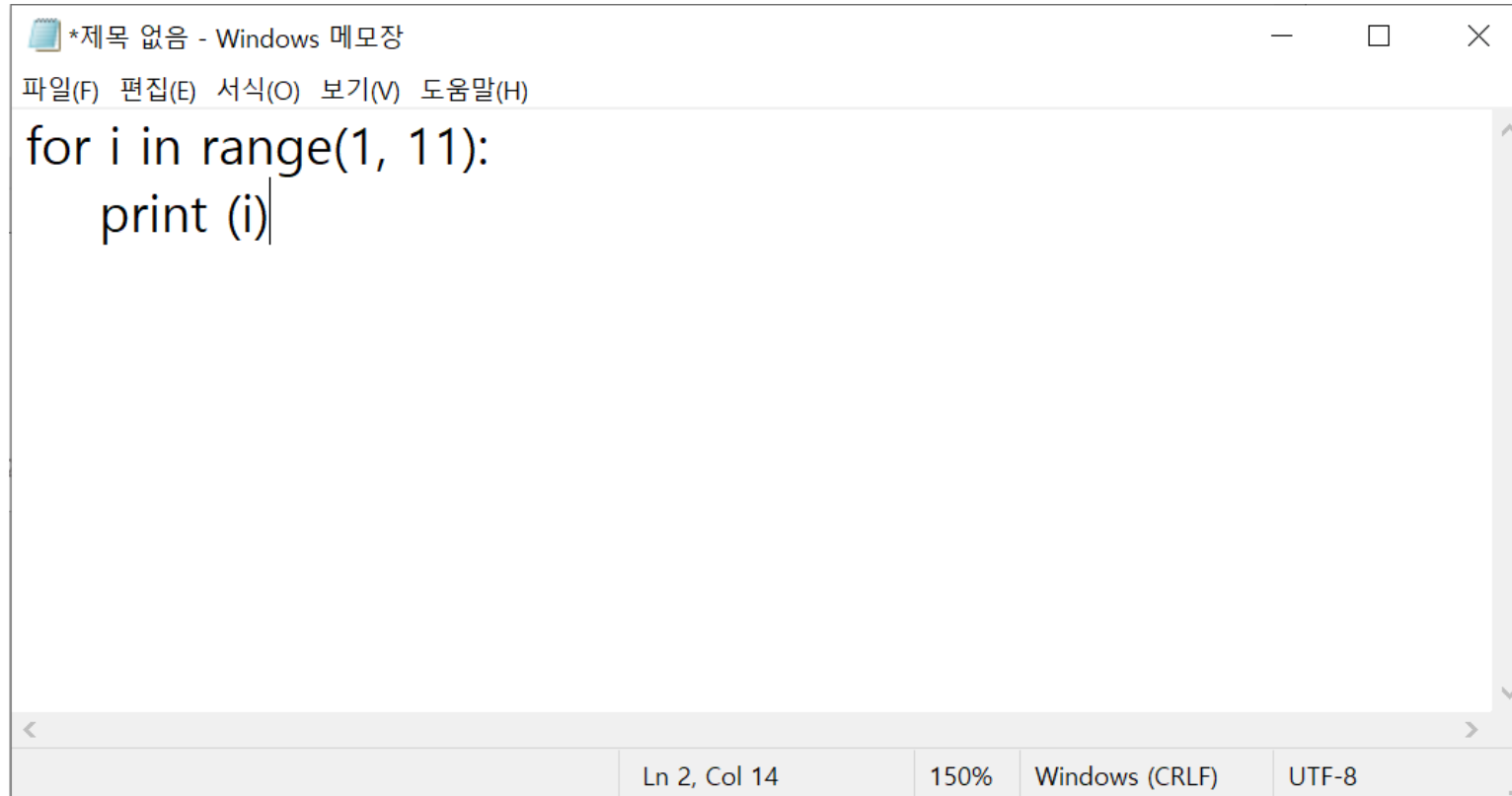
WSL 사용

colab

goormide

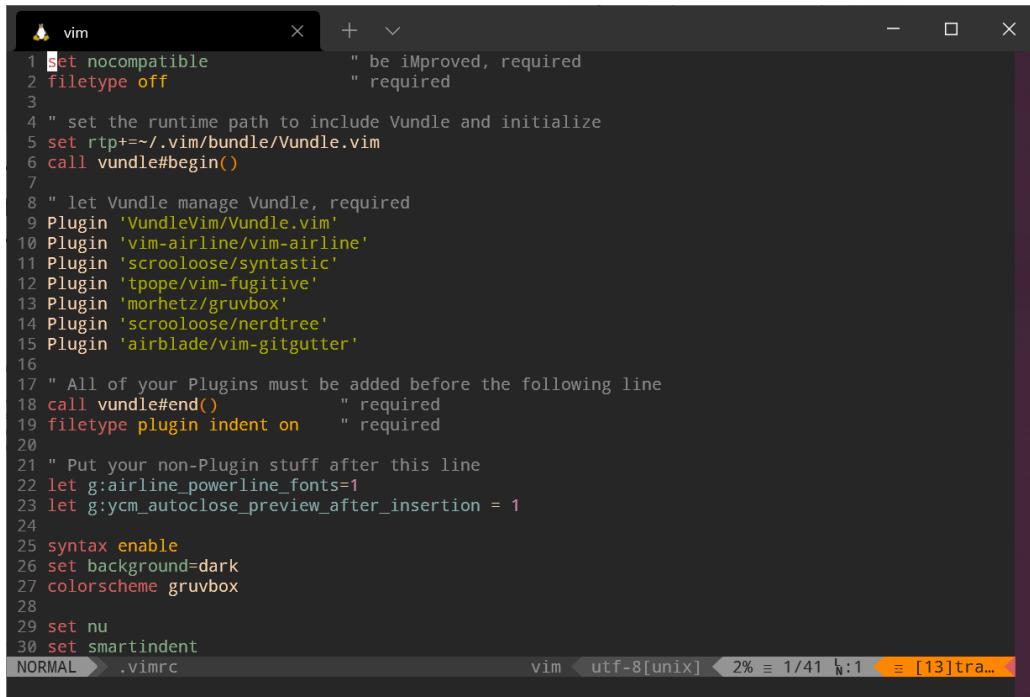
클라우드 기반
서비스 사용

Code Editor



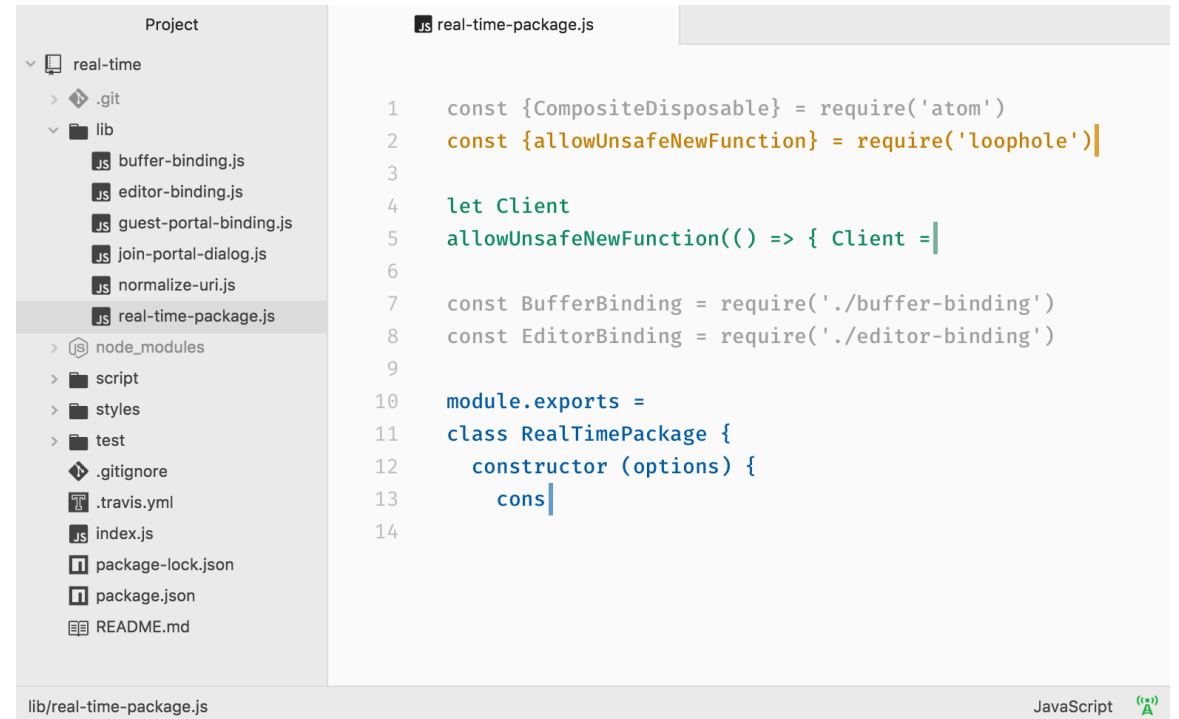
메모장으로도 코딩은 가능하지만....

Code Editor



```
vim
1 set nocompatible " be iMproved, required
2 filetype off " required
3
4 " set the runtime path to include Vundle and initialize
5 set rtp+=~/.vim/bundle/Vundle.vim
6 call vundle#begin()
7
8 " let Vundle manage Vundle, required
9 Plugin 'VundleVim/Vundle.vim'
10 Plugin 'vim-airline/vim-airline'
11 Plugin 'scrooloose/syntastic'
12 Plugin 'tpope/vim-fugitive'
13 Plugin 'morhetz/gruvbox'
14 Plugin 'scrooloose/nerdtree'
15 Plugin 'airblade/vim-gitgutter'
16
17 " All of your Plugins must be added before the following line
18 call vundle#end() " required
19 filetype plugin indent on " required
20
21 " Put your non-Plugin stuff after this line
22 let g:airline_powerline_fonts=1
23 let g:ycm_autoclose_preview_after_insertion = 1
24
25 syntax enable
26 set background=dark
27 colorscheme gruvbox
28
29 set nu
30 set smartindent
NORMAL .vimrc vim utf-8[unix] 2% 1/41 1:1 [13]tra...
```

VIM



```
Project
└─ real-time
   ├── .git
   └── lib
       ├── buffer-binding.js
       ├── editor-binding.js
       ├── guest-portal-binding.js
       ├── join-portal-dialog.js
       ├── normalize-uri.js
       └── real-time-package.js
node_modules
script
styles
test
.gitignore
.travis.yml
index.js
package-lock.json
package.json
README.md

real-time-package.js
1 const {CompositeDisposable} = require('atom')
2 const {allowUnsafeNewFunction} = require('loophole')
3
4 let Client
5 allowUnsafeNewFunction(() => { Client =
6
7 const BufferBinding = require('./buffer-binding')
8 const EditorBinding = require('./editor-binding')
9
10 module.exports =
11 class RealTimePackage {
12   constructor (options) {
13     cons
14
JavaScript
```

Atom

<https://www.jetbrains.com/ko-kr/pycharm>

Integrated Development Environment



Visual Studio Code

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



PyCharm

<https://www.jetbrains.com/ko-kr/pycharm>

Web-based IDE



Jupyter Notebook



Jupyter Lab

Cloud-based IDE

The logo for Google Colab, featuring the word "colab" in a bold, orange, sans-serif font. The "c" and "o" are connected, and the "l" is a single vertical bar.

Google Colab

<https://colab.research.google.com/>

이후 강의에서 메인으로 사용 예정

The logo for Goorm IDE, featuring the word "goormide" in a black, lowercase, sans-serif font.

Goorm IDE

<https://ide.goorm.io/>

Package & Environment Manager

- 세상엔 외부 라이브러리가 너무 많아요
 - 설치 및 관리를 자동화할 도구가 필요
 - Java의 Maven, Node의 NPM, Linux의 APT
 - 다양한 환경 간 쉬운 전환 필요 (Environment Management)



Package & Environment Manager



PIP + Virtual env

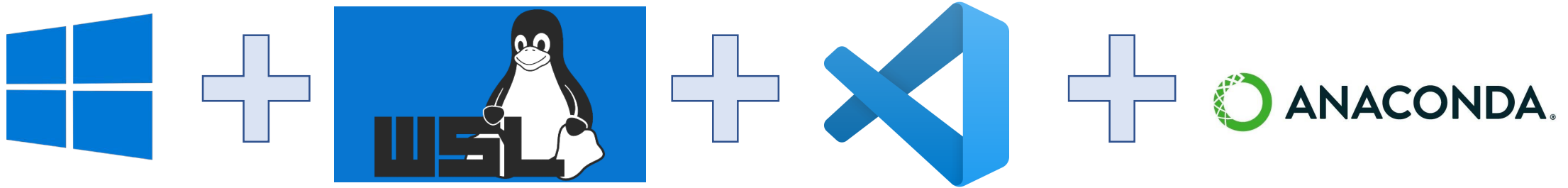
Python 기본 패키지 관리 프로그램



Anaconda3

<https://www.anaconda.com/products/individual>
기계학습 및 수치해석 특화 패키지 관리 프로그램

Overview

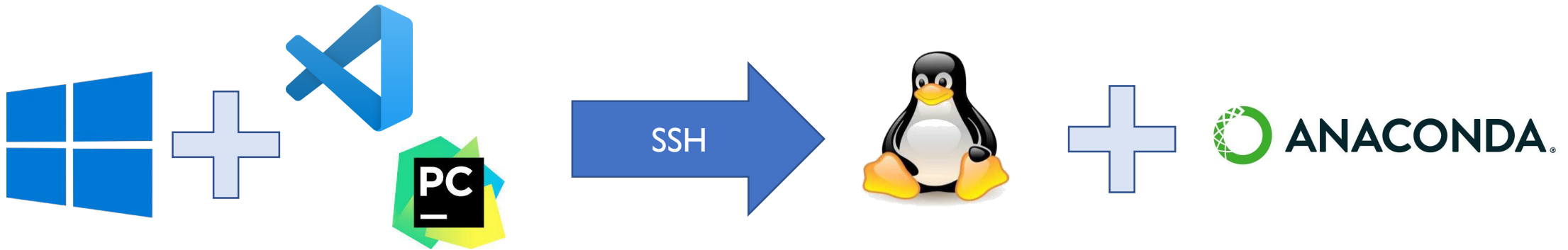


개인적으로 생각하는 Best Local 환경

- 장점: Window와 Linux를 아우르는 개발 환경
- 단점: GPU 지원 안됨 (윈도우11에서 지원 예정), 매우 복잡한 설치

Local

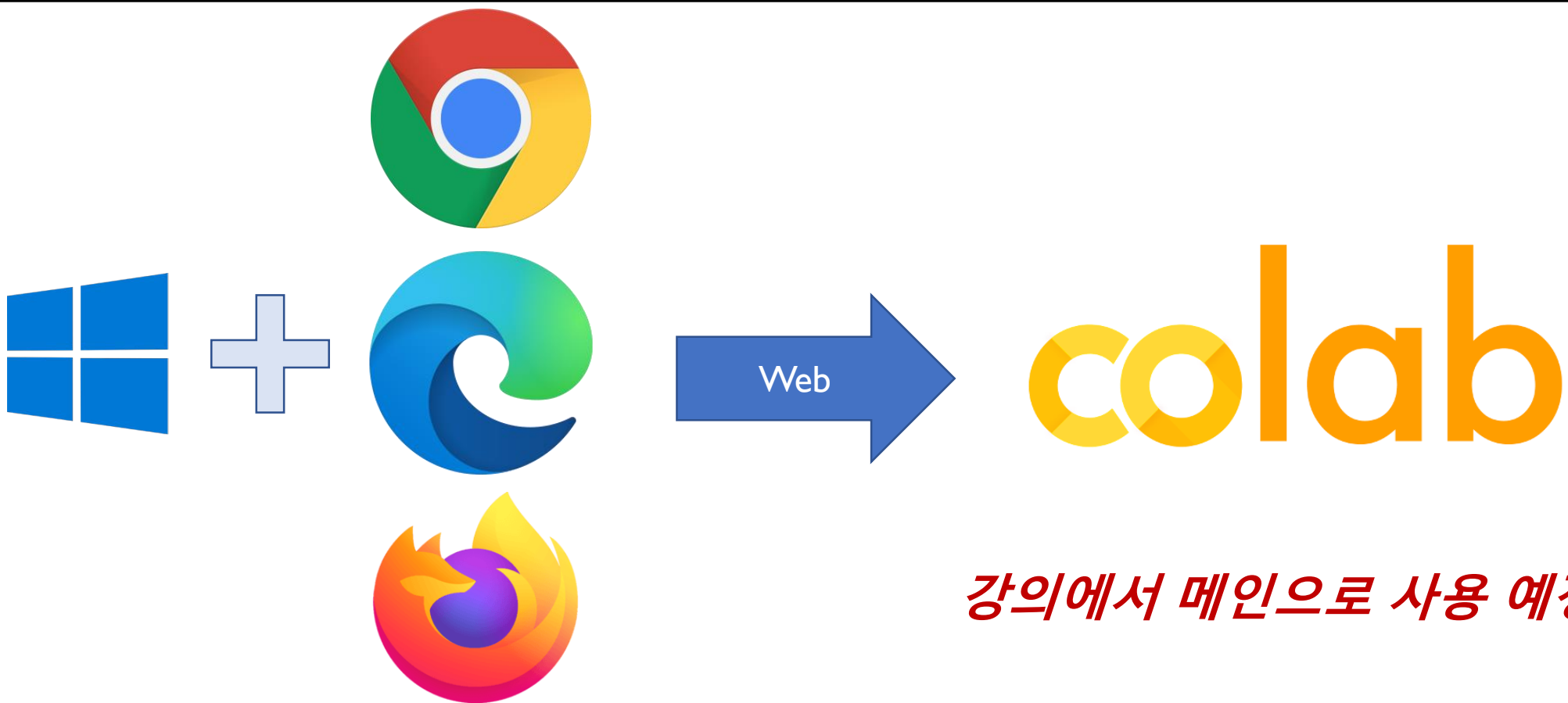
Server



사람들이 많이 쓰는 환경

- 장점: 고성능의 서버자원 이용 가능
- 단점: 서버 필요, Docker 연결이 다소 불편

Overview



강의에서 메인으로 사용 예정

대안이 없는 사람들을 위한 개발 환경

- 장점: 매우 쉽고 간단, 따로 설치할 것이 없음
- 단점: 패키지 관리가 매우 불편, 12시간 Session 만료, 실개발 환경으로 권장하지 않음

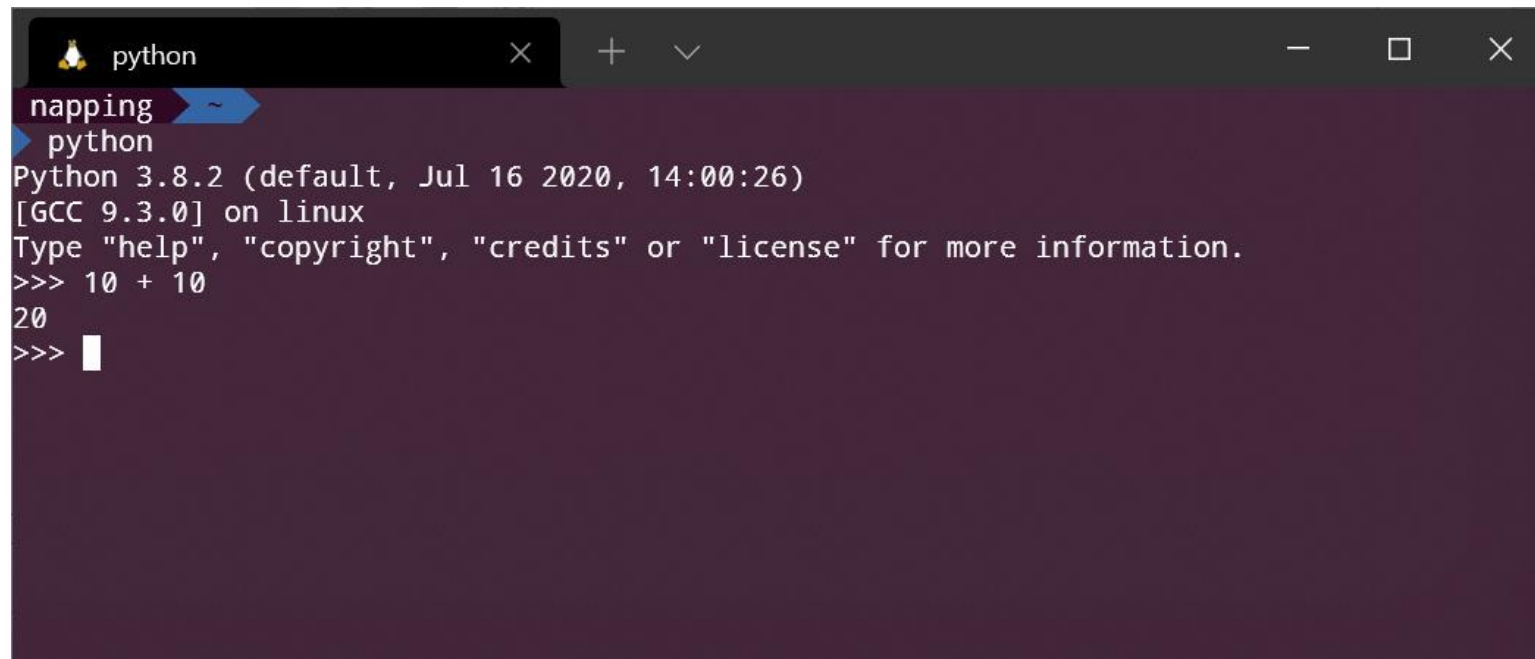
과제 0: 원하는 개발 환경 세팅하기

PYTHON PROGRAMMING

LECTURE 2: JUPYTER

Jupyter Overview

파이썬의 기본 실행 환경 → Interactive Shell

A screenshot of a terminal window titled 'python'. The window shows the prompt 'napping ~' and 'python'. Below this, it displays 'Python 3.8.2 (default, Jul 16 2020, 14:00:26)' and '[GCC 9.3.0] on linux'. It then shows the prompt 'Type "help", "copyright", "credits" or "license" for more information.' followed by the input '>>> 10 + 10' and the output '20'. The prompt '>>>' is followed by a cursor.

```
python
napping ~
python
Python 3.8.2 (default, Jul 16 2020, 14:00:26)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 10 + 10
20
>>> 
```

일반적으로 .py 파일으로 실행하지만, 파이썬의 interactive함을 사용할 수는 없을까?

Jupyter Overview

IP[y]: IPython
Interactive Computing



- Ipython을 커널을 기반으로한 Interactive 파이썬 셀 프로그래밍
 - .ipynb 파일 확장자
 - Jupyter라는 웹 기반 IDE를 기반으로 실행
 - VsCode 및 PyCharm, Colab에서 사용 가능
- 미디어, 코드, 수식 등을 하나의 문서 형태 표현 가능

Jupyter on Colab

asn1.ipynb - Colaboratory

https://colab.research.google.com/drive/1DS37rAOrpVF3GvhlMhuQreaxJm9dZ71i#scrollTo=Ete39aBix...

asn1.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 오전 5:17에 마지막으로 저장됨

댓글 공유

RAM 디스크 수정 가능

1-1. abs

```
def my_abs(number):  
    pass  
  
test1 = 1.7  
test2 = -8  
assert abs(test1) == my_abs(test1)  
assert abs(test2) == my_abs(test2)  
print("통과")
```

1-2. any

```
[ ] def my_any(iterable):  
    pass  
  
test1 = [True, 7 == 4, 'Something', False]  
test2 = [3 > 5, 10 != 10, False, '', None]  
assert all(test1) == my_all(test1)  
assert all(test2) == my_all(test2)  
print("통과")
```

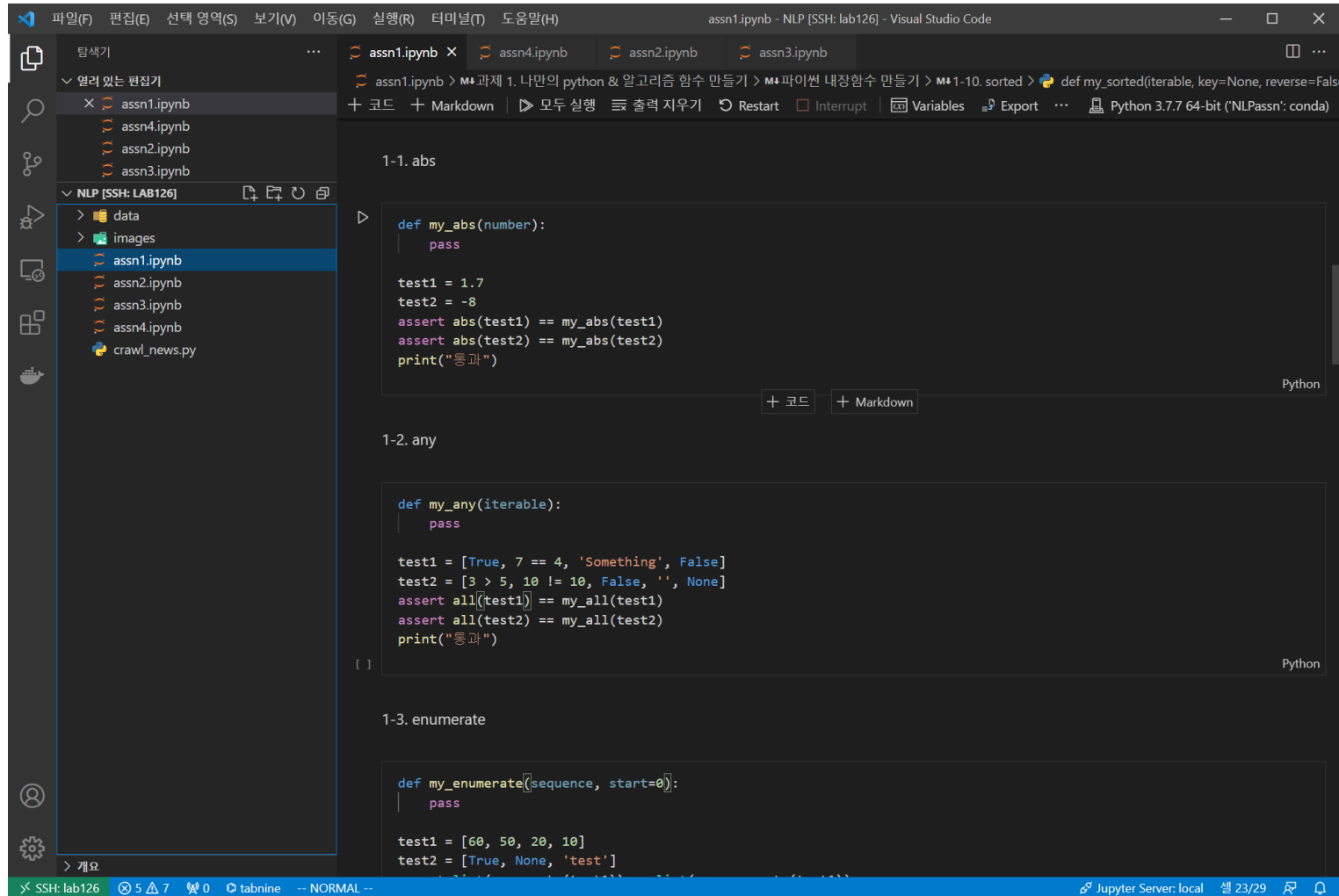
1-3. enumerate

```
[ ] def my_enumerate(sequence, start=0):  
    pass  
  
test1 = [60, 50, 20, 10]
```

디스크 69.07 GB 사용 가능

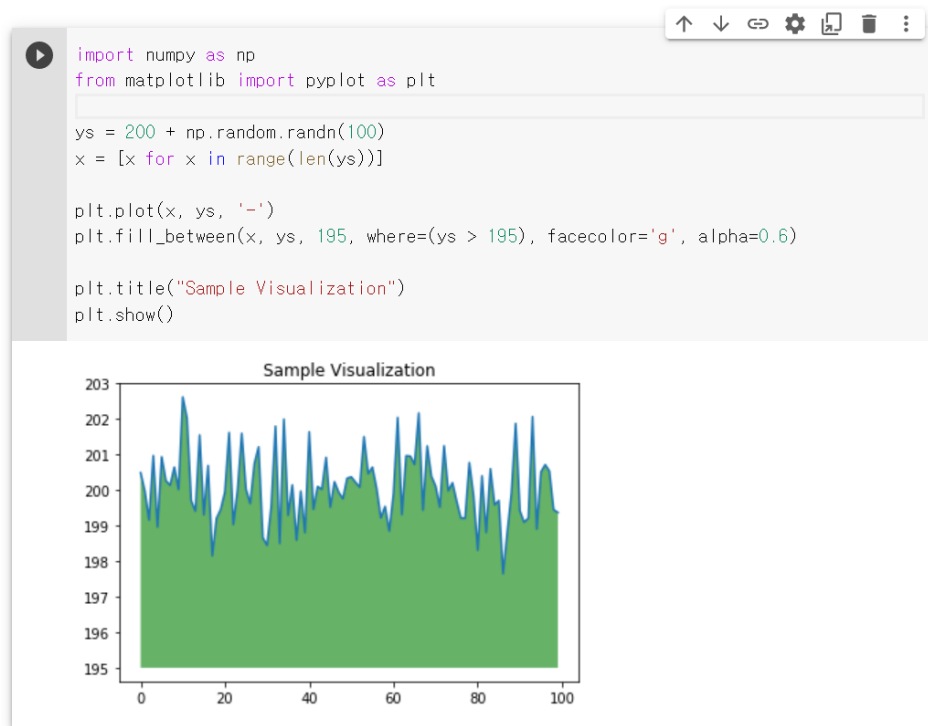
Colab 자체가 Jupyter 기반

Jupyter on VsCode



**VsCode Python 플러그인에
Jupyter Server가 내장됨**

Jupyter Cell



Code Cell

코드를 실행시키고 결과를 확인

The screenshot shows a Jupyter Markdown Cell with the following content:

Colaboratory란?

줄여서 'Colab'이라고도 하는 Colaboratory를 사용하면 브라우저에서 Python을 작성하고 실행할 수 있습니다. Colab은 다음과 같은 이점을 자랑합니다.

- 구성이 필요하지 않음
- GPU 무료 액세스
- 간편한 공유

학생이든, 데이터 과학자든, AI 연구원이든 Colab으로 업무를 더욱 간편하게 처리할 수 있습니다. [Colab 소개 영상](https://www.youtube.com/watch?v=inN8seMm7UI)에서 자세한 내용을 확인하거나 아래에서 시작해 보세요.

Markdown Cell

Markdown / HTML 문법으로 문서화

TODO for Today

- **과제 0: 환경 설정**
 - 여러가지 건드려보기 (단축키를 찾아보아도 좋아요~)
- **(Optional for linux & wsl users)**
 - Linux 기본 명령어 익히기
 - rm, mkdir, cp, cd, mv, ... (기타 구글링)
 - Linux 환경 꾸미기
 - 심심하다면 Shell을 oh-my-zsh 같은 거로 예쁘게 꾸며봅시다.

