

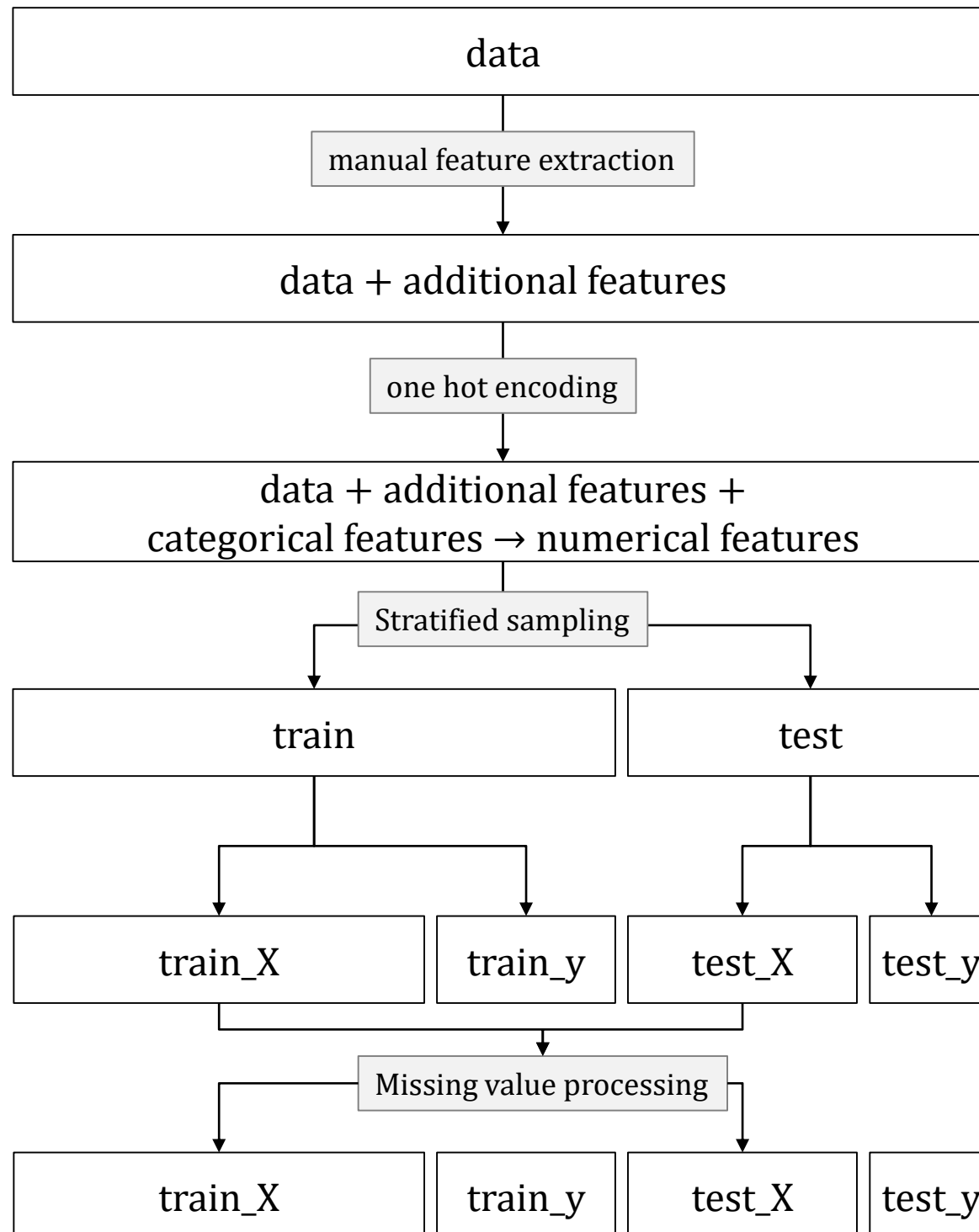
데이터사이언스응용 (Capstone design)

김응희

ehkim@sunmoon.ac.kr

Week 03

지난 주..



지난 주..



	No.	Action	Package/library
Data preprocessing for machine learning	0	Look at the big picture	—
	1	Get the data	tarfile, urllib, pandas
	2	Discover and visualize the data to gain insights	pandas, matplotlib
	3	Prepare the data for Machine Learning algorithms	pandas, scikit-learn, numpy
	4	Select a model and train it	
	5	Fine-tune your model	
	6	Present your solution	
	7	Launch, monitor and maintain your system	joblib, flask

End-to-End Machine Learning Project

No.	Action	Package/library
0	Look at the big picture	—
1	Get the data	tarfile, urllib, pandas
2	Discover and visualize the data to gain insights	pandas, matplotlib
3	Prepare the data for Machine Learning algorithms	pandas, scikit-learn, numpy
4	Select a model and train it	
5	Fine-tune your model	
6	Present your solution	
7	Launch, monitor and maintain your system	joblib, flask

Three ML methods we gonna use today..

**Linear
regression**

**Decision tree
regression**

**Random forest
regression**

4.1 Training and Evaluating on the Training Set

Linear
regression

```
from sklearn.linear_model import LinearRegression

linear = LinearRegression()
linear.fit(train_X, train_y)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
from sklearn.metrics import mean_squared_error
import numpy as np

predictions = linear.predict(train_X)
mse = mean_squared_error(train_y, predictions)
rmse = np.sqrt(mse)
answer_mean = train_y["median_house_value"].mean()
print(str(rmse/answer_mean*100) + "%")
```

```
32.92999142716399%
```

4.1 Training and Evaluating on the Training Set

Decision tree
regression

```
from sklearn.tree import DecisionTreeRegressor
```

```
tree = DecisionTreeRegressor()  
tree.fit(train_X, train_y)
```

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=None,  
                      max_features=None, max_leaf_nodes=None,  
                      min_impurity_decrease=0.0, min_impurity_split=None,  
                      min_samples_leaf=1, min_samples_split=2,  
                      min_weight_fraction_leaf=0.0, pre_sort='deprecated',  
                      random_state=None, splitter='best')
```

```
predictions = tree.predict(train_X)  
mse = mean_squared_error(train_y, predictions)  
rmse = np.sqrt(mse)  
answer_mean = train_y["median_house_value"].mean()  
print(str(rmse/answer_mean*100) + "%")
```

0.0%

4.1 Training and Evaluating on the Training Set

**Linear
regression**

Under fitting

**Decision tree
regression**

Over fitting

**Random forest
regression**

4.2 Evaluating on the Test Set

```
predictions = linear.predict(test_X)
mse = mean_squared_error(test_y, predictions)
rmse = np.sqrt(mse)
answer_mean = test_y["median_house_value"].mean()
print(str(rmse/answer_mean*100) + "%")
```

33.387643759619735%

```
predictions = tree.predict(test_X)
mse = mean_squared_error(test_y, predictions)
rmse = np.sqrt(mse)
answer_mean = test_y["median_house_value"].mean()
print(str(rmse/answer_mean*100) + "%")
```

33.33372756965135%

4.3 Evaluating on the Training and Test Set using Random forest regression

Linear
regression

Decision tree
regression

Random forest
regression

4.3 Evaluating on the Training and Test Set using Random forest regression

```
from sklearn.ensemble import RandomForestRegressor
```

```
forest = RandomForestRegressor(n_estimators=5, random_state=0)  
forest.fit(train_X, train_y["median_house_value"].ravel())
```

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',  
                        max_depth=None, max_features='auto', max_leaf_node  
s=None,  
                        max_samples=None, min_impurity_decrease=0.0,  
                        min_impurity_split=None, min_samples_leaf=1,  
                        min_samples_split=2, min_weight_fraction_leaf=0.0,  
                        n_estimators=5, n_jobs=None, oob_score=False,  
                        random_state=0, verbose=0, warm_start=False)
```

4.3 Evaluating on the Training and Test Set using Random forest regression

```
predictions = forest.predict(train_X)
mse = mean_squared_error(train_y, predictions)
rmse = np.sqrt(mse)
answer_mean = train_y["median_house_value"].mean()
print(str(rmse/answer_mean*100) + "%")
```

12.12961140719765%

```
predictions = forest.predict(test_X)
mse = mean_squared_error(test_y, predictions)
rmse = np.sqrt(mse)
answer_mean = test_y["median_house_value"].mean()
print(str(rmse/answer_mean*100) + "%")
```

26.17531878445498%

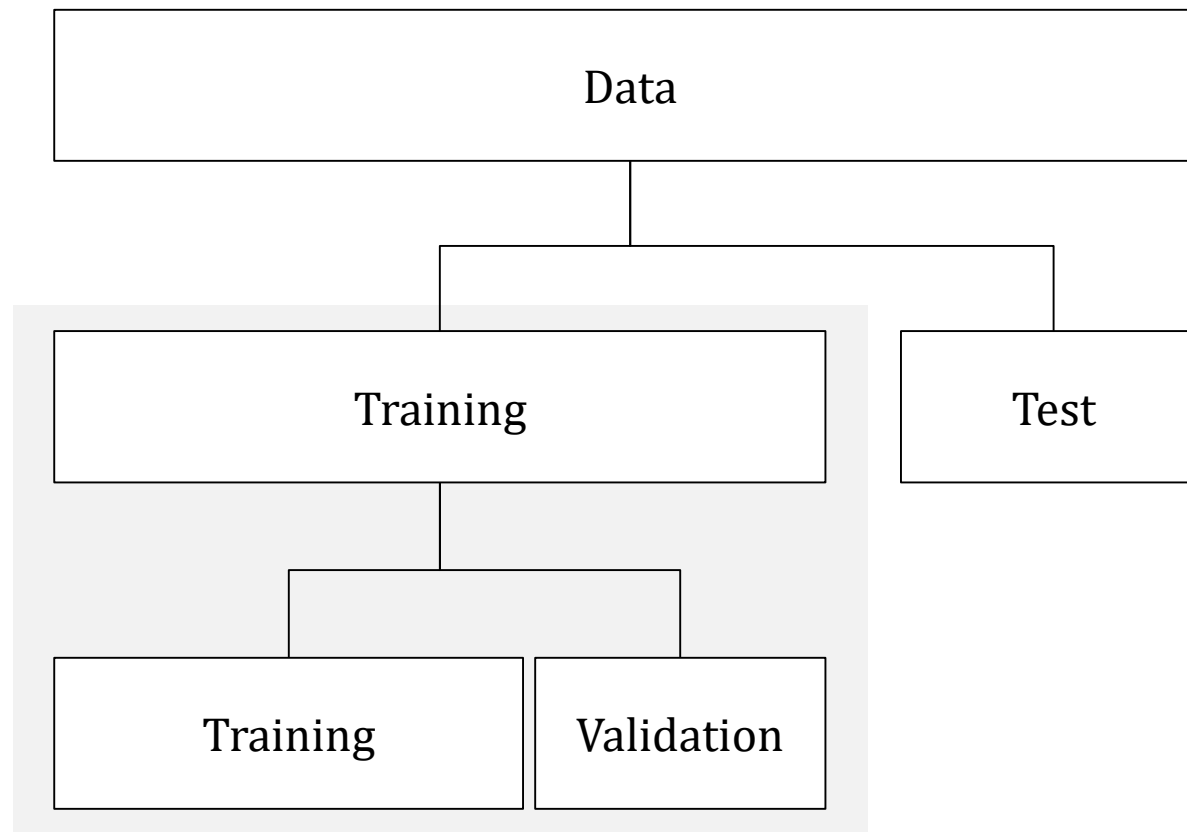
Comparison

	Linear regression	Decision tree regression	Random forest regression
RMSE on training set	32.9%	0%	12.1%
RMSE on test set	33.3%	33.3%	26.1%

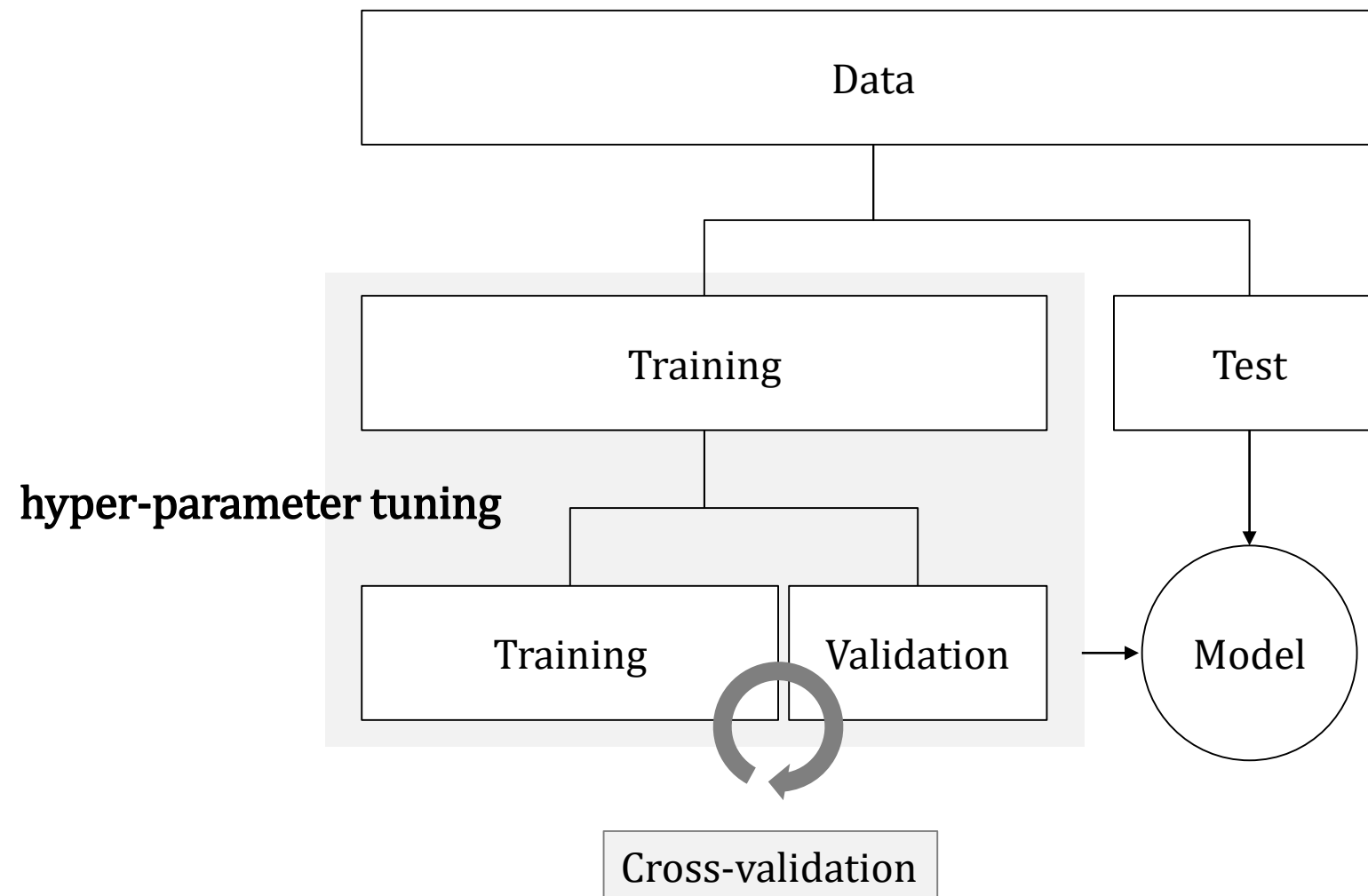
End-to-End Machine Learning Project

No.	Action	Package/library
0	Look at the big picture	—
1	Get the data	tarfile, urllib, pandas
2	Discover and visualize the data to gain insights	pandas, matplotlib
3	Prepare the data for Machine Learning algorithms	pandas, scikit-learn, numpy
4	Select a model and train it	
5	Fine-tune your model	
6	Present your solution	
7	Launch, monitor and maintain your system	joblib, flask

The ideal scenario



The ideal scenario



Grid Search with Cross Validation

```
from sklearn.model_selection import GridSearchCV

param = {"n_estimators": [5, 10, 30],
         "max_depth": [100, 200]}
forest = RandomForestRegressor(random_state=0)
search = GridSearchCV(forest, param,
                      cv=5, scoring="neg_mean_squared_error")
search.fit(train_X, train_y["median_house_value"].ravel())
```

Your fine-tuned model

```
search.best_params_
```

```
{'max_depth': 100, 'n_estimators': 30}
```

```
forest = RandomForestRegressor(max_depth=100,  
                              n_estimators=30,  
                              random_state=0)  
forest.fit(train_X, train_y["median_house_value"].ravel())
```

Your fine-tuned model

```
predictions = forest.predict(train_X)
mse = mean_squared_error(train_y, predictions)
rmse = np.sqrt(mse)
answer_mean = train_y["median_house_value"].mean()
print(str(rmse/answer_mean*100) + "%")
```

9.421054753574424%

```
predictions = forest.predict(test_X)
mse = mean_squared_error(test_y, predictions)
rmse = np.sqrt(mse)
answer_mean = test_y["median_house_value"].mean()
print(str(rmse/answer_mean*100) + "%")
```

23.886510037964545%

Your fine-tune model

	Linear regression	Decision tree regression	Random forest regression	Fine-tuned model
RMSE on training set	32.9%	0%	12.1%	9.4%
RMSE on test set	33.3%	33.3%	26.1%	23.8%

End-to-End Machine Learning Project

No.	Action	Package/library
0	Look at the big picture	—
1	Get the data	tarfile, urllib, pandas
2	Discover and visualize the data to gain insights	pandas, matplotlib
3	Prepare the data for Machine Learning algorithms	pandas, scikit-learn, numpy
4	Select a model and train it	
5	Fine-tune your model	
6	Present your solution	
7	Launch, monitor and maintain your system	joblib, flask

Feature importance

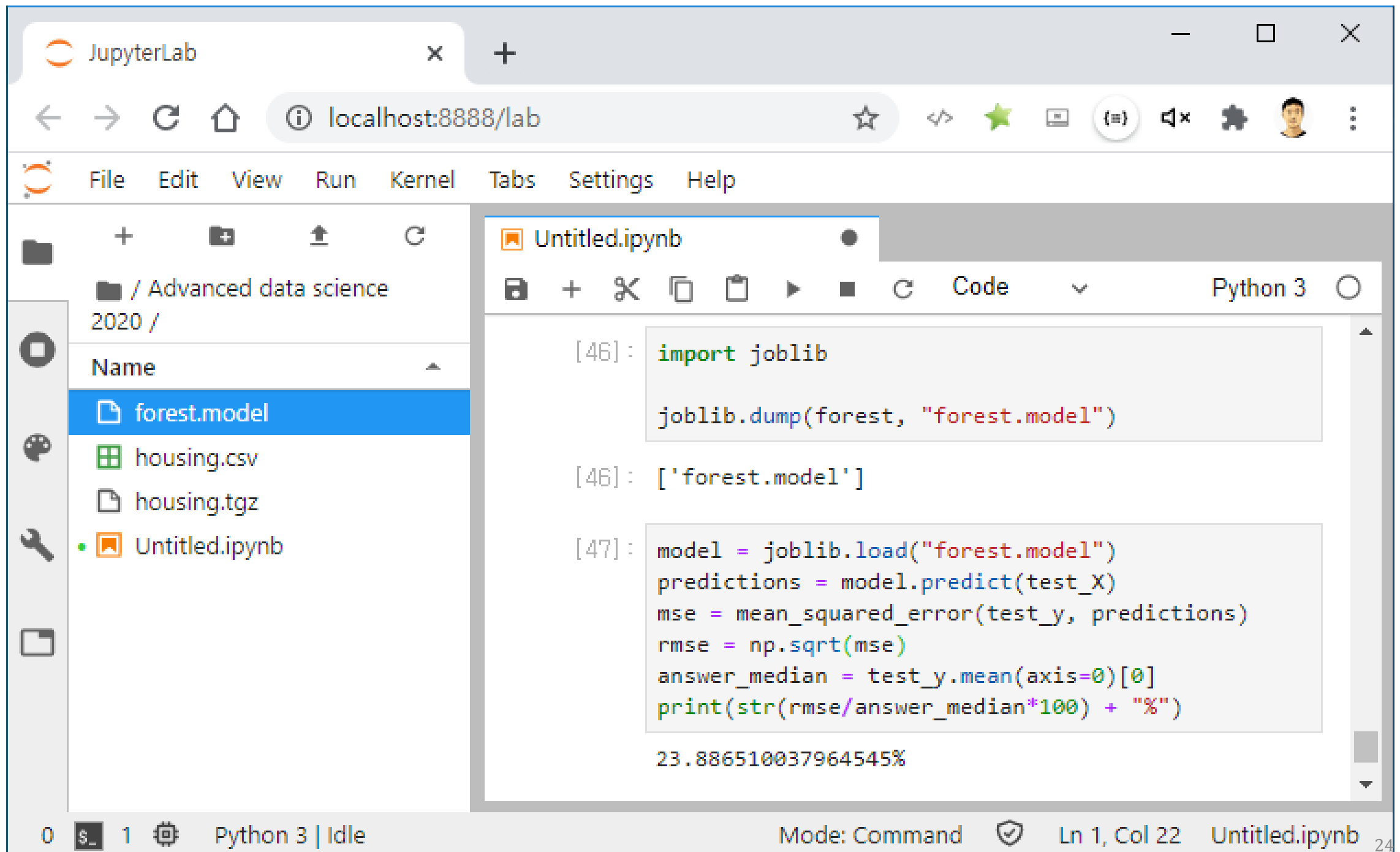
```
importances = forest.feature_importances_  
features = train_X.columns.values  
  
sorted(zip(importances, features), reverse=True)
```

```
[(0.47763306801959393, 'median_income'),  
 (0.14667781030892643, 'INLAND'),  
 (0.09675169156214641, 'longitude'),  
 (0.08728313888933595, 'latitude'),  
 (0.047588373717733196, 'housing_median_age'),  
 (0.03244868607116767, 'bedrooms_per_room'),  
 (0.028850969908527545, 'rooms_per_household'),  
 (0.024088508967163785, 'population'),  
 (0.01855807032696892, 'total_rooms'),  
 (0.015667964018780612, 'total_bedrooms'),  
 (0.01431359816365993, 'households'),  
 (0.006327717669091209, 'NEAR OCEAN'),  
 (0.0026947420610949524, '<1H OCEAN'),  
 (0.001037627384628986, 'NEAR BAY'),  
 (7.8032931180394e-05, 'ISLAND')]
```

End-to-End Machine Learning Project

No.	Action	Package/library
0	Look at the big picture	—
1	Get the data	tarfile, urllib, pandas
2	Discover and visualize the data to gain insights	pandas, matplotlib
3	Prepare the data for Machine Learning algorithms	pandas, scikit-learn, numpy
4	Select a model and train it	
5	Fine-tune your model	
6	Present your solution	
7	Launch, monitor and maintain your system	joblib, flask

Save & load your model



The screenshot displays the JupyterLab web interface in a browser window. The address bar shows `localhost:8888/lab`. The left sidebar contains a file explorer for the directory `/ Advanced data science 2020 /`, listing `forest.model`, `housing.csv`, `housing.tgz`, and `Untitled.ipynb`. The `forest.model` file is selected. The main area shows the `Untitled.ipynb` notebook with two code cells. The first cell (index 46) imports `joblib` and saves a `forest` object to `forest.model`. The second cell (index 47) loads the model, makes predictions on `test_X`, calculates the mean squared error (MSE), and prints the RMSE as a percentage of the median answer. The output of the second cell is `23.886510037964545%`. The bottom status bar indicates the current mode is 'Command' and the kernel is 'Python 3 | Idle'.

JupyterLab

localhost:8888/lab

File Edit View Run Kernel Tabs Settings Help

/ Advanced data science 2020 /

Name

- forest.model
- housing.csv
- housing.tgz
- Untitled.ipynb

Untitled.ipynb

Code Python 3

```
[46]: import joblib

      joblib.dump(forest, "forest.model")

[46]: ['forest.model']

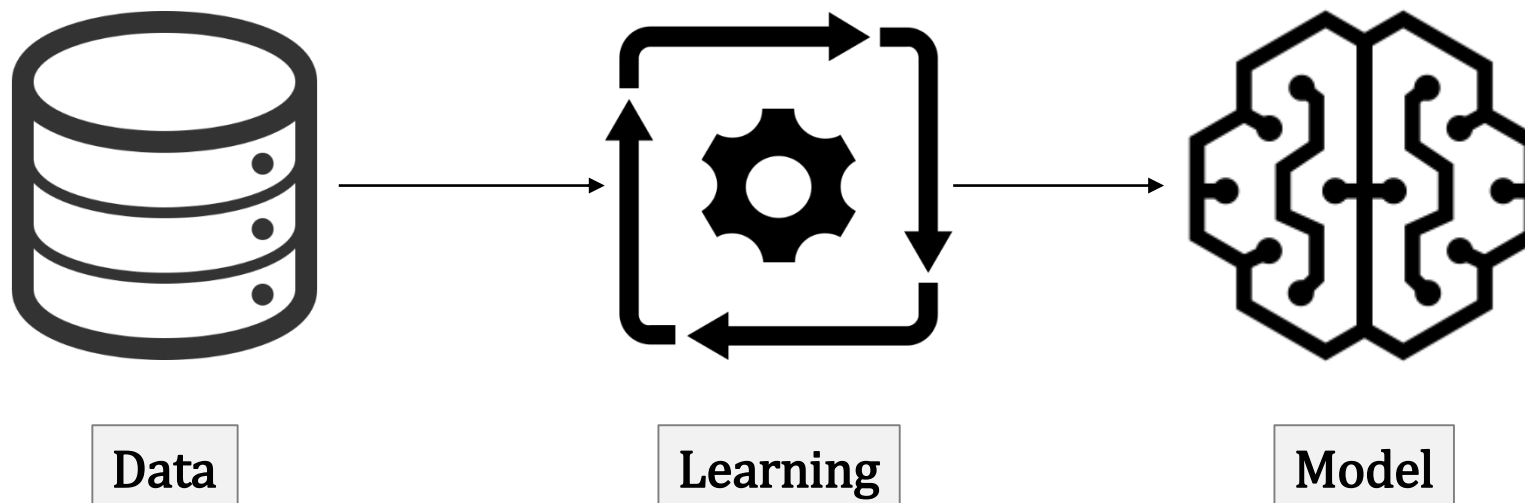
[47]: model = joblib.load("forest.model")
      predictions = model.predict(test_X)
      mse = mean_squared_error(test_y, predictions)
      rmse = np.sqrt(mse)
      answer_median = test_y.mean(axis=0)[0]
      print(str(rmse/answer_median*100) + "%")

      23.886510037964545%
```

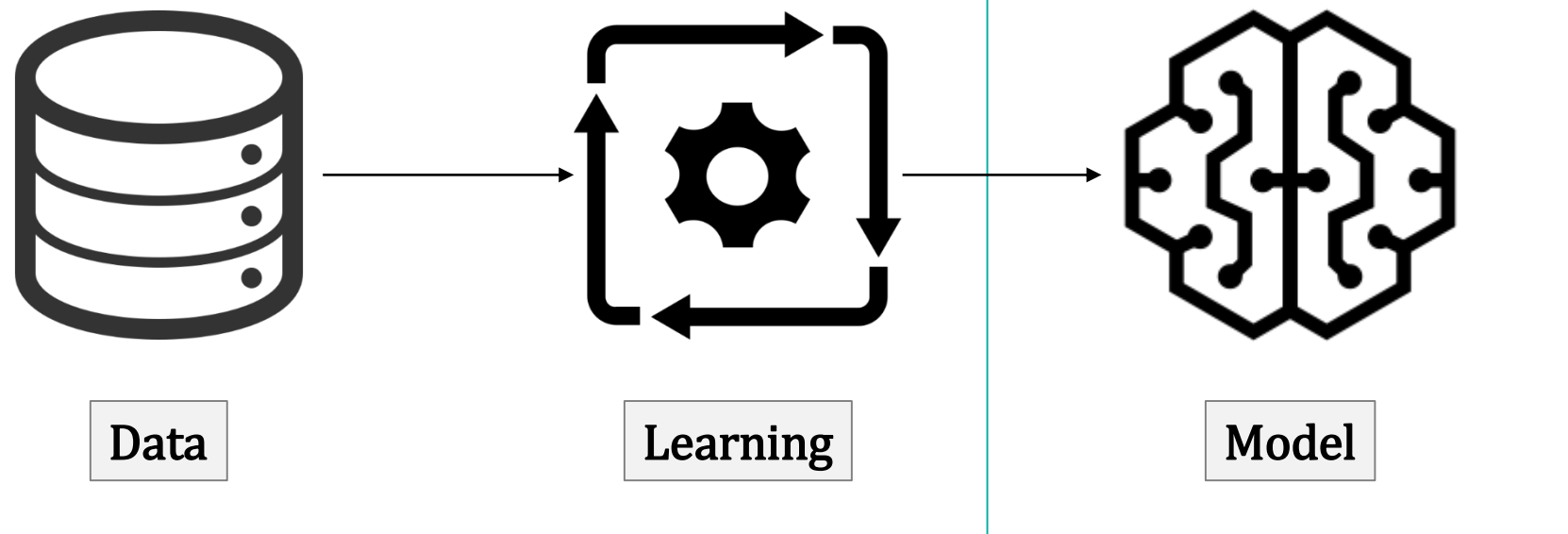
0 \$ 1 Python 3 | Idle Mode: Command Ln 1, Col 22 Untitled.ipynb 24

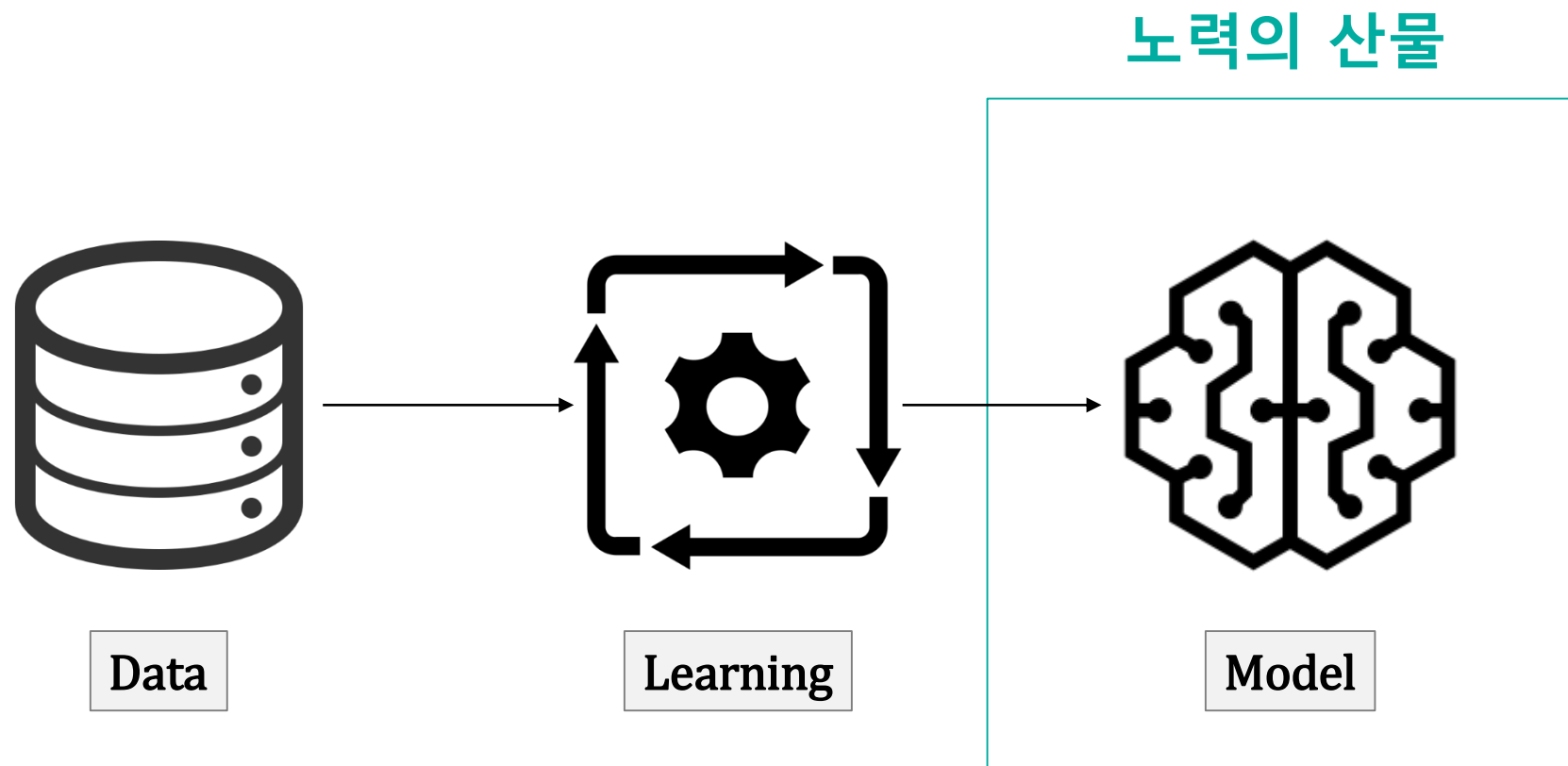
기계 학습 모델 구축 → 많은 자원 소모

- 데이터 수집 및 분석
- 적합한 기계 학습 방법론 선택
 - Linear regression, decision tree, random forest, perceptron, neural network, etc.
- 다양한 hyper parameter 설정
 - Num. of layers, activation function, learning rate, etc.



노력의 산물





프로그래밍 언어의 장벽

지적 재산권(보안)

APIs & REST APIs

Application Programming Interface

Application Programming Interface

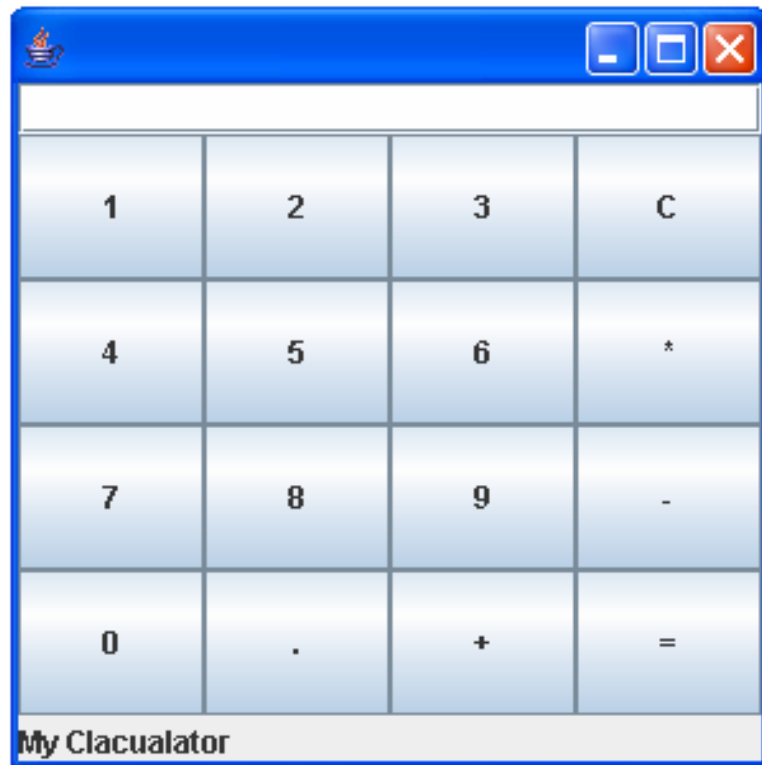
- A way to let **software components** to **talk to each other**



API 경험이 있나요?

API 경험이 있나요?

- Swing (Java)

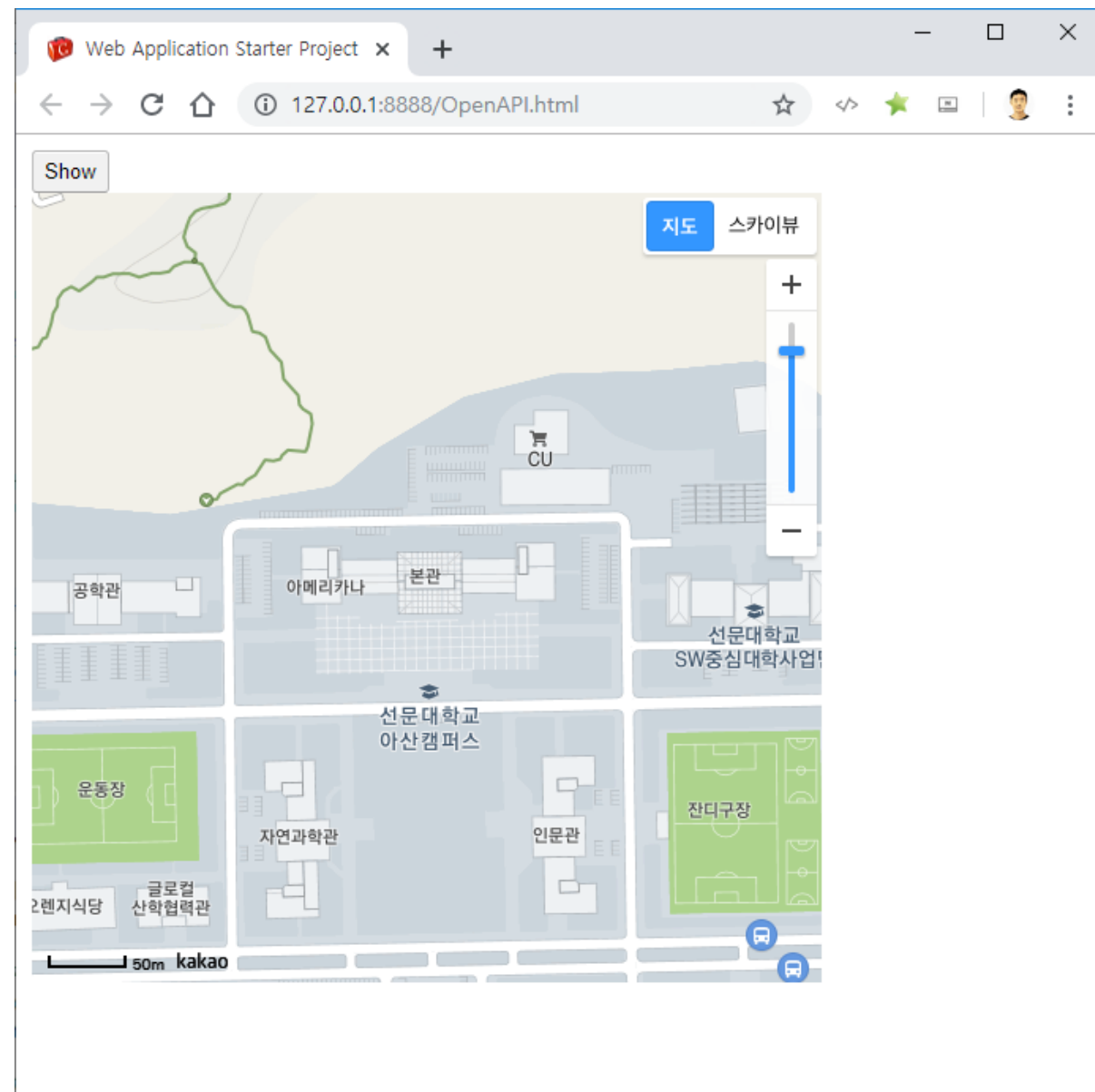
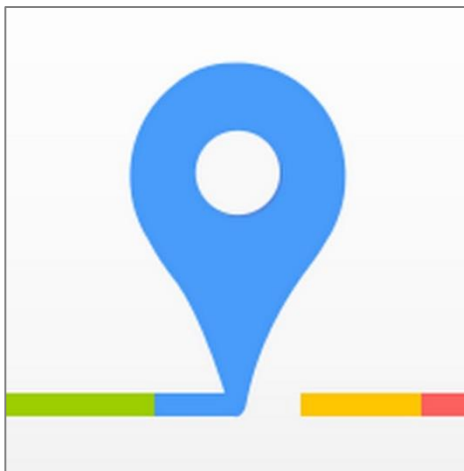


JFrame
+ EXIT_ON_CLOSE: int - rootPane: JRootPane ...
+ JFrame() ... + setTitle(String): void + setSize(int, int): void + setIconImage(Image): void + setVisible(boolean): void ...

```
public static void main(String args)
{
    JFrame frame = new JFrame();
    frame.setTitle("Cacluator");
    frame.setVisible(true);
}
```


API 경험이 있나요?

- Open API



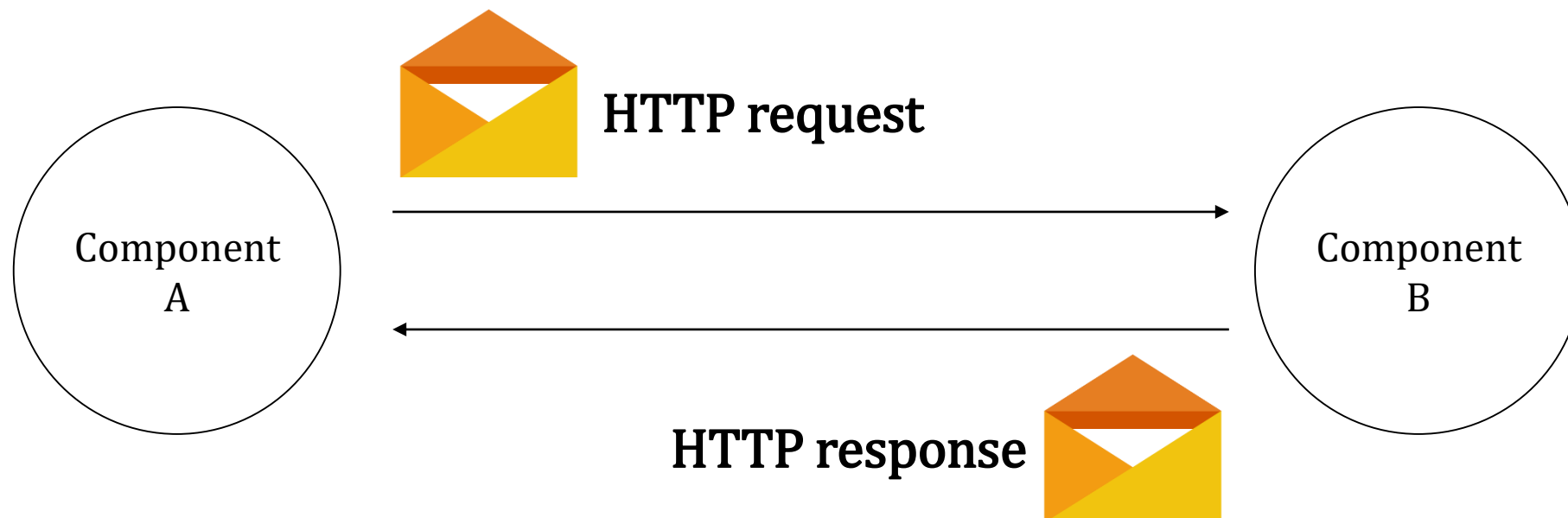
So an **API** could be anything in any form.

The only thing that it has to be is that

it has to be a **way to communicate with a software component.**

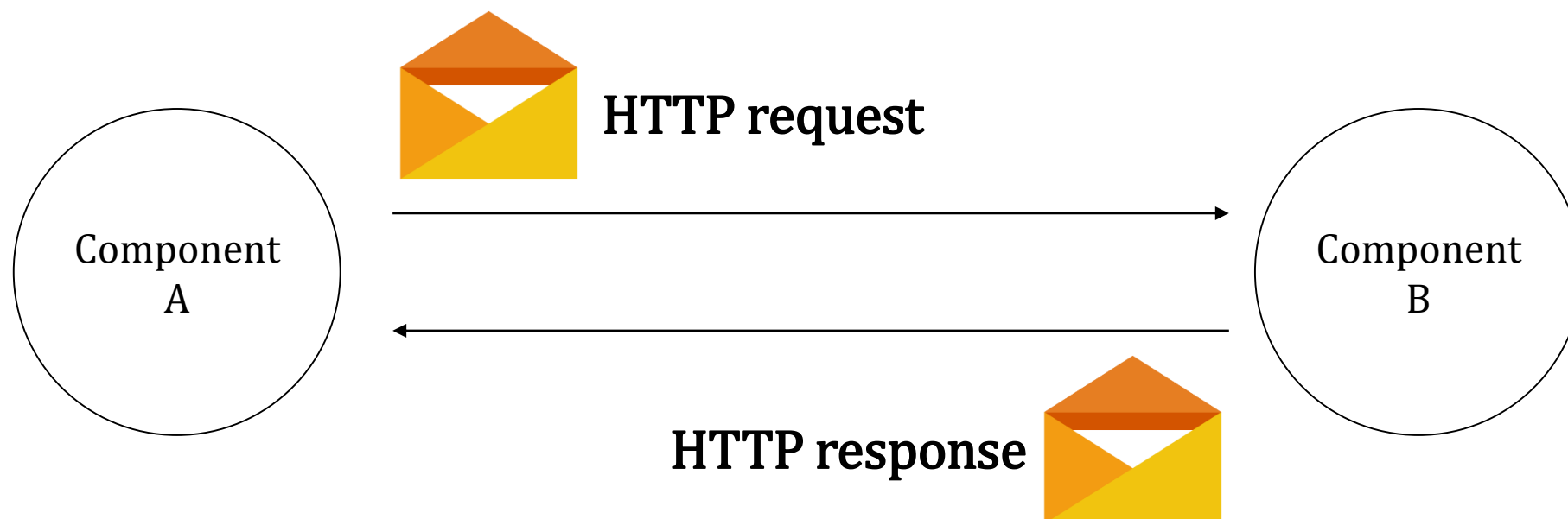
REST API

- HTTP 프로토콜 기반의 API



REST API

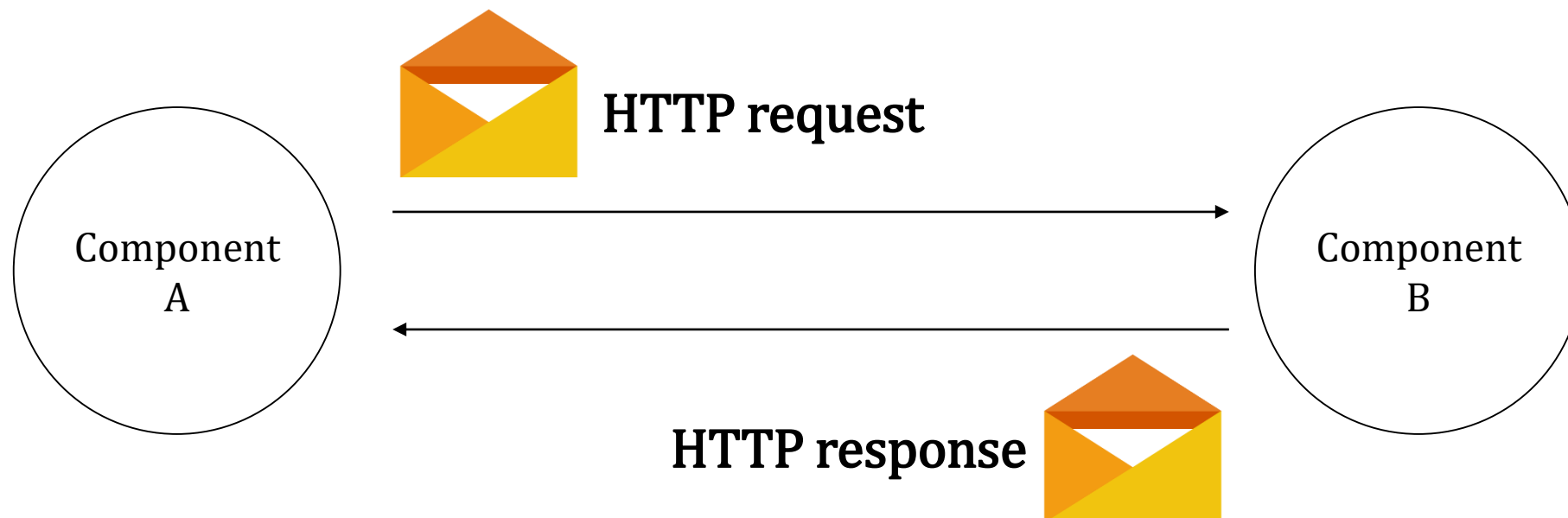
- HTTP 프로토콜 기반의 API



Client language	Server language
Python	Python
Python	Java
Java	Python
...	...

REST API

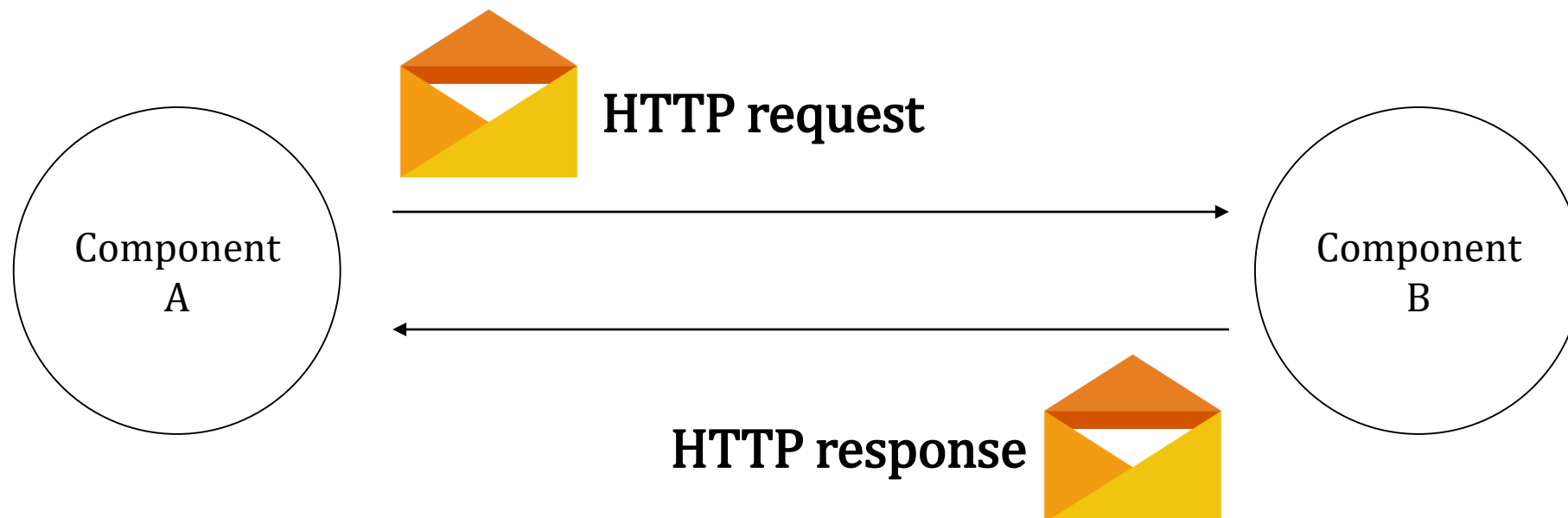
- HTTP 프로토콜 기반의 API



CRUD	HTTP verb
Create	POST
Read	GET
Update	PUT/PATCH
Delete	DELTE

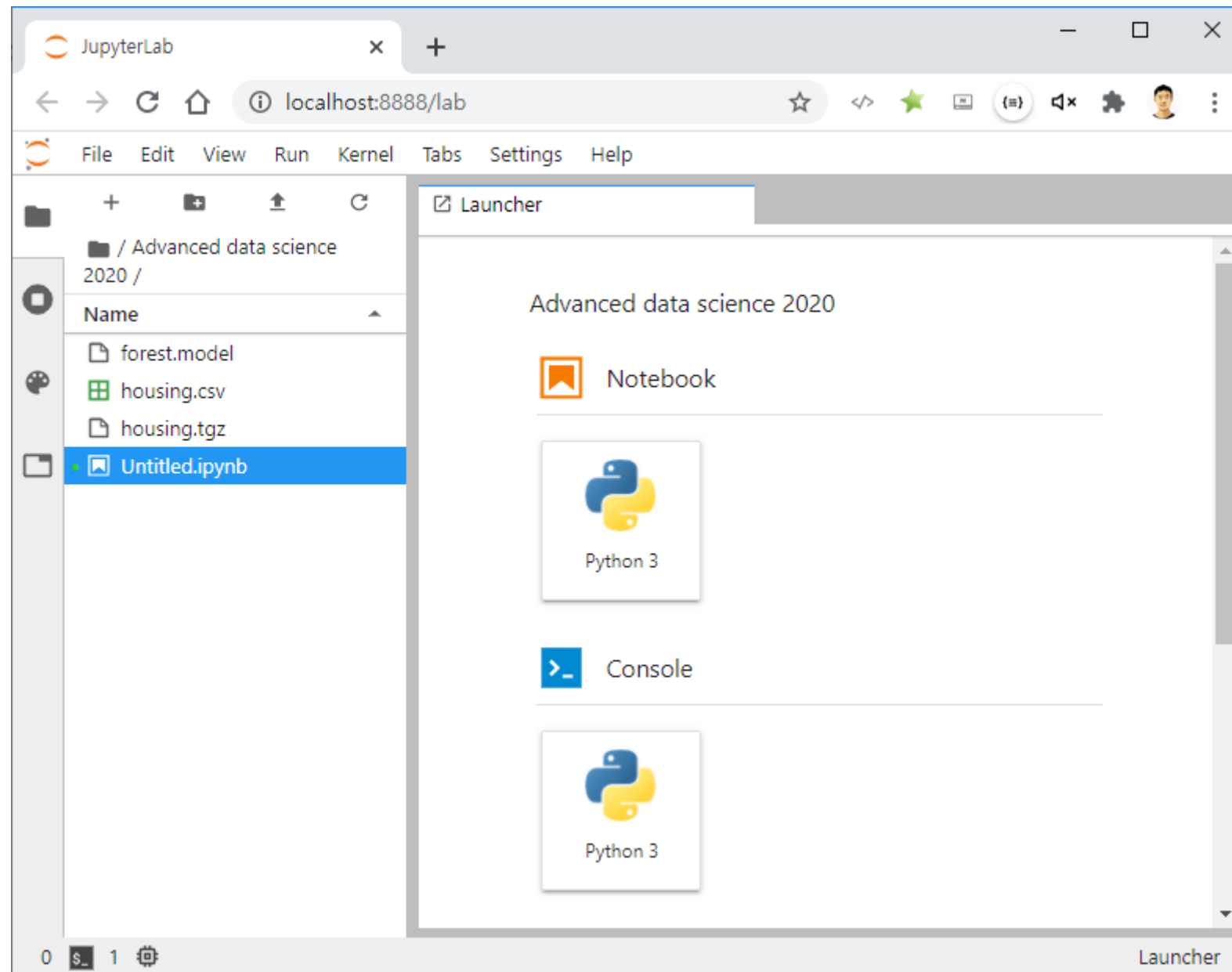
REST API

- HTTP 프로토콜 기반의 API

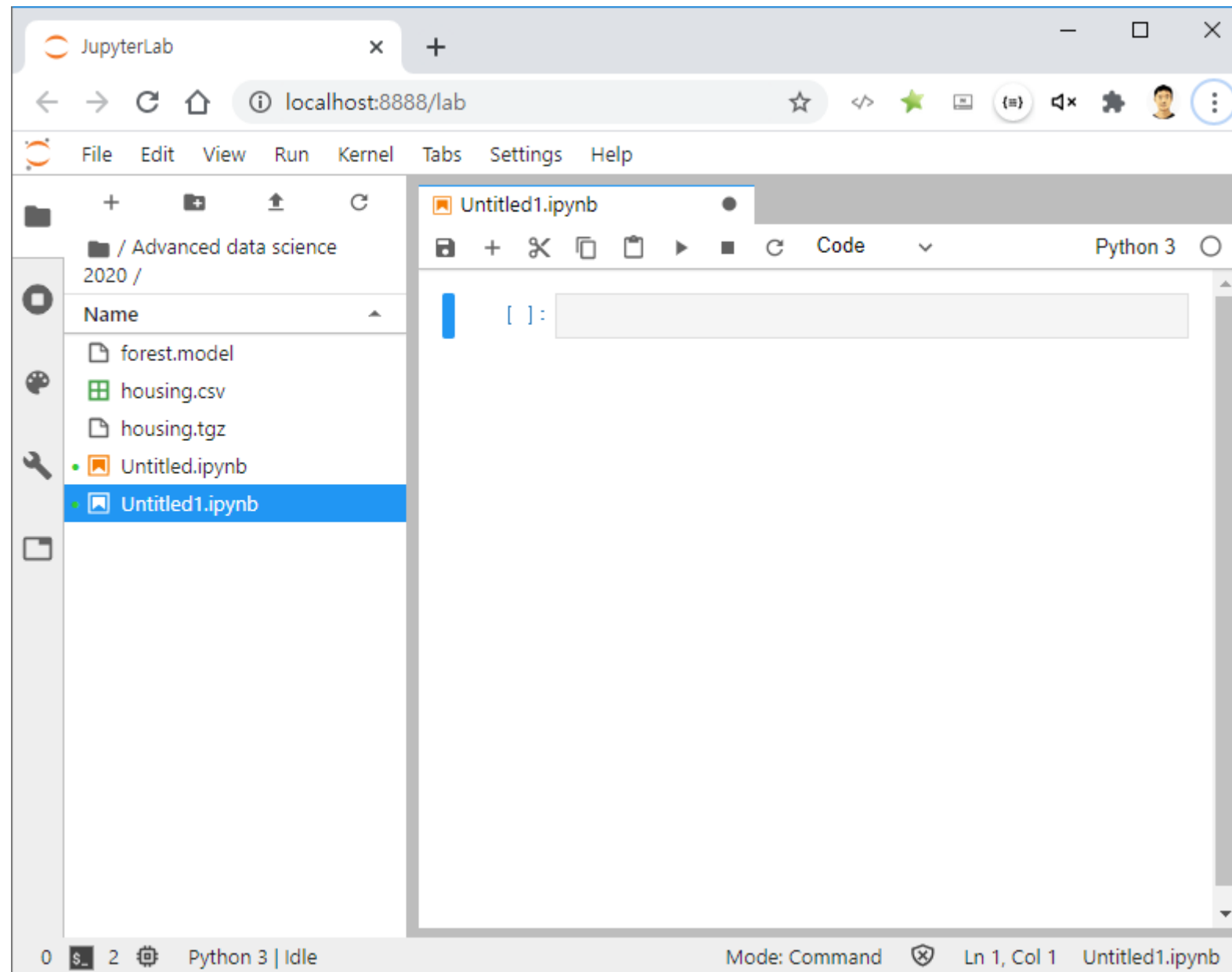


CRUD	HTTP verb
Create	POST
Read	GET
Update	PUT/PATCH
Delete	DELTE

New notebook



New notebook



Sample api

The screenshot shows the JupyterLab web interface in a browser window. The address bar displays `localhost:8888/lab`. The left sidebar shows a file explorer with the directory `/ Advanced data science 2020 /` containing files `forest.model`, `housing.csv`, `housing.tgz`, and two notebooks: `Untitled.ipynb` and `Untitled1.ipynb` (which is selected). The main area displays the `Untitled1.ipynb` notebook in 'Code' view, showing the following Python code:

```
[ ]: from flask import Flask

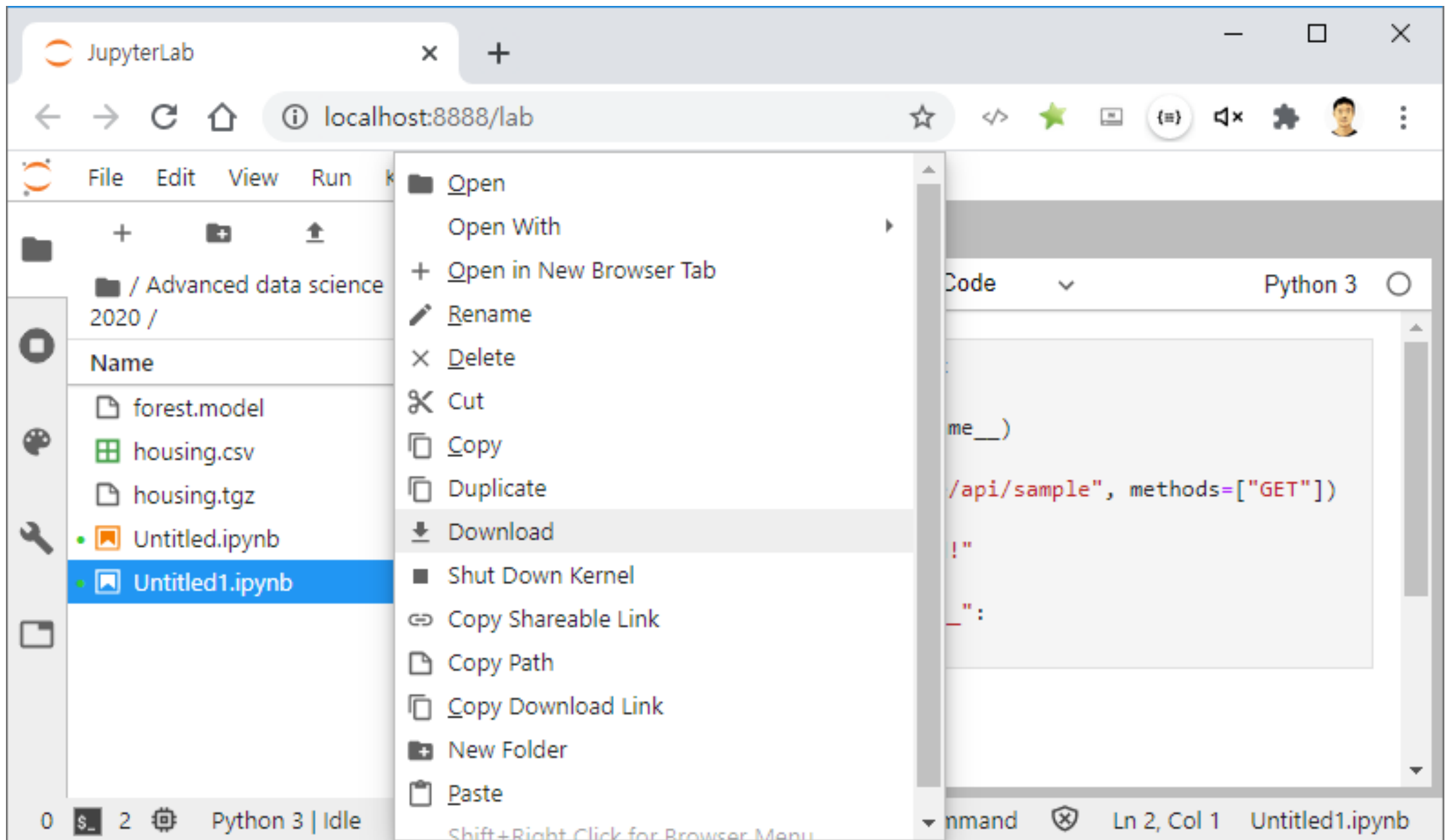
sample_api = Flask(__name__)

@sample_api.route("/gse/api/sample", methods=["GET"])
def hello_world():
    return "Hello World!"

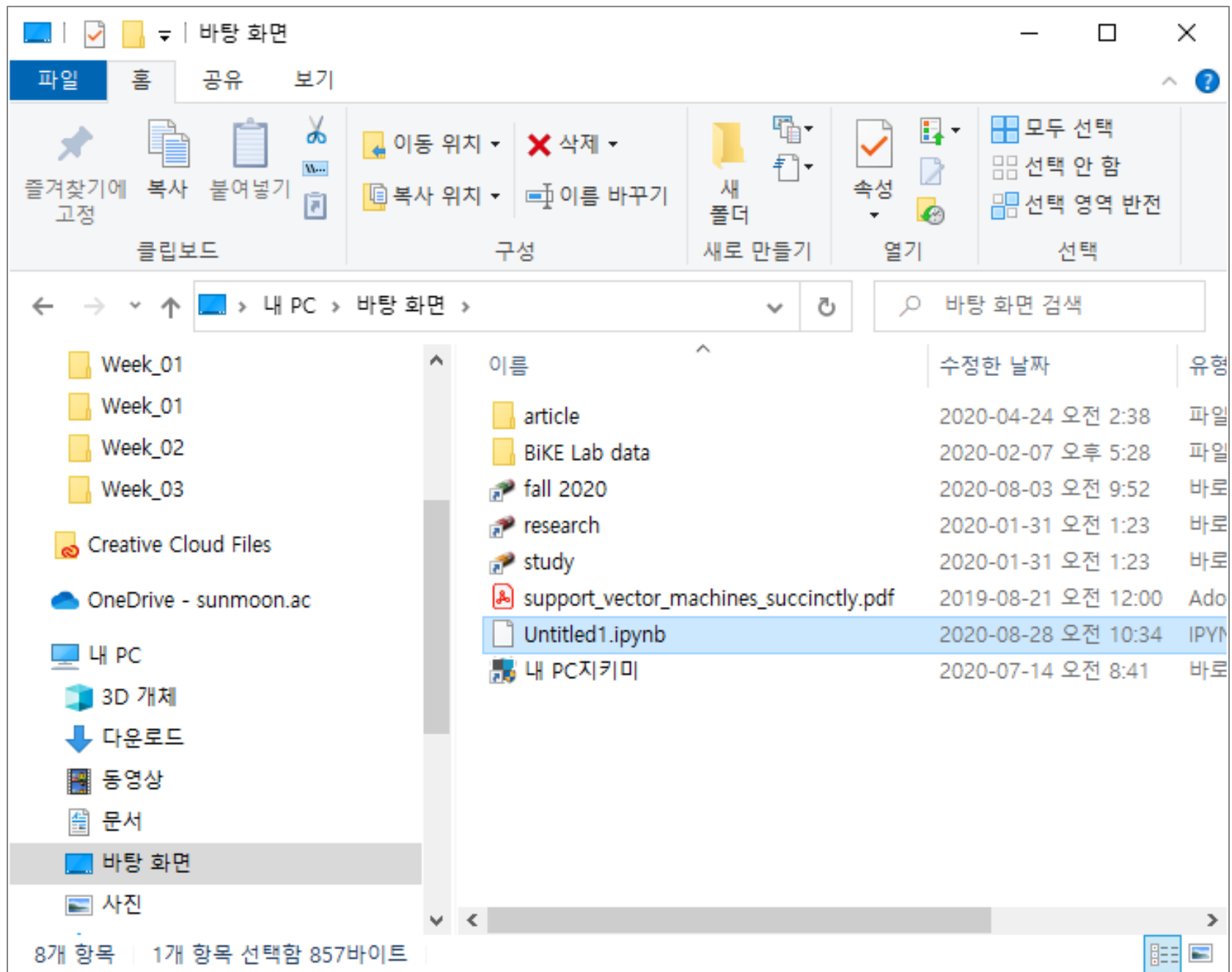
if __name__ == "__main__":
    sample_api.run()
```

The status bar at the bottom indicates the current cell is at line 2, column 1, in the `Untitled1.ipynb` notebook, using the Python 3 kernel.

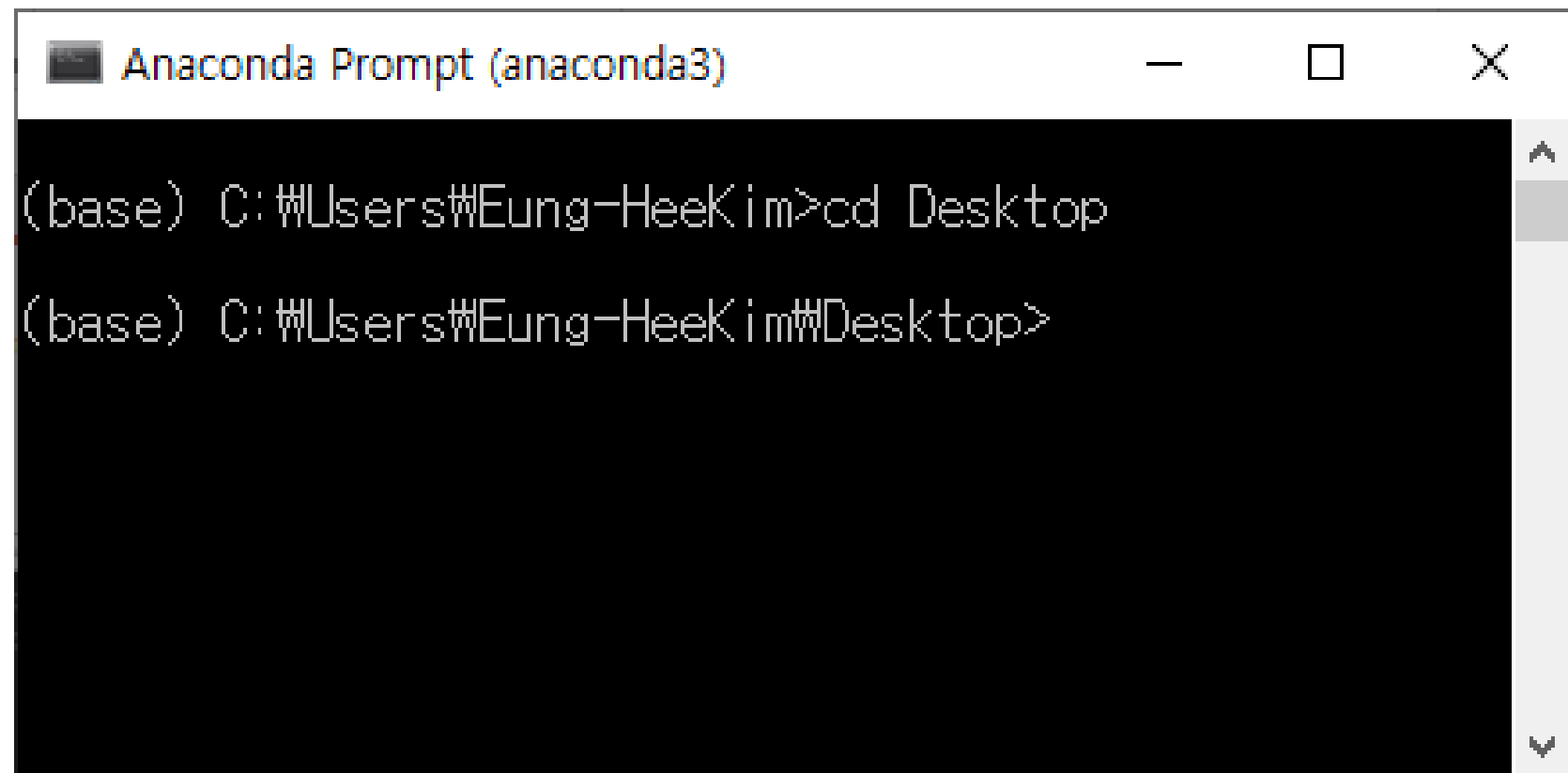
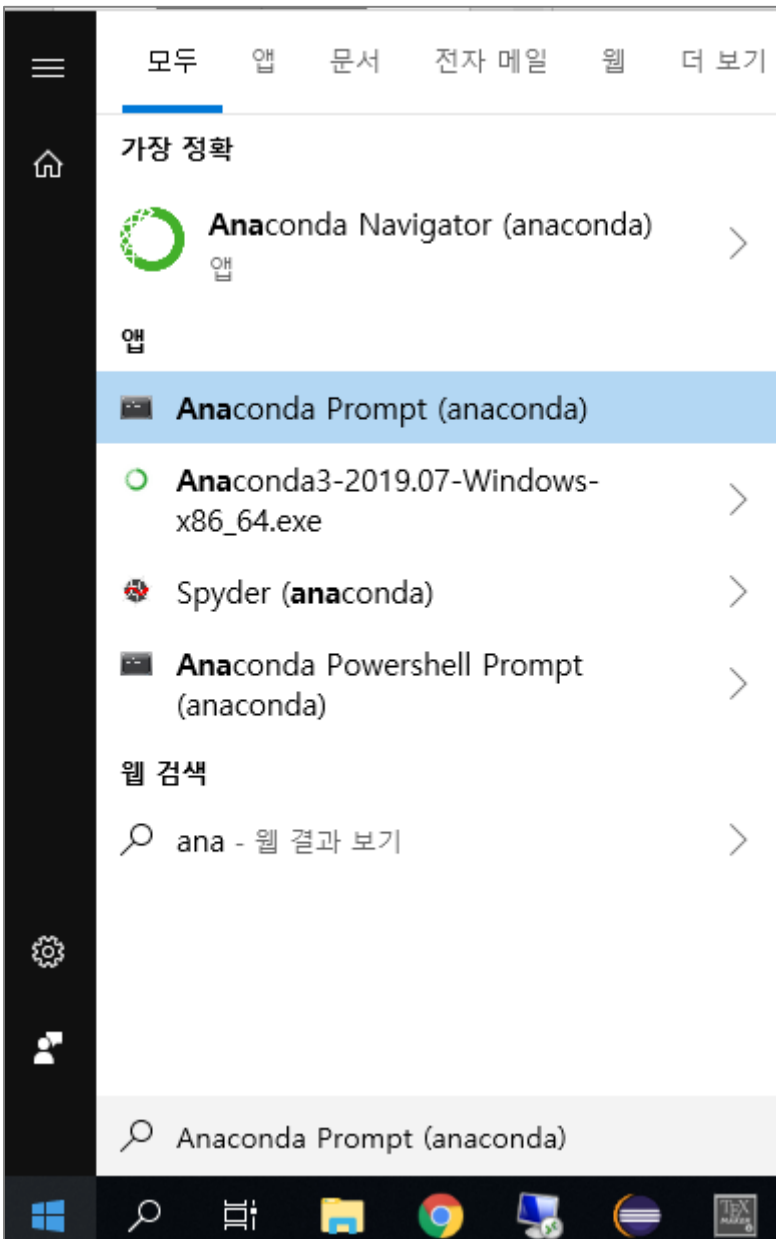
Download > Move to "Desktop"



Download > Move to "Desktop"



Anaconda prompt 실행 및 바탕화면으로 이동



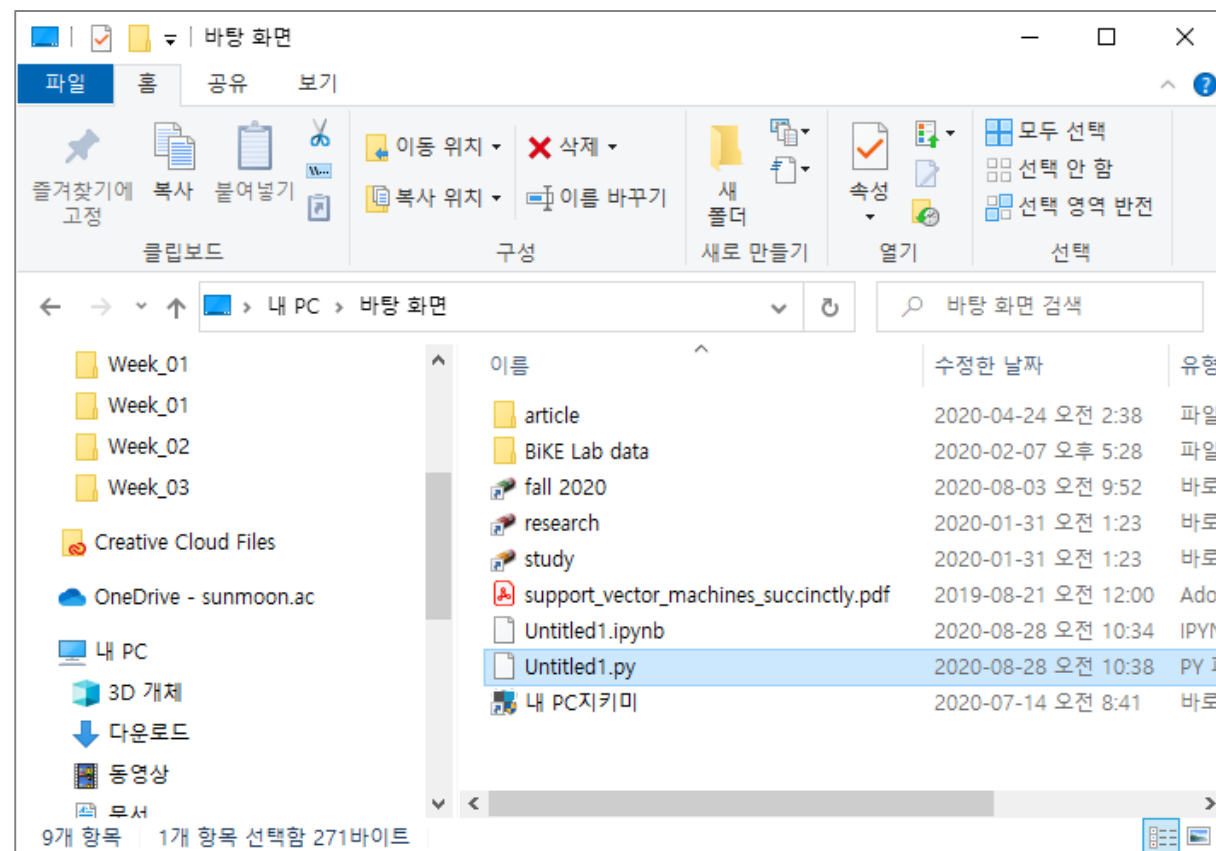
Jupyter (.ipynb) to Python (.py)

```
Anaconda Prompt (anaconda3)

(base) C:\Users\Eung-HeeKim>cd Desktop

(base) C:\Users\Eung-HeeKim\Desktop>jupyter nbconvert --to script Untitled1.ipynb
[NbConvertApp] Converting notebook Untitled1.ipynb to script
[NbConvertApp] Writing 254 bytes to Untitled1.py

(base) C:\Users\Eung-HeeKim\Desktop>
```

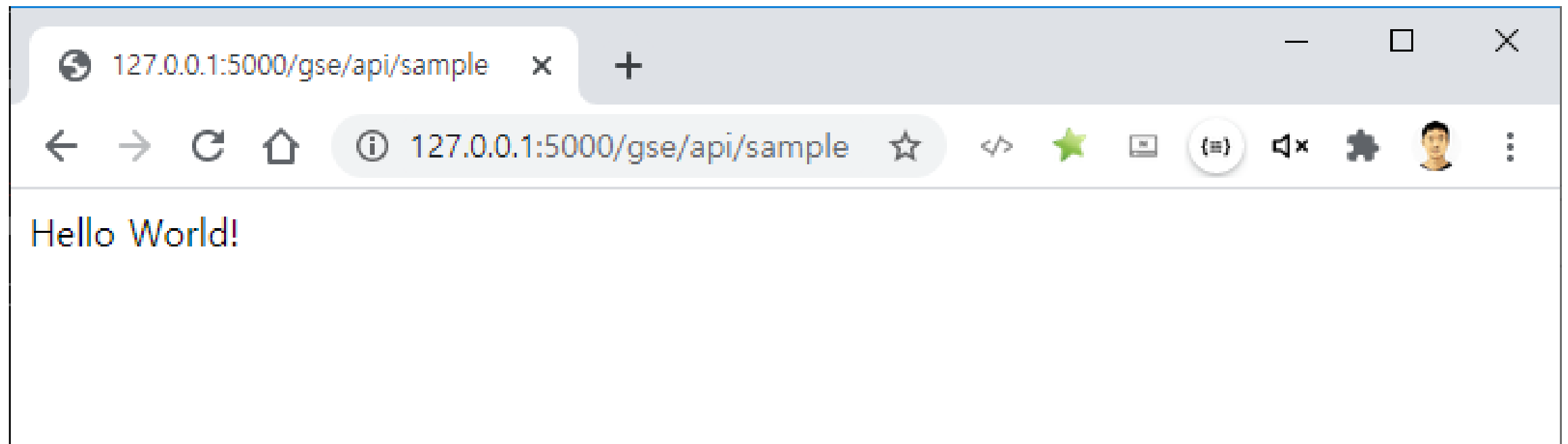


Untitled1.py 실행

```
Anaconda Prompt (anaconda3) - python Untitled1.py

(base) C:\Users\Eung-HeeKim>cd Desktop
(base) C:\Users\Eung-HeeKim\Desktop>jupyter nbconvert --to script Untitled1.ipynb
[NbConvertApp] Converting notebook Untitled1.ipynb to script
[NbConvertApp] Writing 254 bytes to Untitled1.py
(base) C:\Users\Eung-HeeKim\Desktop>python Untitled1.py
* Serving Flask app "Untitled1" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

REST API 호출



Untitled1.py 수정



```
*Untitled1.py - Windows 메모장
파일(F)  편집(E)  서식(O)  보기(V)  도움말(H)

#!/usr/bin/env python
# coding: utf-8

# In[ ]:

from flask import Flask

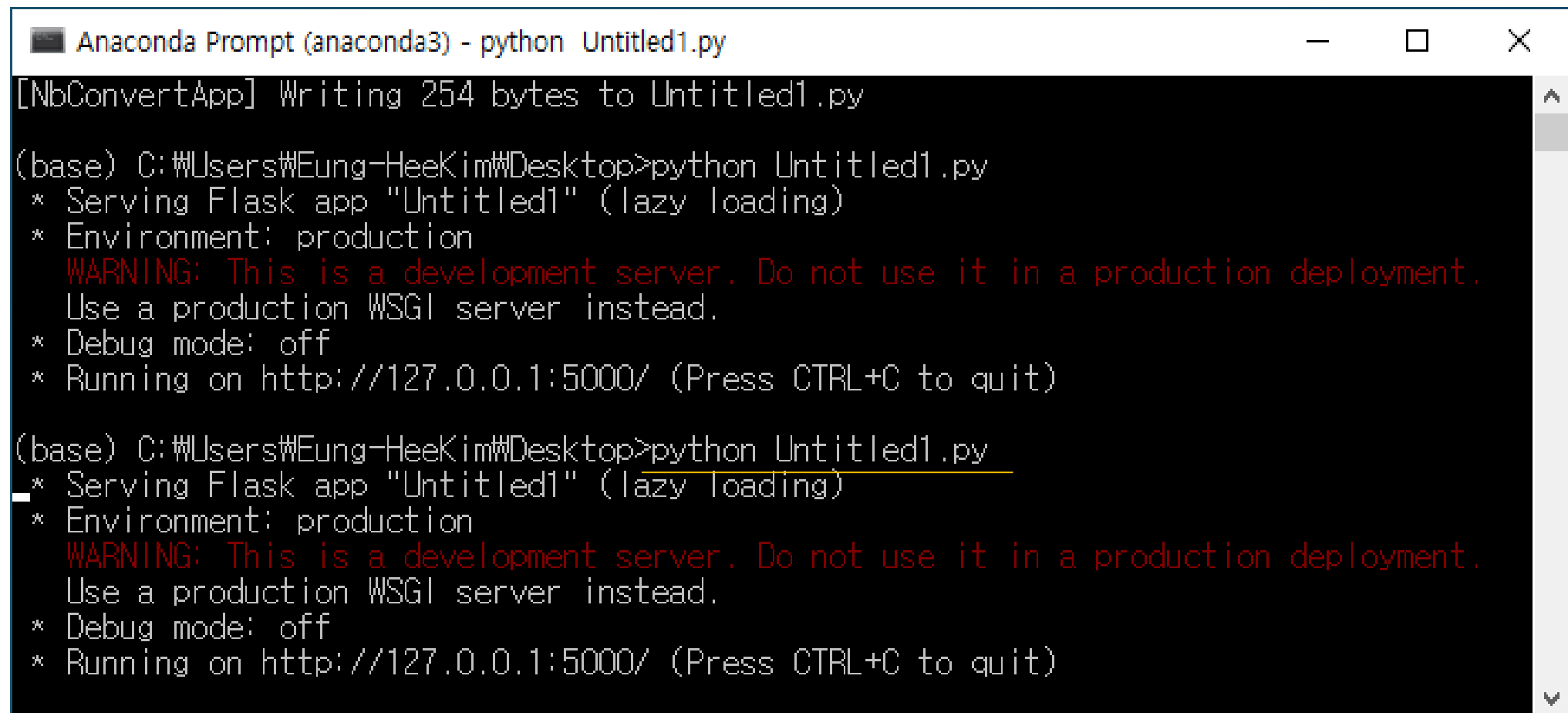
sample_api = Flask(__name__)

@sample_api.route("/gse/api/sample", methods=["GET"])
def hello_world():
    answer = "<html> <body> <h1>Hi</h1> </body> </html>"
    return answer

if __name__ == "__main__":
    sample_api.run()
```

Ln 7, Col 24 100% Windows (CRLF) UTF-8

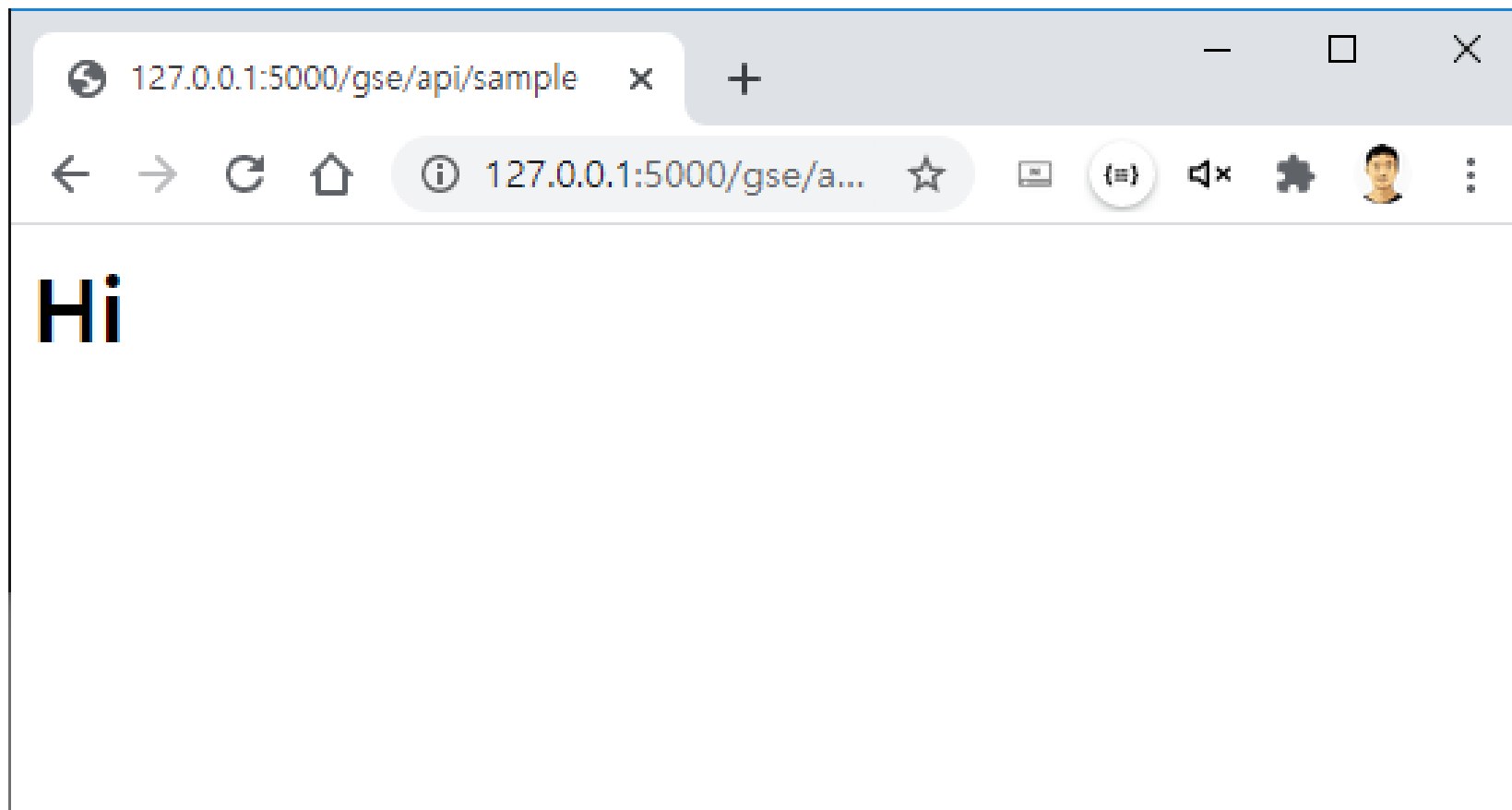
Untitled1.py 재실행
CTRL+C → python Untitled1.py



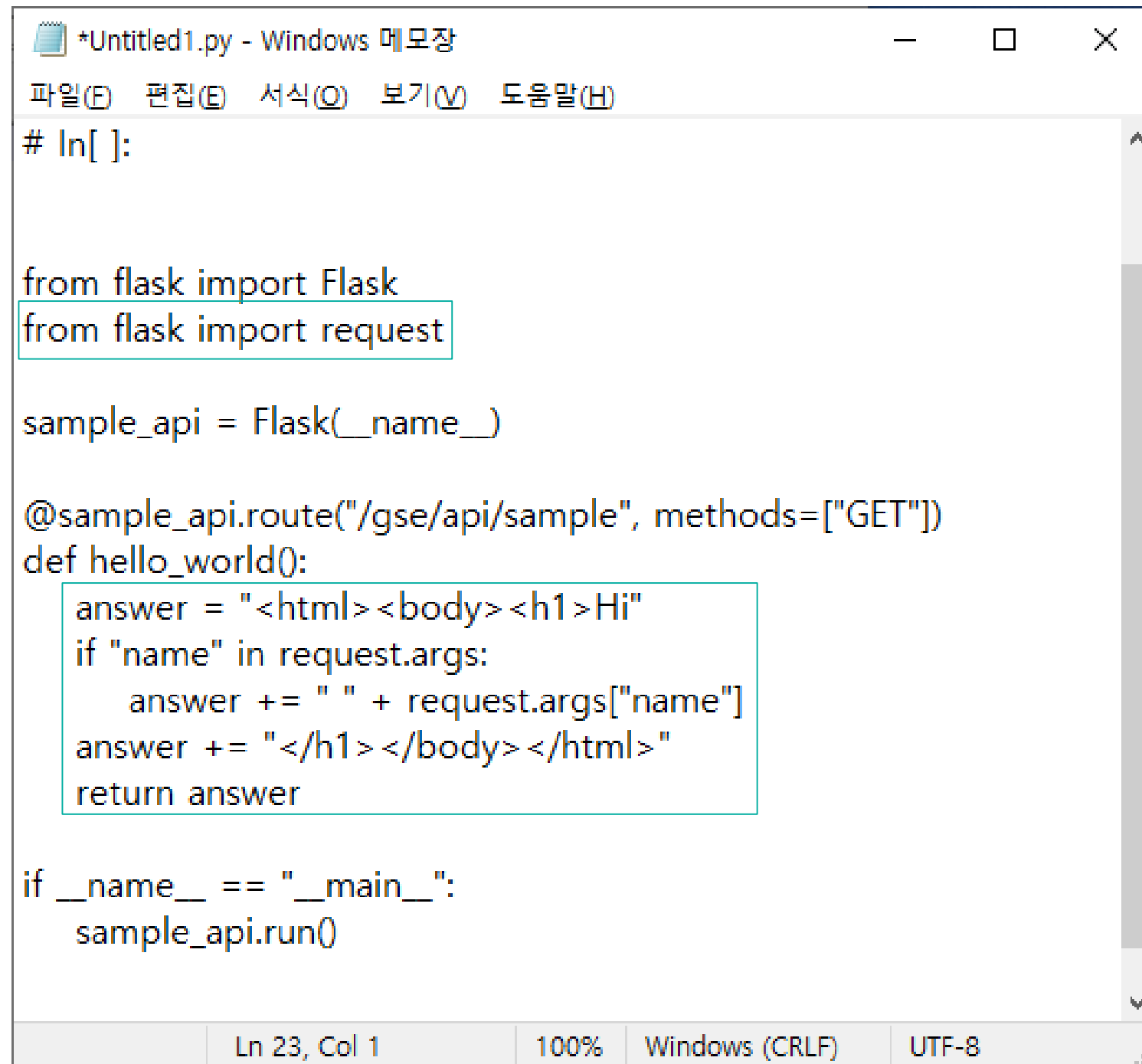
```
Anaconda Prompt (anaconda3) - python Untitled1.py
[NbConvertApp] Writing 254 bytes to Untitled1.py
(base) C:\Users\Eung-HeeKim\Desktop>python Untitled1.py
* Serving Flask app "Untitled1" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

(base) C:\Users\Eung-HeeKim\Desktop>python Untitled1.py
* Serving Flask app "Untitled1" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

REST API 재호출



매개 변수(Parameter) 처리하기



```
*Untitled1.py - Windows 메모장
파일(E)  편집(E)  서식(O)  보기(V)  도움말(H)

# In[ ]:

from flask import Flask
from flask import request

sample_api = Flask(__name__)

@sample_api.route("/gse/api/sample", methods=["GET"])
def hello_world():
    answer = "<html><body><h1>Hi"
    if "name" in request.args:
        answer += " " + request.args["name"]
    answer += "</h1></body></html>"
    return answer

if __name__ == "__main__":
    sample_api.run()
```

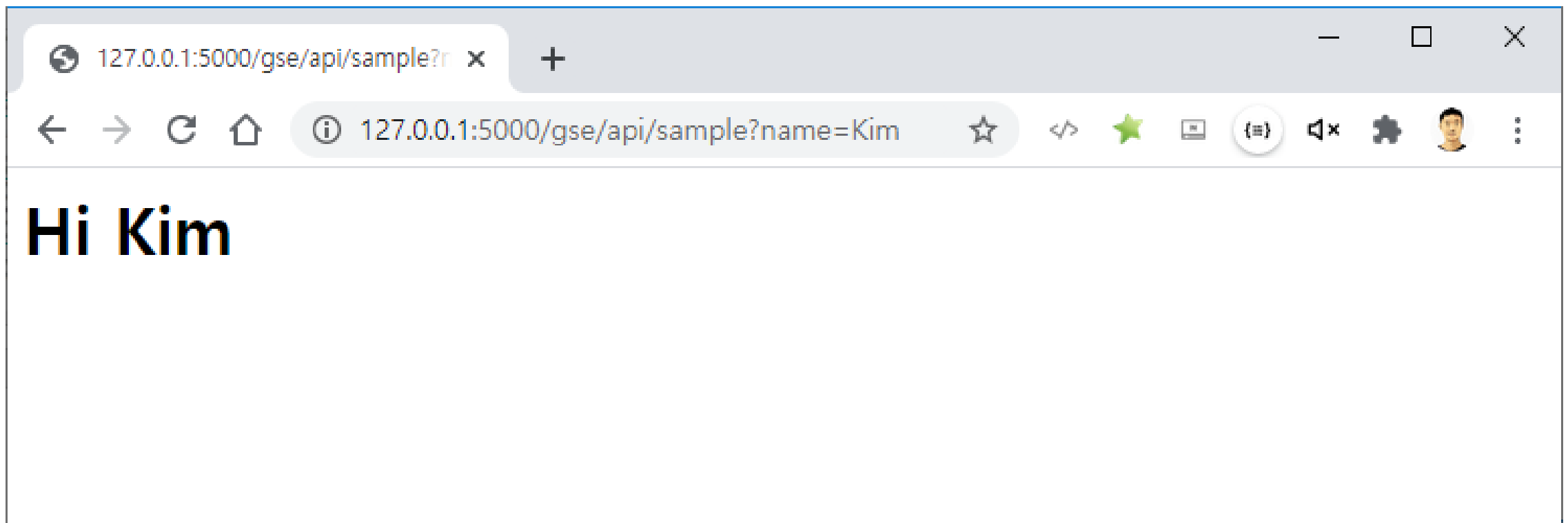
Ln 23, Col 1 100% Windows (CRLF) UTF-8

Untitled.py 재실행

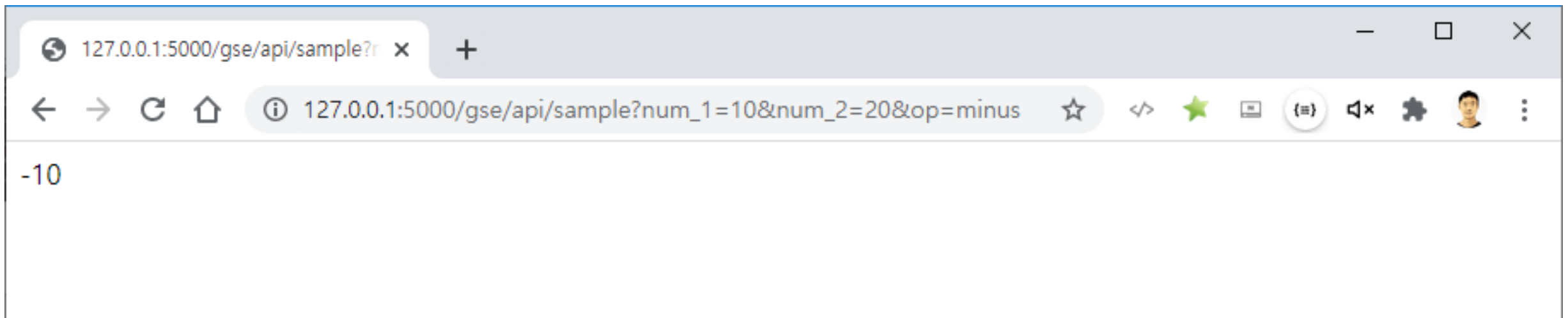
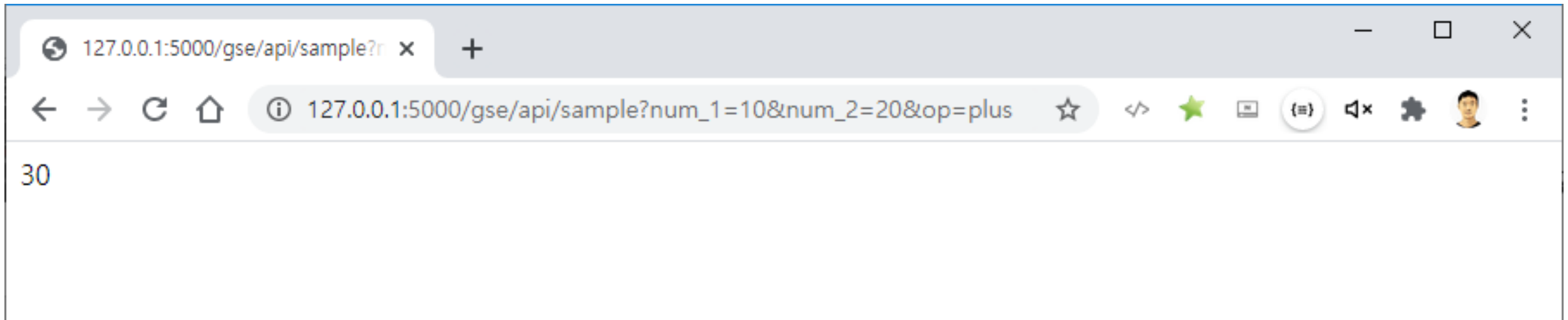
```
Anaconda Prompt (anaconda3) - python Untitled1.py
[NbConvertApp] Writing 254 bytes to Untitled1.py
(base) C:\Users\Eung-HeeKim\Desktop>python Untitled1.py
* Serving Flask app "Untitled1" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

(base) C:\Users\Eung-HeeKim\Desktop>python Untitled1.py
* Serving Flask app "Untitled1" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

REST API 재호출



덧셈 & 뺄셈 REST API 만들기



덧셈 & 뺄셈 REST API 만들기

```
@sample_api.route('/gse/api/sample/', methods=['GET'])
def hello_world( ):
    result = 0
    num_1 = int(request.args["num_1"])
    num_2 = int(request.args["num_2"])
    op = request.args["op"]
    if op == "plus":
        result = num_1 + num_2
    elif op == "minus":
        result = num_1 - num_2
    return str(result)
```

REST API 호출 in Python

The screenshot displays a JupyterLab environment. The top browser window shows the URL `localhost:8888/lab`. The left sidebar shows a file explorer with the directory `/ Advanced data science 2020 /` containing files like `forest.model`, `housing.csv`, `housing.tgz`, and several `Untitled.ipynb` files. The main editor area shows a code cell in `Untitled2.ipynb` with the following Python code:

```
[1]: import requests

url = "http://127.0.0.1:5000/gse/api/sample"

params = {"num_1": 10, "num_2": 20, "op": "minus"}

response = requests.get(url, params)

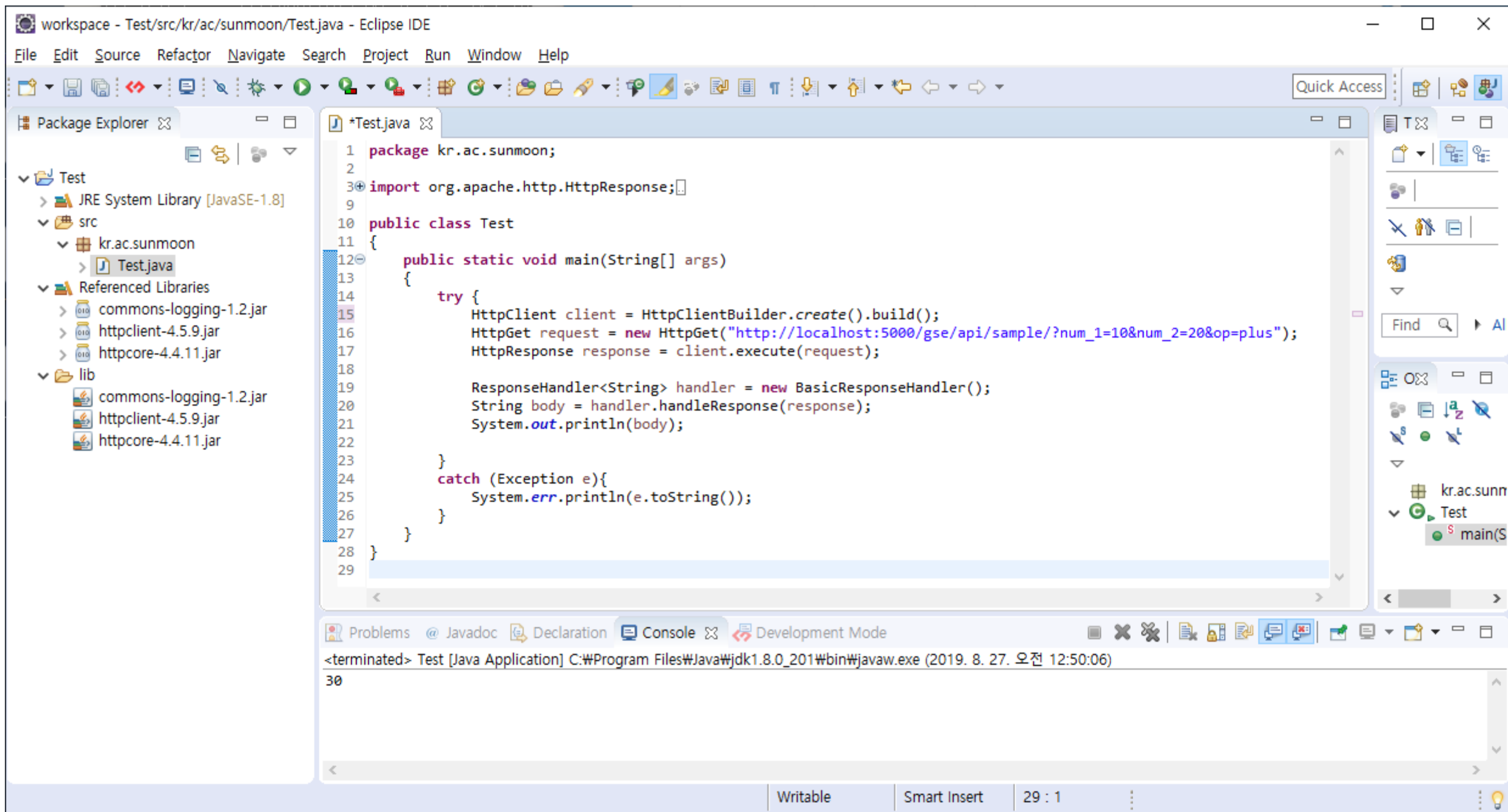
print(response.text)
```

The output of the code cell is `-10`. The bottom status bar indicates the current mode is `Edit`, the kernel is `Python 3`, and the cursor is at `Ln 1, Col 1` in `Untitled2.ipynb`.

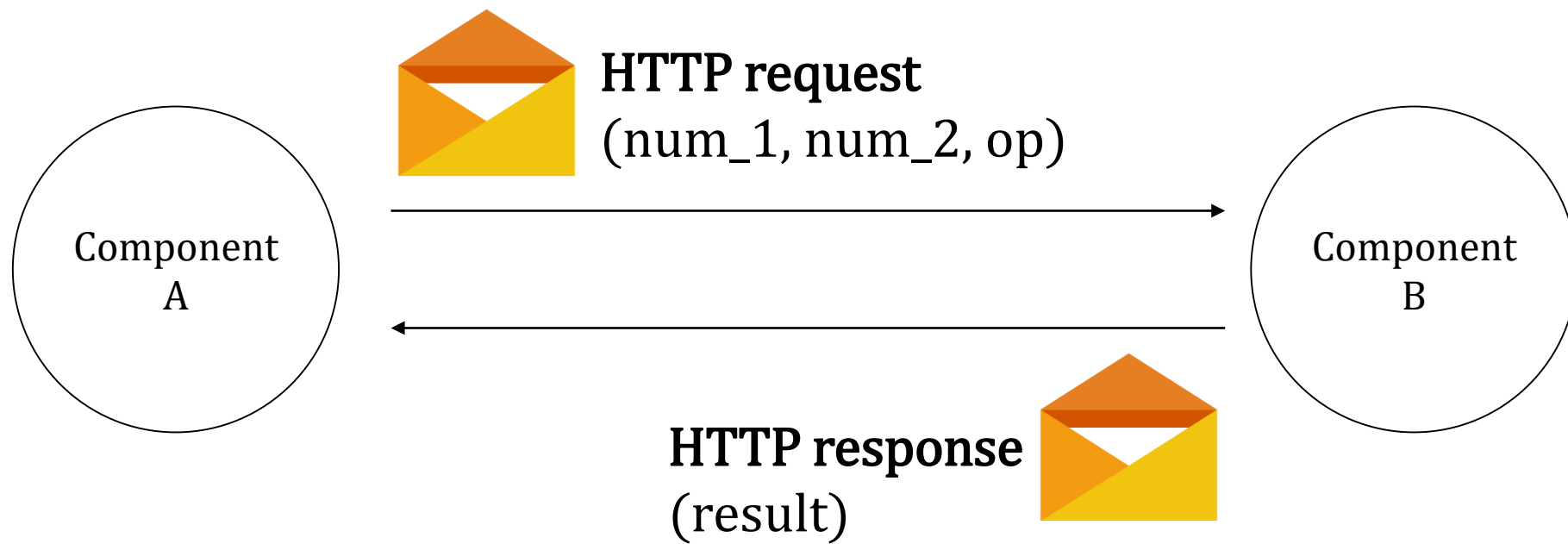
REST API 호출 in Java

- Eclipse 실행
- Java project 생성
- kr.ac.sunmoon 패키지 생성
- lib 폴더 생성
- e-강의동 > 데이터사이언스응용 > 03주차 강의자료
 - 3개의 jar 파일 다운로드 및 lib 폴더에 복사 → build path에 추가
 - Test.java 파일 다운로드 및 kr.ac.sunmoon 패키지에 복사 → 실행

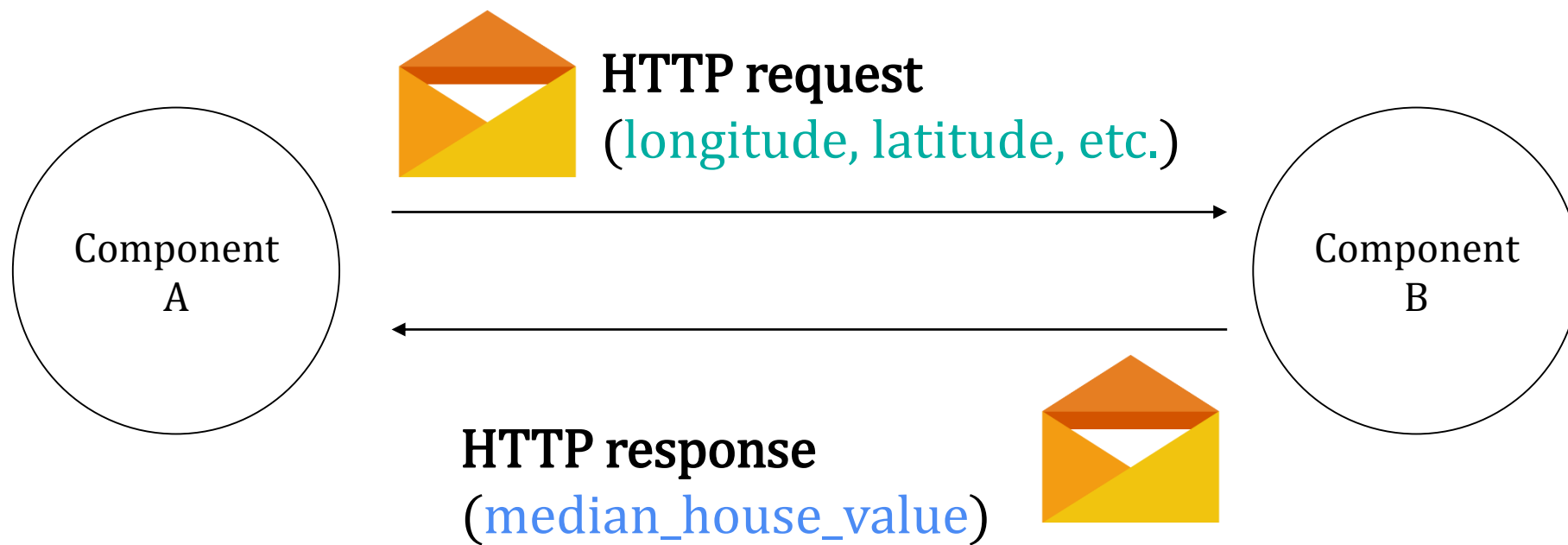
REST API 호출 in Java



What if..



What if..



데이터사이언스응용 Capstone Design

학문 분야 별로 습득한 전문지식을 바탕으로,
학생 스스로 설계, 제작, 평가하는
창의적 종합설계 프로그램

교과명에 캡스톤디자인 또는 종합설계를 부기

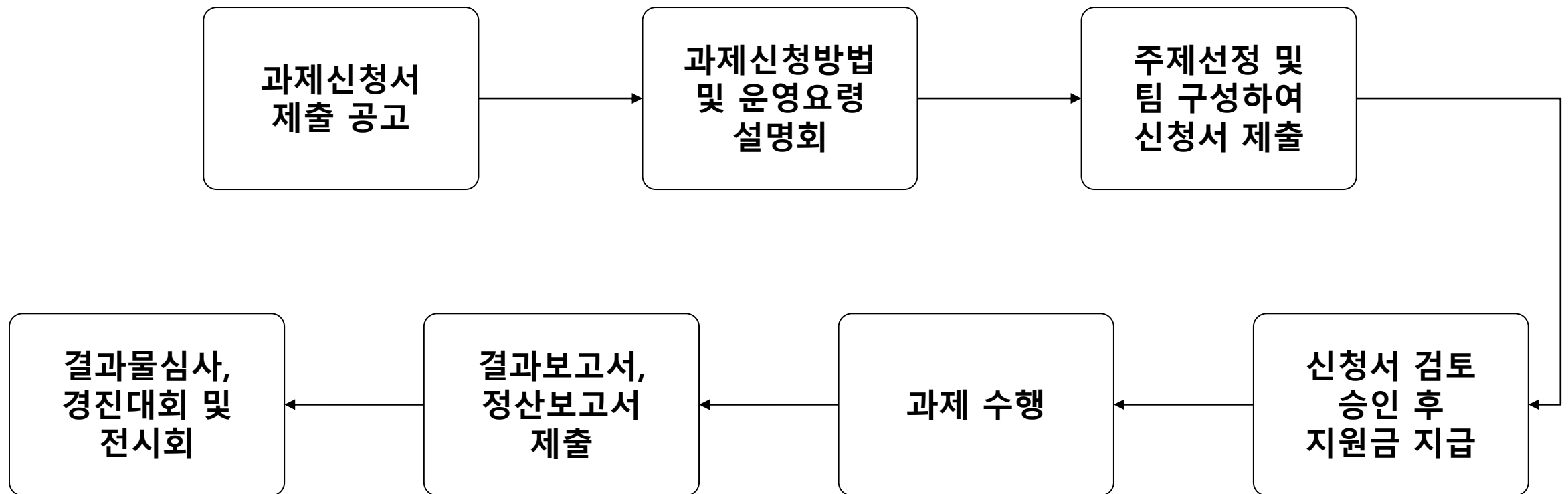
학기 단위로 운영하고 최종결과물 제출



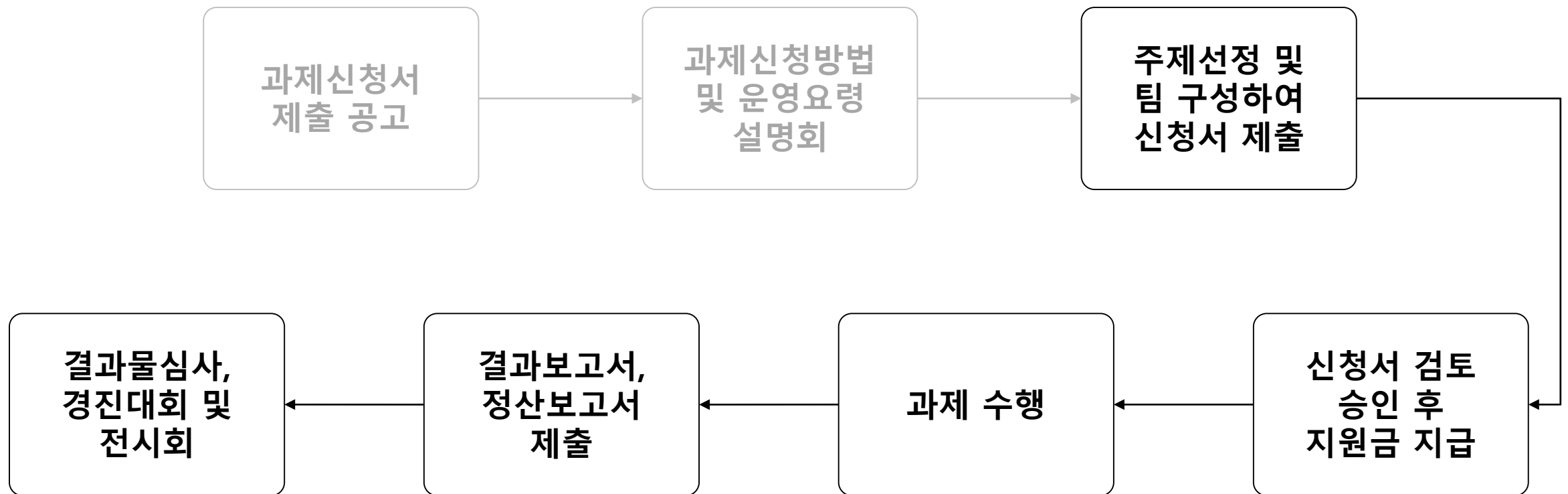
결과물 도출, 시제품 제작 등을 위한
실험·실습비가 지급

학부생의 창의성과 실무능력, 팀워크,
발표능력 향상, 리더의 역할을
수행할 수 있는 능력을 보유한 창의인재로
육성하기 위한 프로그램

캡스톤디자인 진행 가이드



캡스톤디자인 진행 가이드



2020학년도 2학기 지원개요

1. 지원 기간

- ~ 2020년 12월 21일 (월)

2. 지원 대상

- 캡스톤 디자인을 수강하고 있는 3학년 이상 재학생 3명 2명 이상으로 구성된 팀

3. 지원 규모

- 팀당 200,000원

4. 지원 내용

- 실험/실습비, 인쇄비, 문헌구입비, 설문조사비, 교통비, 회의비 등

팀 구성 현황

순번	팀명	팀원
1	제니리아	쿠마자와 유이
		후쿠미쓰 치아키
2	Ajsoftware	노승욱
		스피겔 크릴
3	H:J	이양희
		이수정
		이혜인
4	안시성	우메모토세이야
		방대호
		노무라 타카미치
5	철딱서니	정철우
		김선민
6	AKI	호즈미요시아키
		오타오아키
7	YOLO	유제훈
		키타야마요시아키

캡스톤디자인 운영 프로세스

	학생	교수	학부	센터
학부(과)별 신청 (학기 초)	아이디어 제출	지원금 통장 개설, 팀 구성	신청총괄표 제출 (원본, 협조전)	심사 및 승인, 지원금 지급
팀별 신청 (학기 초)	신청서 및 계획서 제출	주제 설정 및 방향 제시	신청서 및 계획서 수합 및 제출 (원본, 협조전)	과제 등록 보완 요청
과제 수행 (학기 중)	과제 수행	과제 지도	지원금 사용 관리 (소모품 신청)	과제 수행 관리 (소모품 확정, 검수)
결과보고서 제출 (학기 말)	결과보고서 및 정산보고서 제출	결과보고서 및 지원금 사용 확인	결과보고서 및 정산보고서 제출 (원본, 협조전)	서류 검토 및 보완 요청, 지원금 반납 요청
지원금 잔액 반납 (학기 말)			지원금 잔액 반납	반납 확인, 실적 및 결과 보고

세부지원 일정 및 절차

일정	내용	절차
09/01 ~ 09/28	학과별 캡스톤 디자인 지원금 신청 접수	· [서식1] 캡스톤디자인 지원금 신청총괄표, 교육과정표, 강의계획서, 진도표, 통장사본(잔액0원), 체크카드 사본
09/01 ~ 10/14	팀별 과제신청서 접수	· [서식2] 캡스톤디자인 지원신청서, 캡스톤디자인 계획서
09.01 ~ 12.21	학과별 지원금 운영	· [서식3] 캡스톤디자인 지원금 정산보고서, 영수증(카드매출전표, 세금계산서 등), 견적서, 거래명세서, 소모품 구매 신청서, 사진(재료, 활동모습 등), 학과별 통장사용 내역(통장사본)
~ 12/21	캡스톤디자인 결과보고서 및 결과물 제출	· [서식4] 캡스톤디자인 결과보고서, 수행결과, 수행사진, 예산 집행 결과 · [서식5] 캡스톤 디자인 설문지, 학과별 통장사용내역(통장사본 등)

지원금 세부 항목 소개

항목	유의사항
재료 및 실험실습비	<ul style="list-style-type: none"> · 작품제작을 위한 재료/부품/시약/가공비 등 · 문구용품의 경우, 1인당 1만원까지 사용가능 · 신청금 내에서 액수 제한 없이 사용 가능 · 영수증, 견적서, 거래명세서, 소모품신청서
인쇄·제본비	<ul style="list-style-type: none"> · 최종 성과물 인쇄 및 제본 · 신청금 내에서 액수 제한 없이 사용 가능 · 영수증, 견적서, 거래명세서, 소모품 신청서, 인쇄물 시안, 표지 시안
도서 및 문헌구입비	<ul style="list-style-type: none"> · 작품제작/과제수행과 관련된 도서 구입 · 과제 종료 후, 산학협력교육센터로 반납 · 영수증, 견적서, 거래명세서, 소모품 신청서 · 최대 10만원까지 사용 가능
설문조사비	<ul style="list-style-type: none"> · 과제 수행과 관련된 설문조사 시 사용가능 · 최대 10만원까지 사용 가능 · 학생 기념품: 개당 2천원 이하 · 일반인 기념품: 개당 5천원 이하
교통비	<ul style="list-style-type: none"> · 과제 수행을 위한 외부 활동시 교통비 실비 지급 · 최대 10만원까지 사용 가능 · 택시 요금 사용 불가 · 시내(천안/아산) 활동 시: 1인당 1일 8천원 · 교통비 실비 영수증(기차표, 시외버스 티켓, 교통카드 내역) · 학생 사비 지출 후 학과 통장에서 환불 이체
회의비	<ul style="list-style-type: none"> · 과제수행과 관련된 회의시 사용 가능 · 최대 6만원까지 사용 가능 · 하루 1인당 1만원까지 사용가능 · 활동보고서

지원금 사용 유의 사항

구분	유의사항
공통	· 영수증에 품목이 표시되도록 발급(견적서 생략 가능)
	· 구매한 재료의 사진 첨부 필수
	· 오후 10시 이후, 주말 혹은 공휴일 카드 사용 불가
	· 교통비 사용 후 계좌 이체 시, 신분증 사본 + 통장 사본 제출
회의비	· 학과 사무실에서 체크카드 대여하여 사용(일지 작성)
	· 코나킹 사용 불가(산학협력단 운영 기업)
	· 회의 증빙 사진이 첨부된 활동보고서 필수 제출
지원불가	· 소프트웨어, 컴퓨터 관련 주변 기기(예: USB 및 외장 하드 등)
	· 인쇄 소모품(예: 프린터 토너 및 복사용지 등)
	· 사무장비 (예: 책장, 책상, 의자 등)
	· 중고나라 등 개인과의 거래 불가

안내 문서 및 서류 양식

- e-강의동 > 공지사항 > 캡스톤디자인 관련 서류
 - 2020학년도 2학기 캡스톤디자인 지원 안내 사항.hwp
 - 캡스톤디자인 제출 서류 모음.hwp

The screenshot shows the Seonmun University e-Learning portal. The header includes the university logo, the text '선문대학교 e-강의동', a language dropdown set to '한국어', and user information for '김응희' with a notification badge showing '21' and a '로그아웃' button. Navigation tabs for '교육현황', '커뮤니티', and '소개' are present. A left sidebar titled '강의과목' lists various course categories, with '공지사항' highlighted. The main content area is titled '공지사항' and shows a breadcrumb trail: 'H > 데이터사이언스응용(CapstoneDesign) > 공지사항'. Below this is a search bar with the text '검색' and a 'search' button. A table lists notices with columns for '번호' (Number), '제목' (Subject), '첨부' (Attachment), and '공개일' (Publication Date). The table contains two entries: one about a Zoom meeting ID and another about Capstone Design documents. A '글쓰기' (Write) button is located at the bottom right of the notice list.

번호	제목	첨부	공개일
	데이터사이언스응용 Zoom 회의 ID 및 수업 일정 김응희 조회 80		2020.08.28 오후 2:00
1	캡스톤디자인 관련 서류 김응희 조회 0		2020.09.16 오후 3:30

이번 시간 내 알려주어야 할 사항

순번	팀명	팀원	주제
1	제니리아	쿠마자와 유이	
		후쿠미쓰 치아키	
2	Ajsoftware	노승욱	
		스피겔 크릴	
3	H:J	이양희	
		이수정	
		이혜인	
4	안시성	우메모토세이야	
		방대호	
		노무라 타카미치	
5	철딱서니	정철우	
		김선민	
6	AKI	호즈미요시아키	
		오타오아키	
7	YOLO	유제훈	
		키타야마요시아키	

Thank you