



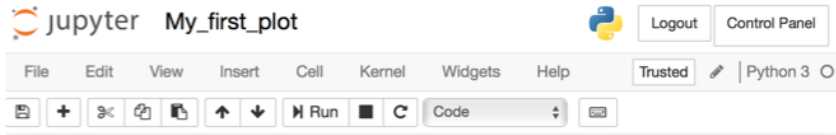
AI 프로그래밍 3

융합학과 권오영

oykwon@koreatech.ac.kr

JUPYTER NOTEBOOK

Jupyter Notebook



My first plot

We will use our favorite libraries, **NumPy** and **Matplotlib**, to make a plot of a periodic function. First, our beautiful equation:

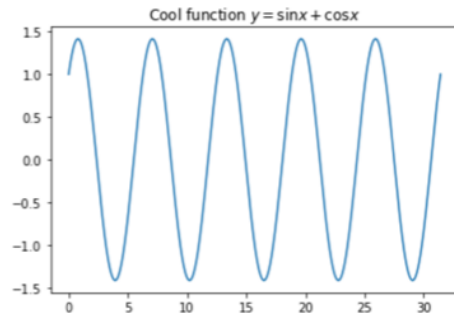
$$y = \sin x + \cos x$$

```
In [1]: import numpy
        from matplotlib import pyplot
        %matplotlib inline
```

The `numpy.linspace()` function creates an array of equally spaced numbers.

```
In [2]: x = numpy.linspace(0, 10*numpy.pi, 10**3)
        y = numpy.sin(x) + numpy.cos(x)
```

```
In [3]: pyplot.plot(x,y)
        pyplot.title('Cool function $ y = \sin\{x\} + \cos\{x\} $');
```



Jupyter header and tool bar.

A markdown cell, with title, explanation, and equation.

A code cell, setting things up with needed libraries.

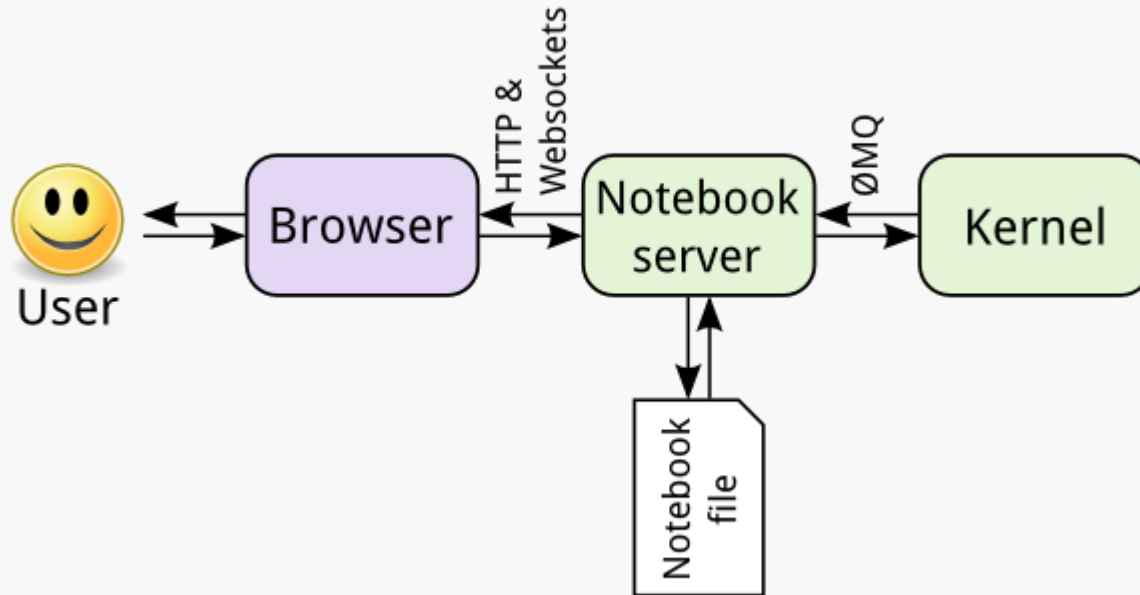
A short explanation.

Code cells assigning two array variables, then making a line plot.

- 주피터 (Jupyter) 프로젝트: 대화형 및 탐색적 컴퓨팅을 위한 오픈 소스 도구를 개발하는 광범위한 협업 프로젝트 (코드와 문서(콘텐츠)의 결합)
- Julia, Python, R 이 중점이지만 다양한 언어지원
- Jupyter Notebook은 2014년 후반부터 데이터 과학을 위해 가장 선호하는 환경으로 채택되면서 인기가 폭발적으로 증가
- Jupyter 노트북은 실행 코드, 방정식, 시각화 및 설명 텍스트 혼합을 지원하는 문서

주피터 노트북

- 노트북의 가장 중요한 구성 요소는 셀로, 노트북의 전체 내용은 셀로만 구성
 - 셀은 텍스트 또는 코드의 두 가지 형식 중 하나를 취함
 - 코드 셀은 입력 영역, 표시 영역 및 출력 영역의 세 영역으로 구성

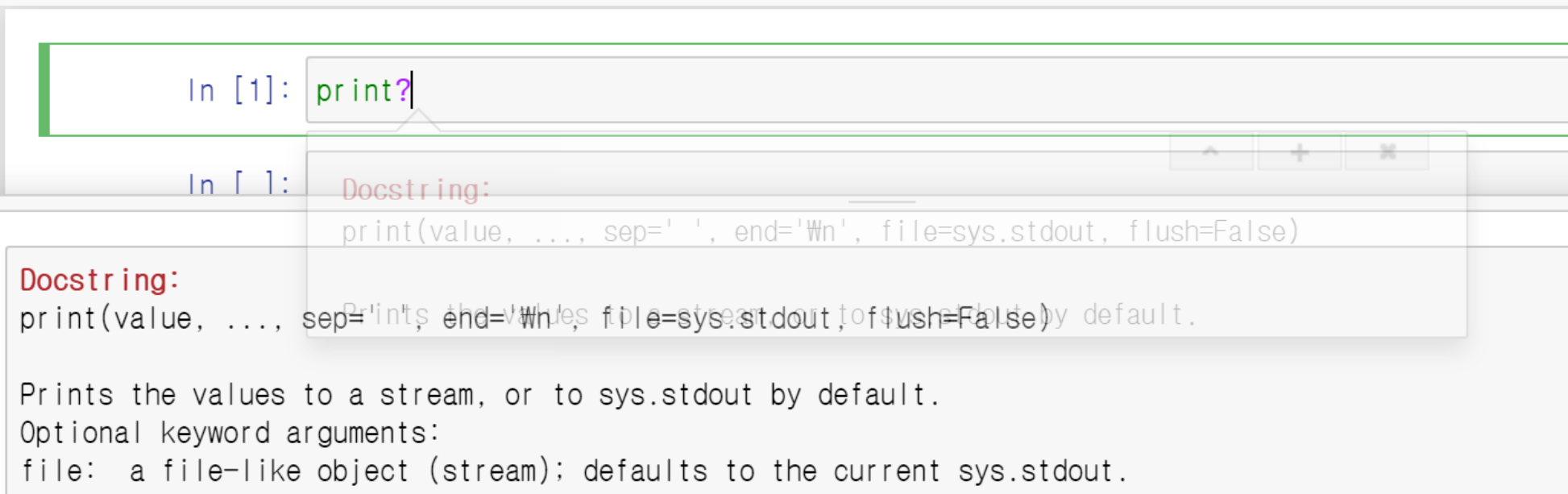


https://jupyter.readthedocs.io/en/latest/architecture/how_jupyter_ipython_work.html

주피터 노트북

- 입력 영역은 In []: 로 표시 (코드 입력)

- 노트북 셀에서 Tab 키는 자동 완성기능으로 이용되고, 완성된 메소드 shift-tab을 누르거나, 메소드 뒤에 물음표를 입력하고 셀을 실행하면 도움말이 나타남



The screenshot shows a Jupyter Notebook interface. The top part is a code input area with the text 'In [1]: print?'. Below this, a tooltip window is open, displaying the docstring for the 'print' function. The tooltip contains the following text:

```
In [ ]: Docstring:
        print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

        Prints the values to a stream, or to sys.stdout by default.
        Optional keyword arguments:
        file: a file-like object (stream); defaults to the current sys.stdout.
```

주피터 노트북

- **매직(magic) 명령어는 Jupyter 내에서만 작동하고 사용자가 언어/커널 관련 기능에 액세스 할 수 있도록 하는 메타 명령임**
 - 매직 명령어는 %로 시작하고, 전체 셀에 영향을 끼치는 명령은 %%로 시작함
 - %matplotlib inline 은 노트북에 포함 된 정적 이미지를 만들고,
%matplotlib notebook 은 상호작용(확대, 축소등)이 가능한 이미지를 생성함
 - %run은 외부 스크립트 (및 기타 노트북)를 실행하고 출력을 캡처하여 노트북에 표시
(예, %run my_script.py)

주피터 노트북 단축키

- 셀 선택 모드 (Command Mode)
[esc] 또는 [ctrl] + [m]를 눌러 셀이 아래와 같이 파란색이 된 상태(셀 선택 모드)에서 해당 단축키 누름

```
In [ ]: a = 10
```

위로 셀 추가

아래로 셀 추가

선택 셀 삭제

선택 셀 잘라내기 (삭제로 써도 무방)

선택 셀 복사하기

선택 셀 아래에 붙여넣기

선택 셀과 아래 셀과 합치기

실행결과 열기/닫기

Markdown으로 변경

Code로 변경

파일 저장

선택 셀의 코드 입력 모드로 돌아가기

[a]

[b]

[d][d] (d를 두 번 누름)

[x]

[c]

[p]

[shift] + [m]

[o]

[m]

[y]

[ctrl] + [s] 또는 [s]

[enter]

주피터 노트북 단축키

- 코드 입력 모드 (Edit Mode)
선택모드에서 [enter]를 누르거나 셀 내부를 클릭하면 아래와 같이 초록색이 된 상태(코드 입력 모드)가 되면 이 상태에서 다음과 같은 단축키 사용

```
In [ ]: a = 10|
```

선택 셀의 코드 전체 선택

[ctrl] + [a]

선택 셀 내 실행 취소

[ctrl] + [z]

선택 셀 내 다시 실행

[ctrl] + [y]

커서 위치 라인 주석처리

[ctrl] + [/]

선택 셀 코드 실행

[ctrl] + [enter]

선택 셀 코드 실행 후 다음 Cell로 이동 (없으면 새로 추가)

[shift] + [enter]

커서 위치에서 셀 둘로 나누기

[shift] + [ctrl] + [-]

파일 저장

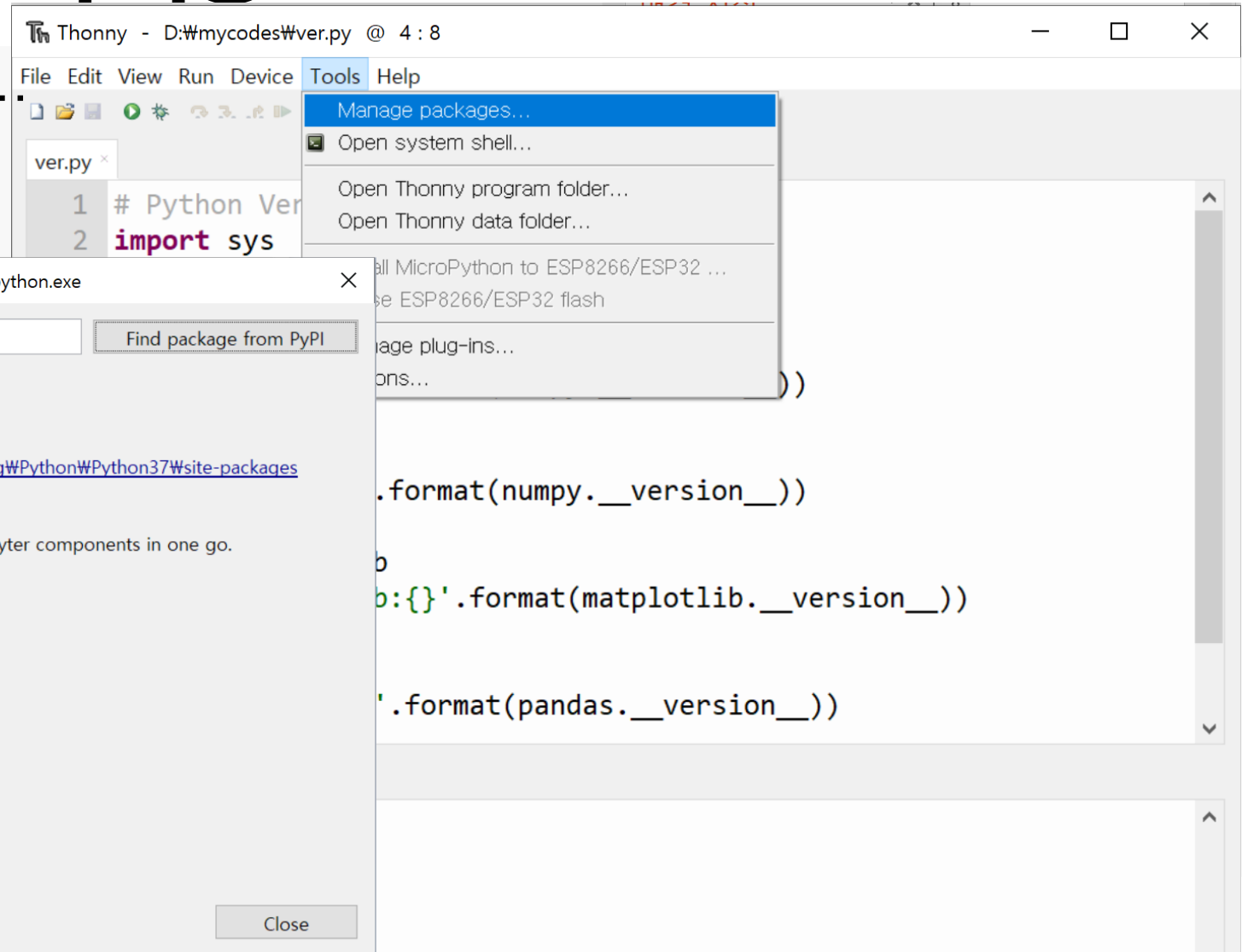
[ctrl] + [s]

셀 선택 모드로 돌아가기

[esc] 또는 [ctrl] + [m]

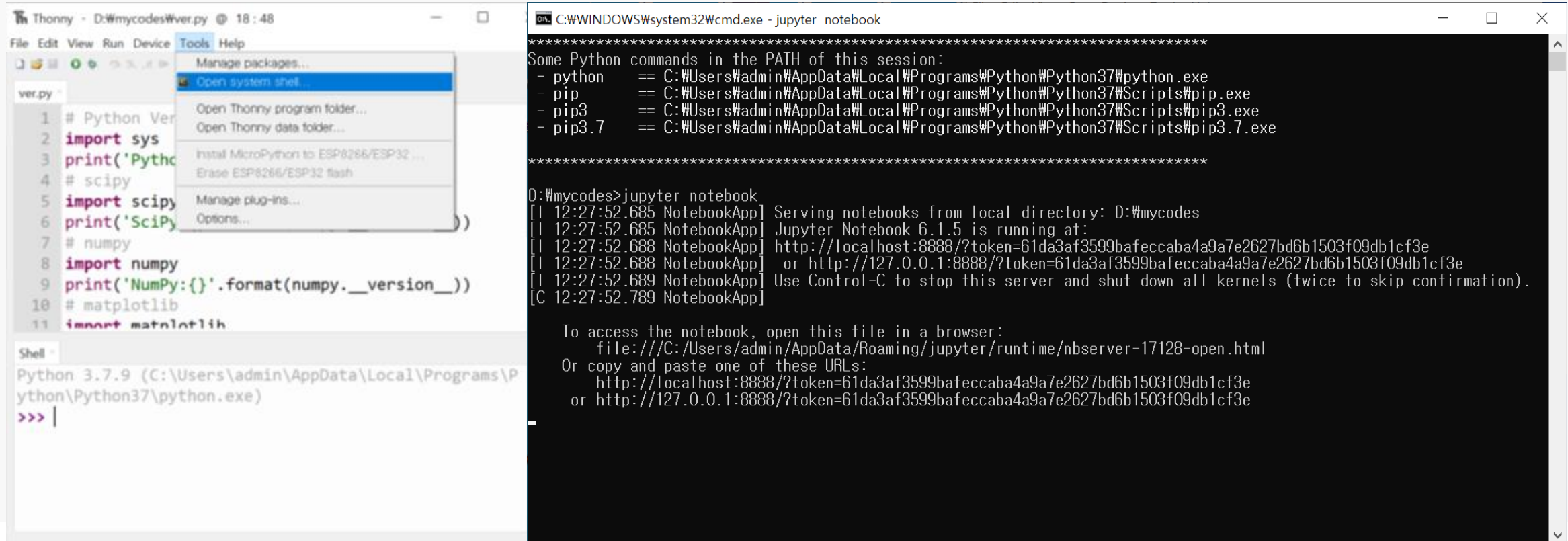
Thonny에서 주피터 노트북 사용

- Tools -> Manage packages...
- 필요한 패키지 설치



Thonny에서 주피터 노트북 사용

- Tools -> Open system shell...
- Command 창에서 주피터 노트북 실행 (jupyter notebook)



Thonny에서 주피터 노트북 사용

Home Page - Select or create a new tab

localhost:8888/tree

jupyter

Quit Logout

Files Running Clusters

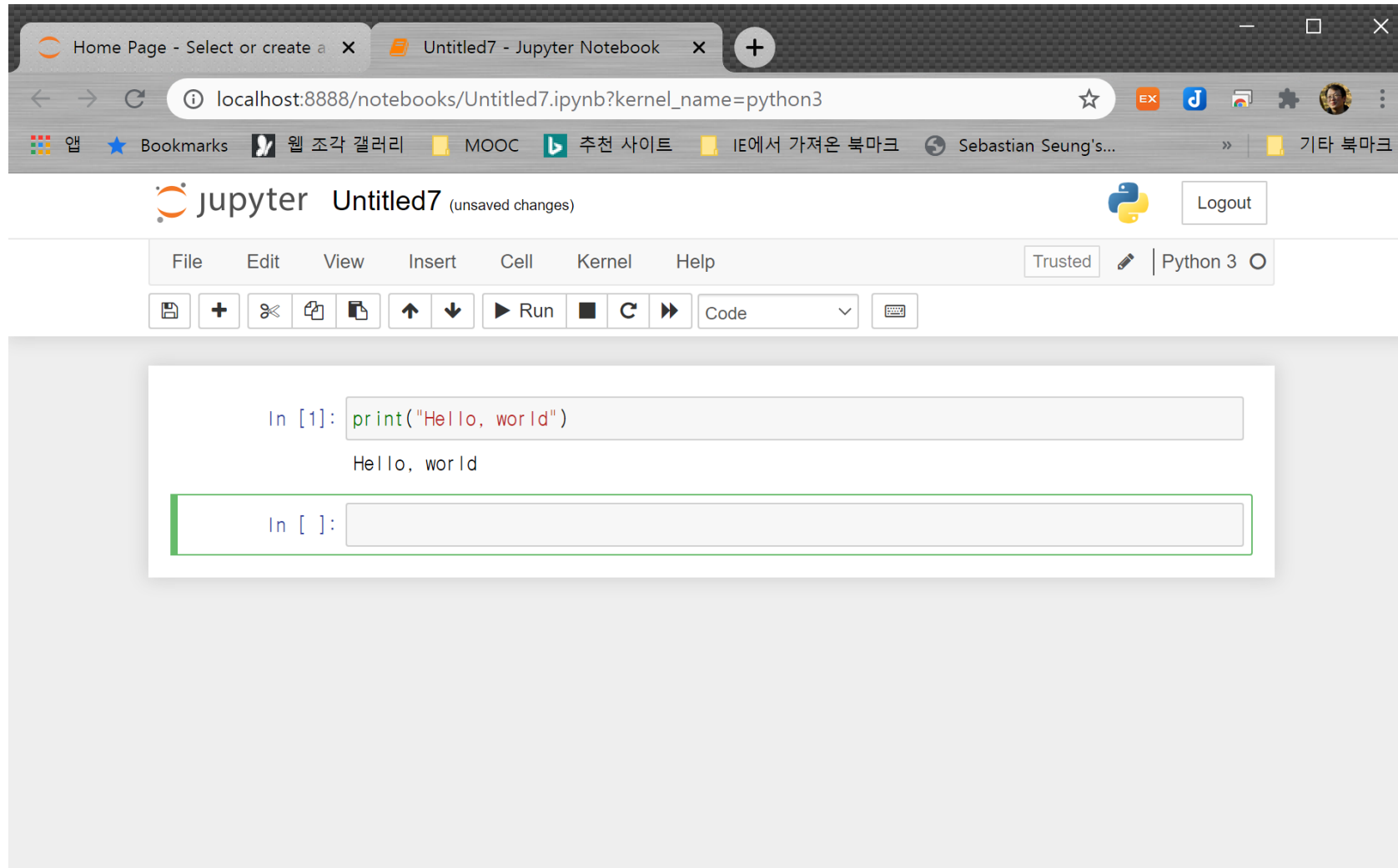
Select items to perform actions on them.

Upload New

Notebook:
Python 3
Other: Create a new notebook with Python 3
Text File
Folder
Terminal

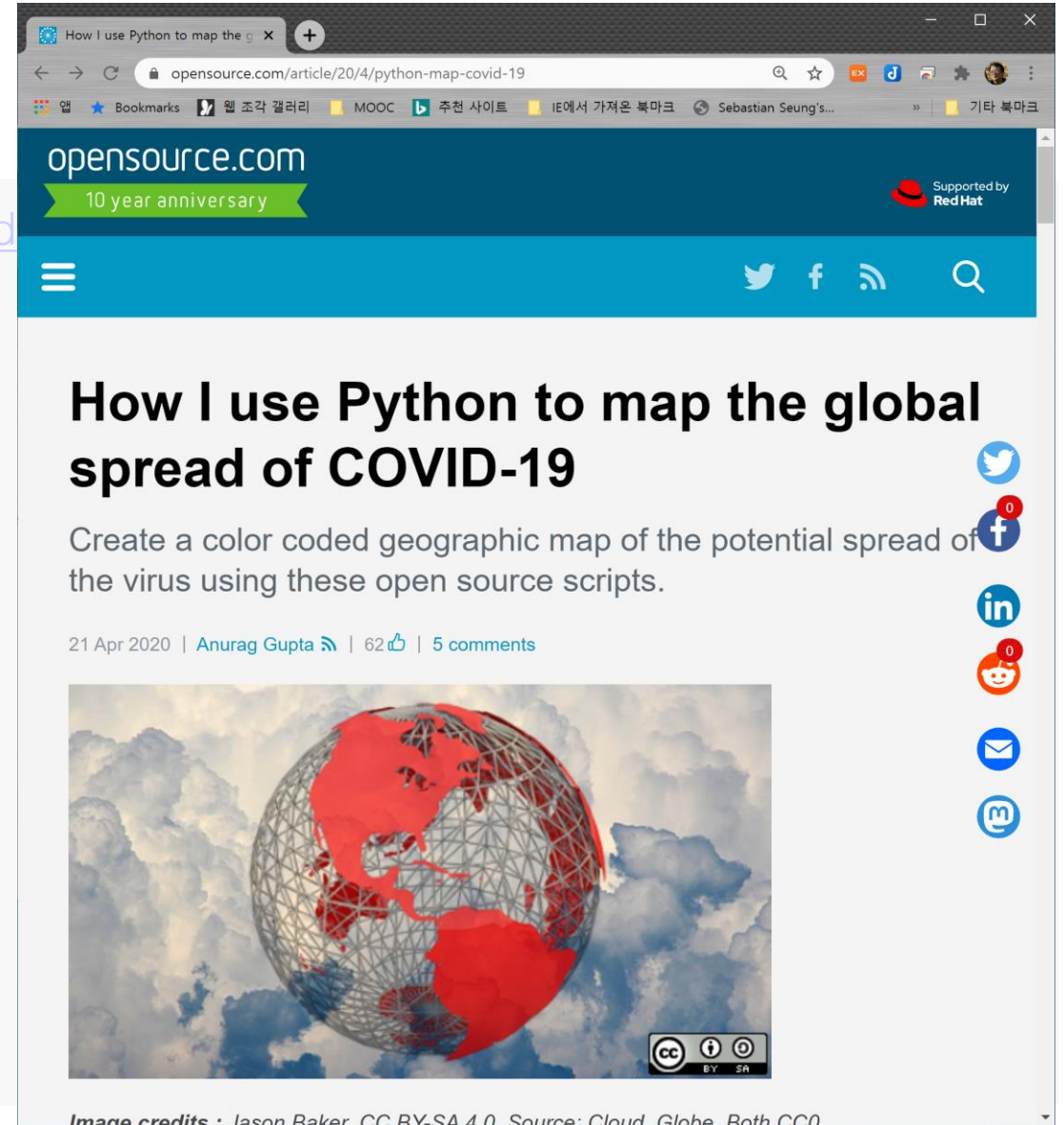
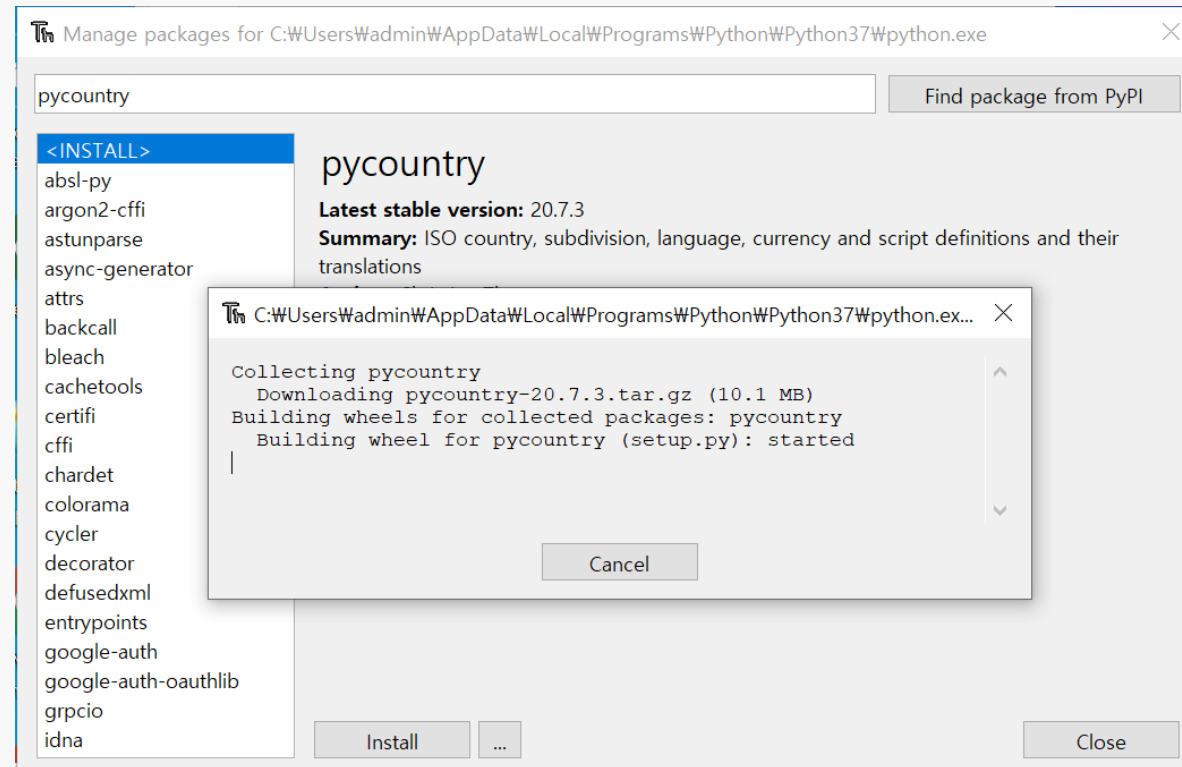
Name	Size	Modified
0		
financePyPKG		
myAlenv		
tradingSimulator		
Untitled Folder		
Untitled.ipynb	72 B	2달 전
Untitled1.ipynb	1.14 kB	2시간 전
Untitled2.ipynb	666 B	6일 전

Thonny에서 주피터 노트북 사용

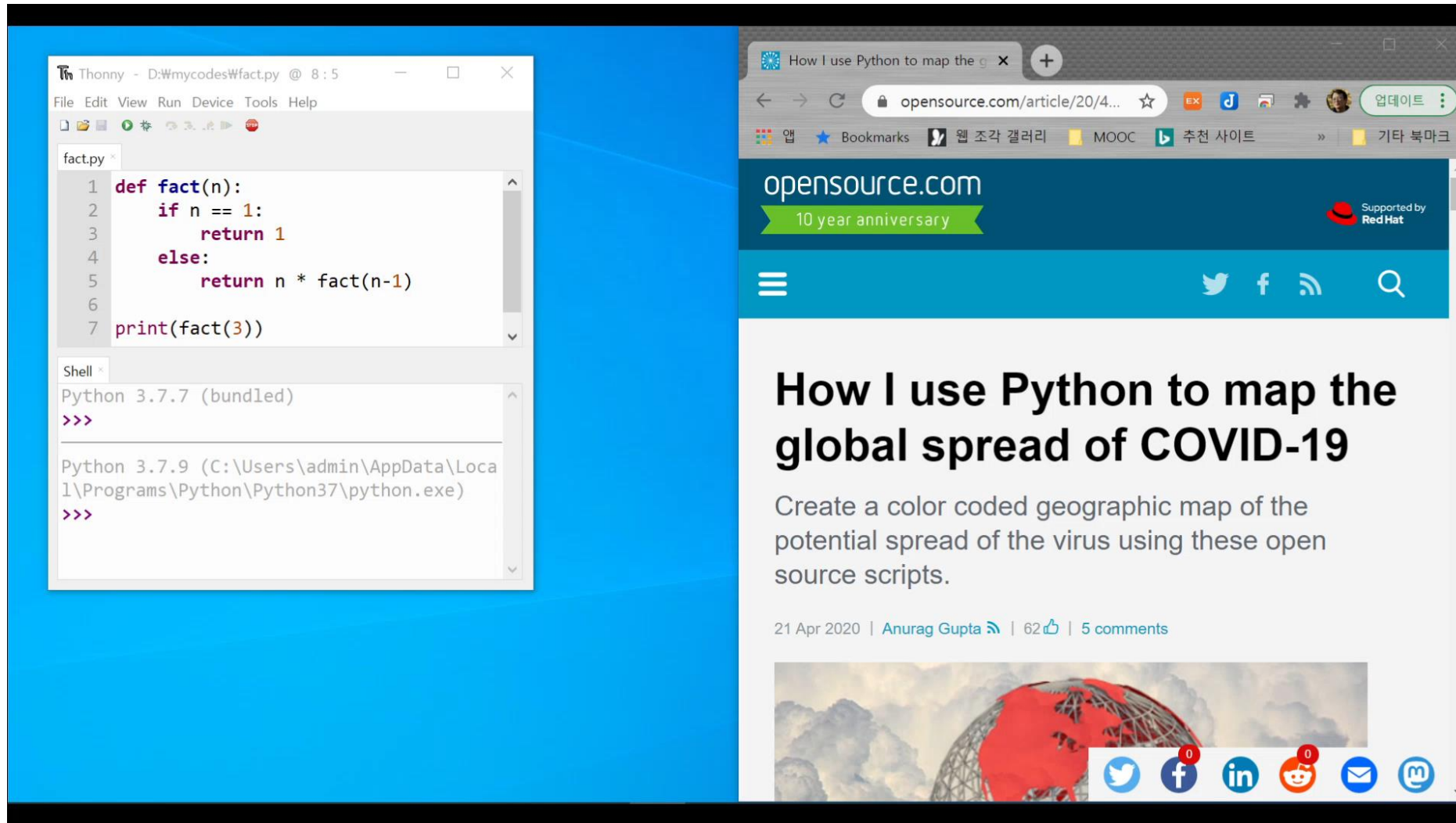


주피터 노트북 사용

<https://opensource.com/article/20/4/python-map-covid>

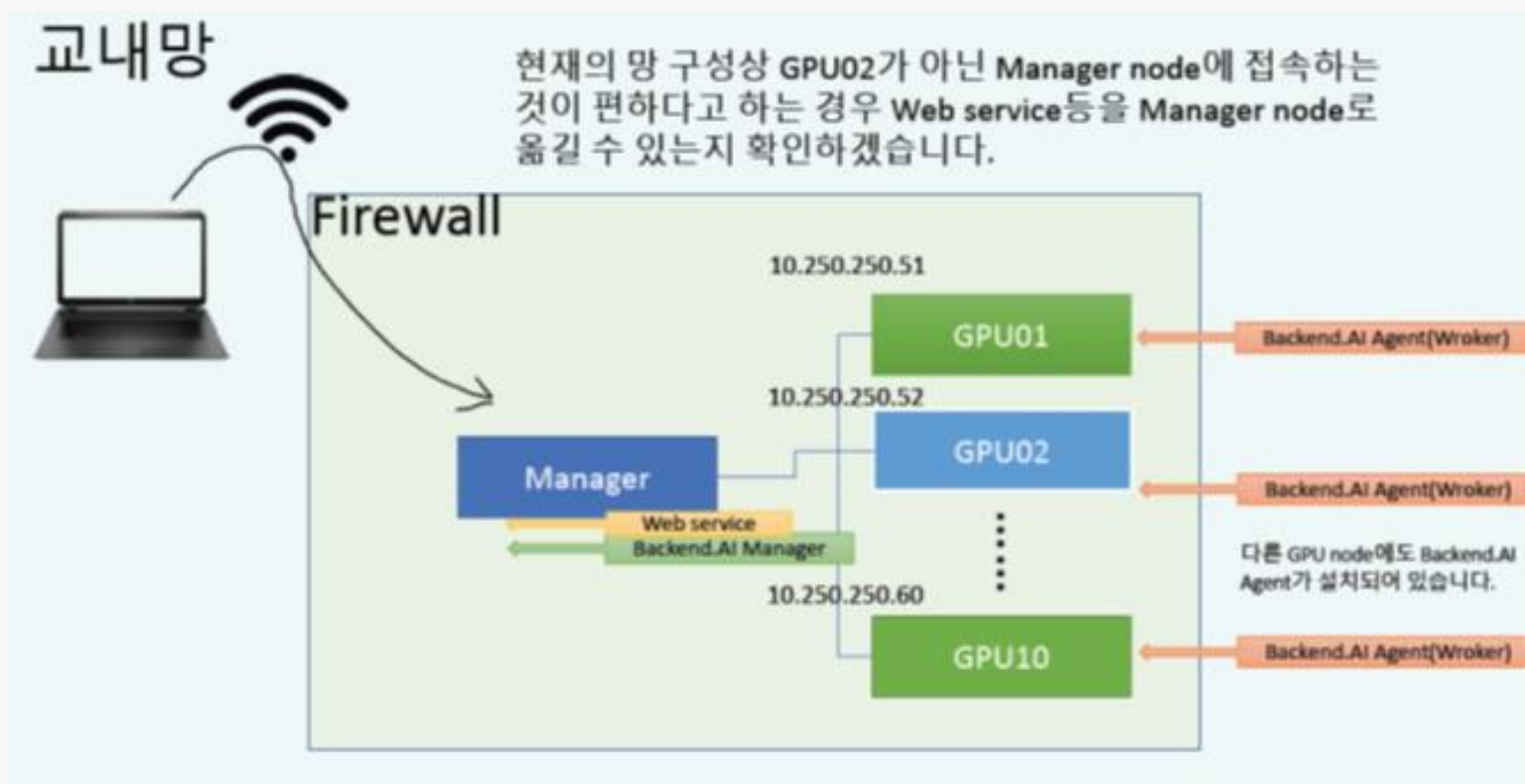


주피터 노트북 사용



클라우드 활용

- 교내 GPU 클러스터 구성하여 클라우드 서비스 제공



클라우드 활용

Backend.AI KRTech

Summary

Dashboard

Start Menu

+ START

Upload files

Create a new keypair

Maintain keypairs

Resource Statistics

Resource Group: default

CPU: 4/237

RAM: 8.00/1506.36GB

GPU: 0/24.00

Session: 1/30

System Resources

1 Active Sessions

Administration

Update environment images

Check resources

Start new session

Environments* TensorFlow (krtech)

Version* 2.2 / Python 3.6 / CUDA10.1

Resource Group* default Session name (optional)

Folder to mount

Resource allocation* large (4CPU 8GB 64MB)

Custom allocation

CPU 4 Core

RAM 8.00 GB

Shared Memory 0.0625 GB

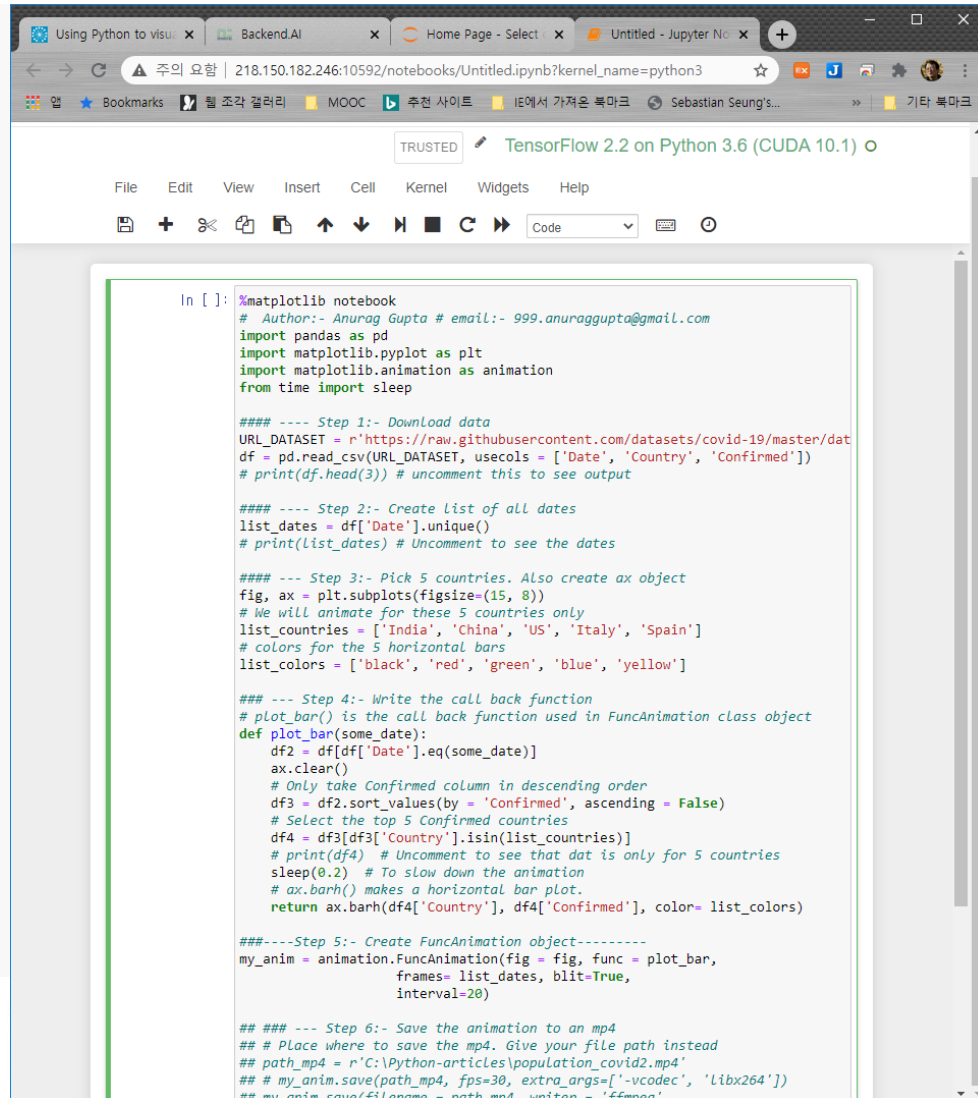
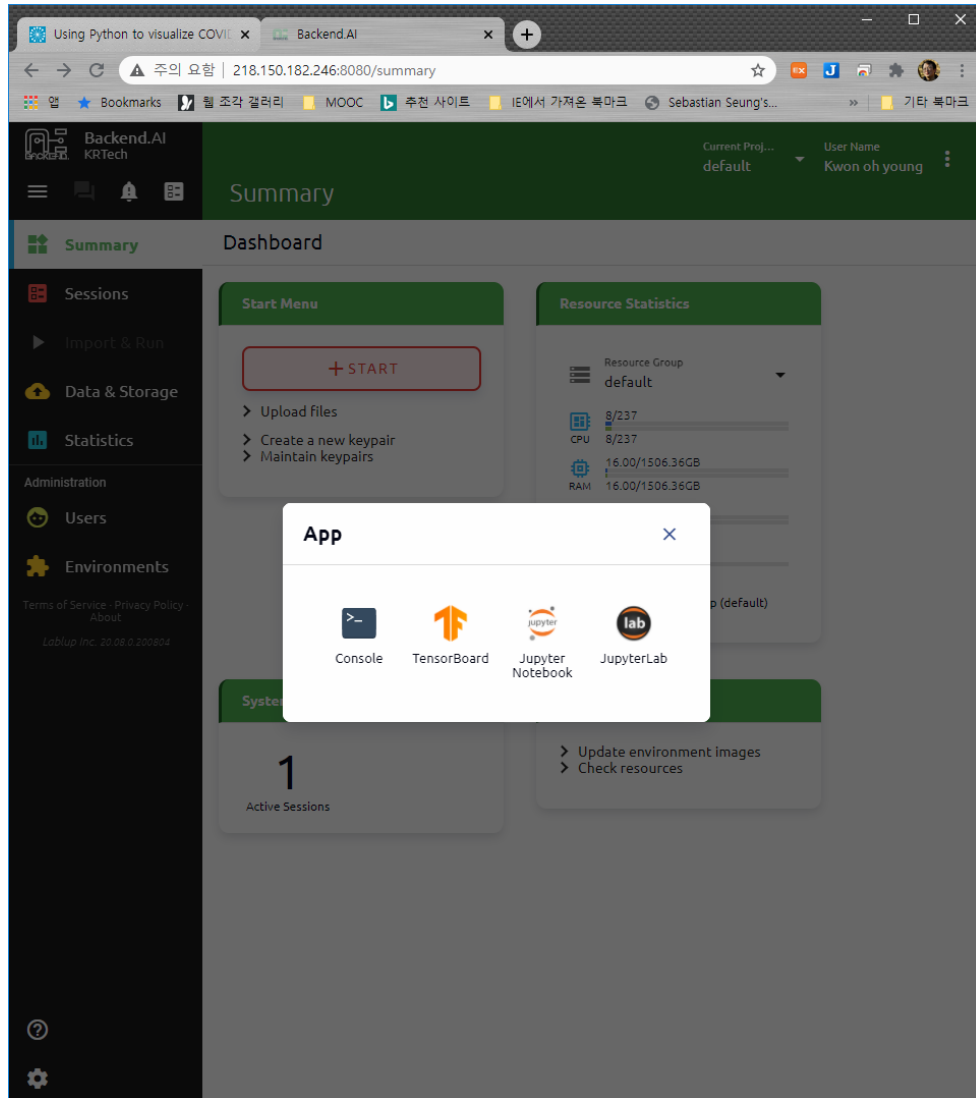
GPU 0.00 GPU

Sessions 1 #

Set session owner

LAUNCH

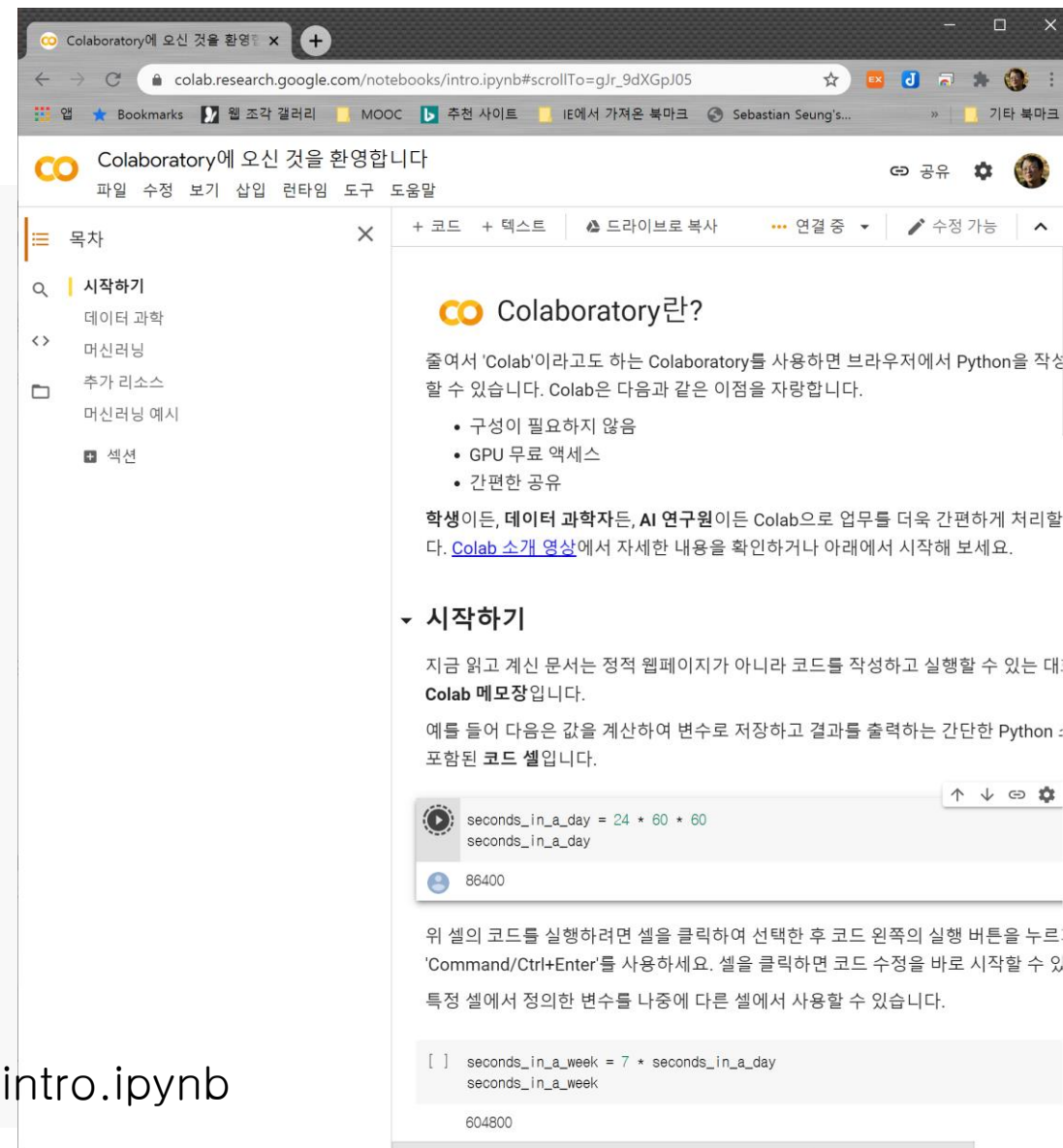
클라우드 활용



클라우드 활용

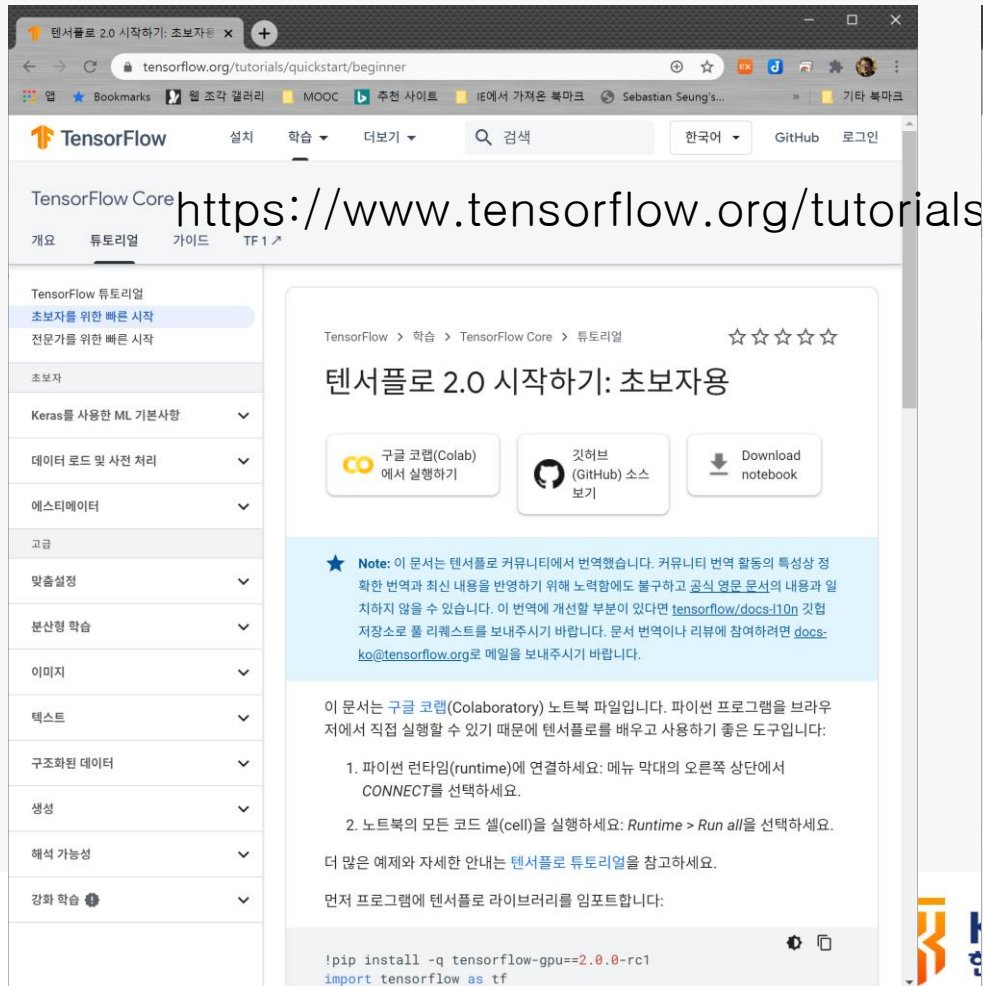
- 구글 colab
 - 구글이 제공
 - Jupyter notebook 확장
 - GPU/TPU 무료 액세스

<https://colab.research.google.com/notebooks/intro.ipynb>



클라우드 활용

• 인공지능활용(Tensorflow)



https://www.tensorflow.org/tutorials

TensorFlow Core

개요 튜토리얼 가이드 TF 1

TensorFlow 튜토리얼

초보자를 위한 빠른 시작

전문가를 위한 빠른 시작

초보자

Keras를 사용한 ML 기본사항

데이터 로드 및 사전 처리

에스티메이터

고급

맞춤설정

분산형 학습

이미지

텍스트

구조화된 데이터

생성

해석 가능성

강화 학습

TensorFlow > 학습 > TensorFlow Core > 튜토리얼

☆☆☆☆☆

텐서플로 2.0 시작하기: 초보자용

구글 코랩(Colab)에서 실행하기

깃허브(GitHub) 소스 보기

Download notebook

Note: 이 문서는 텐서플로 커뮤니티에서 번역했습니다. 커뮤니티 번역 활동의 특성상 정확한 번역과 최신 내용을 반영하기 위해 노력함에도 불구하고 공식 영문 문서의 내용과 일치하지 않을 수 있습니다. 이 번역에 개선할 부분이 있다면 [tensorflow/docs-110n](https://www.tensorflow.org/docs-110n) 깃헙 저장소로 풀 리퀘스트를 보내주시기 바랍니다. 문서 번역이나 리뷰에 참여하려면 docs-ko@tensorflow.org로 메일을 보내주시기 바랍니다.

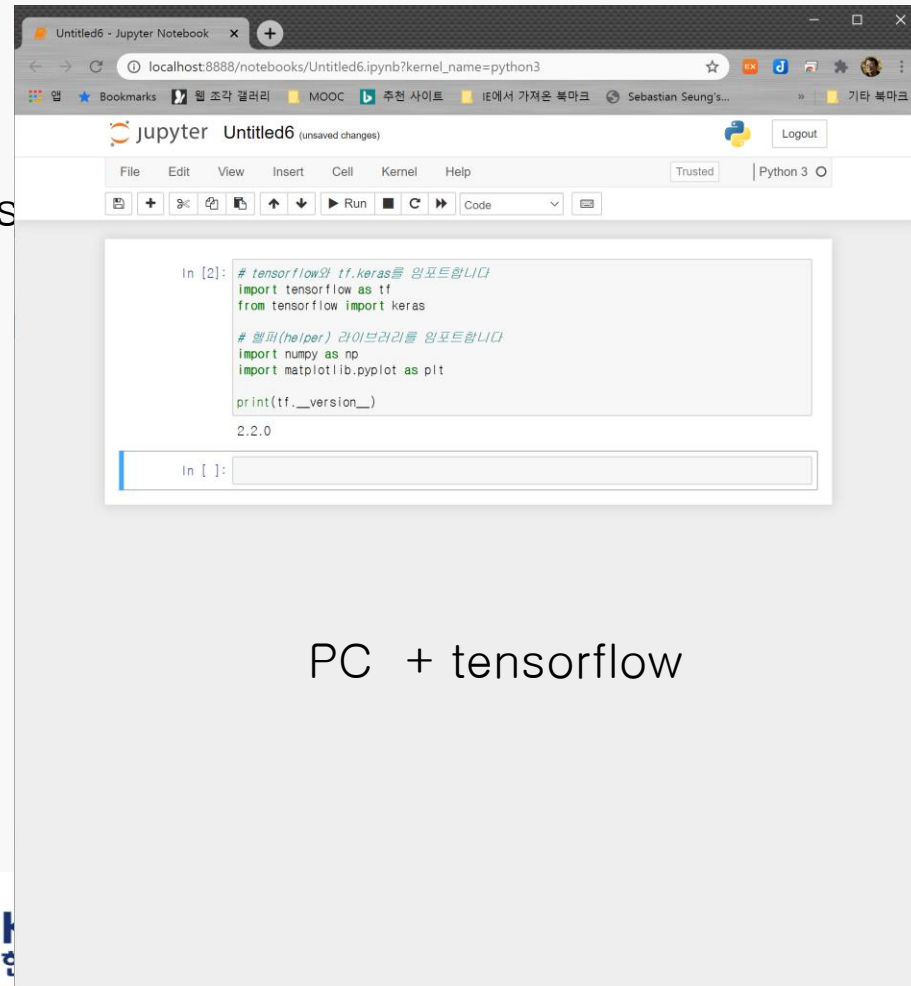
이 문서는 구글 코랩(Colaboratory) 노트북 파일입니다. 파이썬 프로그램을 브라우저에서 직접 실행할 수 있기 때문에 텐서플로를 배우고 사용하기 좋은 도구입니다:

1. 파이썬 런타임(runtime)에 연결하세요. 메뉴 막대의 오른쪽 상단에서 **CONNECT**를 선택하세요.
2. 노트북의 모든 코드 셀(cell)을 실행하세요: **Runtime > Run all**을 선택하세요.

더 많은 예제와 자세한 안내는 [텐서플로 튜토리얼](#)을 참고하세요.

먼저 프로그램에 텐서플로 라이브러리를 임포트합니다:

```
!pip install -q tensorflow-gpu==2.0.0-rc1
import tensorflow as tf
```



Untitled6 - Jupyter Notebook

localhost:8888/notebooks/Untitled6.ipynb?kernel_name=python3

Jupyter Untitled6 (unsaved changes)

File Edit View Insert Cell Kernel Help Trusted Python 3

In [2]:

```
# tensorflow와 tf.keras를 임포트합니다
import tensorflow as tf
from tensorflow import keras

# 헬퍼(helper) 라이브러리를 임포트합니다
import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__)
```

2.2.0

In []:

PC + tensorflow

Computational Thinking: A beginner's guide to problem-solving
and programming
written by Karl Beecher

컴퓨팅사고 정리

컴퓨팅사고란? (What is computational thinking?)

- ❖ 컴퓨터를 활용하여 문제를 해결하는 방안
- ❖ 유사한 개념인 Procedural Thinking (Papert, 1980): 컴퓨터를 사용해서 문제를 해결하는 방법으로 주어진 문제에 대하여 컴퓨터가 수행할 수 있는 알고리즘적인 해결책을 찾는 방안
- ❖ CT의 핵심개념(core concept)
 - logical thinking, algorithmic thinking, decomposition, generalisation and pattern recognition, modelling, abstraction, evaluation
- ❖ CT의 주변개념(peripheral concept)
 - data representation, critical thinking, computer science, automation, simulation/visualisation

컴퓨팅사고란? (What is computational thinking?)

❖ 컴퓨팅사고의 사용 예

- Pipelining a graduation ceremony
- Predicting climate change
- Sorting music charts (Sorting exam score)
- Assisting police, lawyers and judges

❖ 컴퓨팅사고 ≠ Computer Science

- 컴퓨팅사고는 문제해결의 하나의 접근법으로 컴퓨터 과학의 원리와 사례들을 활용하여 컴퓨터에서 수행 가능한 해결책을 만드는 것이다. 단지 프로그래머를 위한 것이 아니다. 컴퓨팅 사고는 다양한 영역에 적용 가능하다.

컴퓨터 소프트웨어

컴퓨터 소프트웨어

- ❖ Computer Software is a set of program instructions, including related data and documentation, that can be executed by computer.
- ❖ We need an artificial language to write a program.
- ❖ Each programming language has a set of primitive constructs, a syntax, a static semantics, and a semantics.
 - Primitive constructs ~ words
 - Syntax ~ 문장을 구성하는 방법 (문법) : cat dog boy.
 - Static semantics ~ 문장이 의미가 있는가를 정의:
I are big. (are → am)
 - Semantics ~ 문장이 가지고 있는 진정한 의미

컴퓨터 소프트웨어

❖ Python

- Primitive constructs ~ literals (number, string),
infix operators (+, /), etc.
- Syntax ~ $3.2 + 3.2$
- Static semantics ~ $3.2/'abc'$
(문법적으로는 맞지만 숫자를 문자로 나눌 수 없어 static semantics를 위반)
- Semantics ~ 올바른 프로그램은 한가지 의미만을 가짐
(자연어는 모호함이 있을 수 있다. 즉 동일한 문장이지만 상황에 따라 칭찬일 수도 있고 비난일 수도 있다.)

컴퓨터 소프트웨어

- ❖ Syntax error
 - 가장 흔하지만 그리 심각한 문제를 야기하지는 않음
 - 대부분의 언어는 문법오류가 발생한 지점을 잘 알려준다.
- ❖ Static semantic error
 - 배열의 인덱스 범위, 서로 다른 타입의 연산 등
 - Java: static semantic checking 을 많이 실행
 - C, Python: static semantic checking이 약함
- ❖ Semantic error: 심각한 오류 (프로그램 논리의 오류)

컴퓨터 소프트웨어

- ❖ Semantic error: 심각한 오류 (프로그램 논리의 오류)
 - Crash (메모리 참조 오류등으로 프로그램이 중단됨)
 - Never stop (종결조건의 오류등으로 무한루프에 빠지는 상황)
 - ✓ 프로그램 수행시 중간 결과값들을 출력하게 코딩 (개발시)
 - 수행을 마치고 결과를 생성
 - ✓ 결과가 올바를 수 도 있고, 틀릴 수 도 있음
- ❖ 디버깅(Debugging): 오류를 발견하고 수정하는 과정
- ❖ Software Testing: 소프트웨어가 기대하는 결과와 일치하는 실제 결과를 내놓지 점검하는 과정
 - 소프트웨어 개발 모든 과정에서 테스트는 필요
(TDD; Test Driven Development)
 - 결과가 불일치할 경우 디버깅 수행

컴퓨터 소프트웨어

❖ Programming Step (Problem solving step)

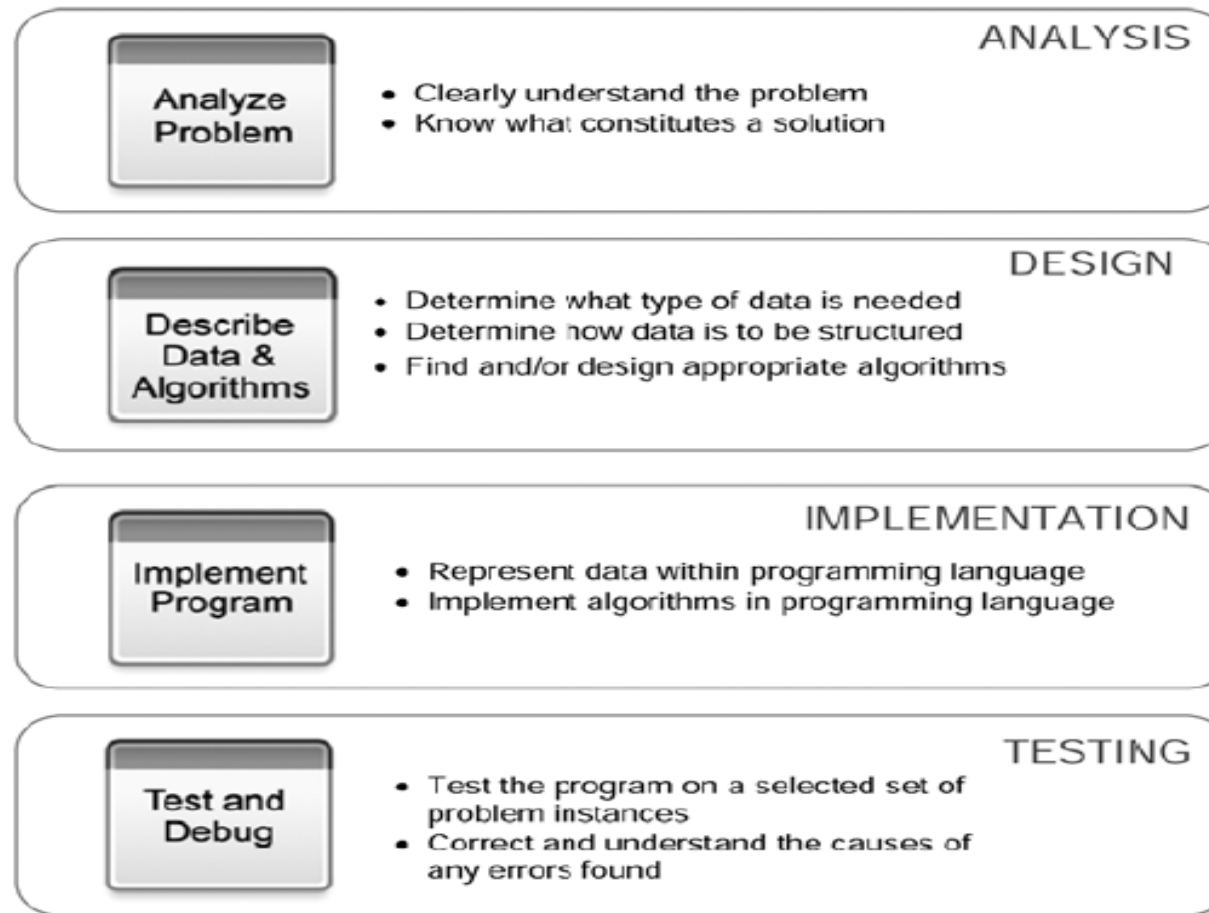


FIGURE 1-22 Process of Computational Problem Solving

디버깅

- ❖ Overt or covert:
 - overt bug: 명백한 징후가 있는 버그 (예, 프로그램이 붕괴되거나 예상보다 오래(아마 영원히) 수행)
 - covert bug: 명백한 징후가 없는 버그 (프로그램이 문제없이 결론에 도달한다. 하지만 잘못된 답변을 제공한다.)
- ❖ Persistent or intermittent:
 - persistent bug: 프로그램이 동일한 입력으로 실행될 때 마다, 매번 버그가 발생
 - intermittent bug: 프로그램이 동일한 입력, 동일한 조건하에서 수행될지라도 오류가 간헐적으로 발생
- ❖ (overt, persistent) : 쉽게 고칠 수 있다.
프로그램의 버그가 overt, persistent 가 되게 프로그래밍
→ defensive programming
- ❖ (overt, intermittent)
- ❖ (covert, persistent)
- ❖ (covert, intermittent) : 가장 수정이 어려운 버그

디버깅

- ❖ print -> 가장 단순하고, 가장 중요한 디버깅 도구
 - 의심되는 곳 전후에 프린트 문을 삽입
- ❖ 디버깅은 바람직하지 않은 동작(undesirable behavior)에 대한 설명을 찾는 과정이다.
 - Start(시작): 사용 가능한 데이터를 살펴봄 → 시험결과(test result), 프로그램 코드
 - Next(다음): 모든 데이터와 일치한다고 생각하는 가설을 세움
(가설: while 문의 종결조건이 잘못되었다.)
 - Next(다음): 가설을 반박할 가능성이 있는 반복 가능한 실험을 설계하고 수행
(반박실험: while loop의 앞뒤에 print 문 삽입해서 while 문의 종결조건을 살펴볼 수 있다.)
 - Finally(마지막으로): 어떤 실험을 시도했는지 기록
(기록을 하지 않으면, 동일한 버그를 처리하는 과정을 반복할 수 있다.)

디버깅 힌트

- ❖ *Look for the usual suspects.* E.g., have you (일반적인 용의자를 찾아라. 당신은)
 - 잘못된 순서로 함수에 인수를 전달했는가?
 - 이름을 잘못 입력했는가? (대문자대신 소문자를 입력)
 - 변수를 다시 초기화하는데 실패했는가?
 - 두 실수(부동소수점)의 값을 비교할 때 거의 같음 대신 같음으로 비교했는가?
(실수연산은 학교에서 배웠던 연산과 동일하지 않다. -> 왜?)
 - 두 객체의 동일성을 비교했을 때 값의 동일성을 비교했는가?
(값이 같음을 비교할 수 도 있고, 구조가 같음을 비교할 수 도 있다. 상황에 따라 올바른 비교를 했는지 검증할 필요가 있다.)
 - 일부 내장(built-in) 함수가 부작용을 일으킨다는 것을 잊었는가?
 - 함수를 호출할 때 함수 이름만 쓰고 ()를 잊었는가?
 - 의도하지 않은 별칭을 만들었는가?
 - 당신이 일반적으로 만드는 다른 실수를 하지 않았는가?

디버깅 힌트

- ❖ 왜 프로그램이 원하는 작업을 수행하지 않는지 묻지 말고, 대신 왜 그것이 무엇을 하고 있는지 물어보자. (*Stop asking yourself why the program isn't doing what you want it to. Instead, ask yourself why it is doing what it is.*)
대답하기 쉬운 질문이고, 프로그램을 고치는 방법을 알아내는 첫 단계가 됨
- ❖ 버그는 당신이 생각하는 곳에 있지 않을 수 있다는 것을 명심하자. (*Keep in mind that the bug is probably not where you think it is.*)
어디를 살펴 보아야 할지 결정하는 실질적인 방법 중 하나는 버그가 어디에 있는지 묻는 것임.
(다른 모든 요소를 제거하라. 그러면 남는 한가지가 진실이다.-셜록홈즈)
- ❖ 다른 사람에게 문제를 설명해보라. (*Try to explain the problem to somebody else.*)
종종 누군가에게 문제를 설명하려고 하면 놓친 것들을 보게 된다.

Debugging 힌트

- ❖ 읽은 모든 것을 믿지 마라. (*Don't believe everything you read.*)
코드가 문서의 주석(코멘트)이 제시하는 것을 수행하지 않을 수 있다.
- ❖ 디버깅을 멈추고 문서 작성을 시작하라. (*Stop debugging and start writing documentation.*)
다른 관점에서 문제에 접근하는데 도움을 준다.
- ❖ 물러나서 내일 다시 시도하자. (*Walk away, try again tomorrow.*)
버그를 나중에 수정한다는 의미일 수 있지만, 버그를 찾는데 소비되는 시간을 많이 줄일 수 있다. (집착하다보면 매몰되어 더 많은 시간을 소비하게 된다. 결국 프로그래밍 문제를 해결하기 위한 일을 일찍 시작하고, 시간을 갖고 문제를 해결해나가자.)
- ❖ 버그를 발견하면 바로 수정하지 말고, 이 버그가 관측된 대부분의 오류를 설명할 수 있는지, 수정시 다른 영향을 어떨지 잘 살펴보고, 계속 버그를 찾아서 가능하면 한번에 수정하는 것이 좋다.