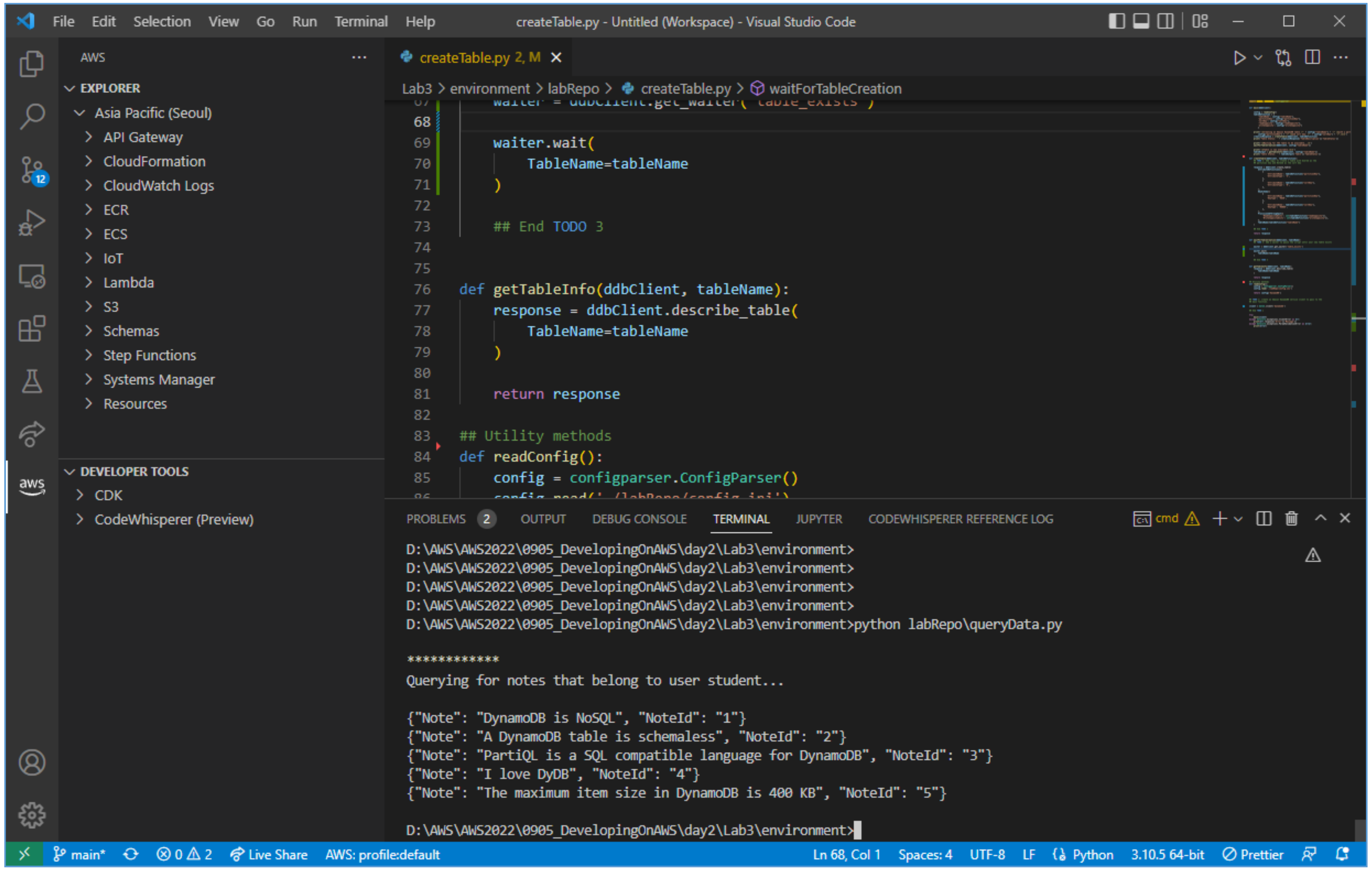
# Developing On AWS

박경규

## 1. AWS 개발환경

# 로컬 환경 : IDE(VS Code)

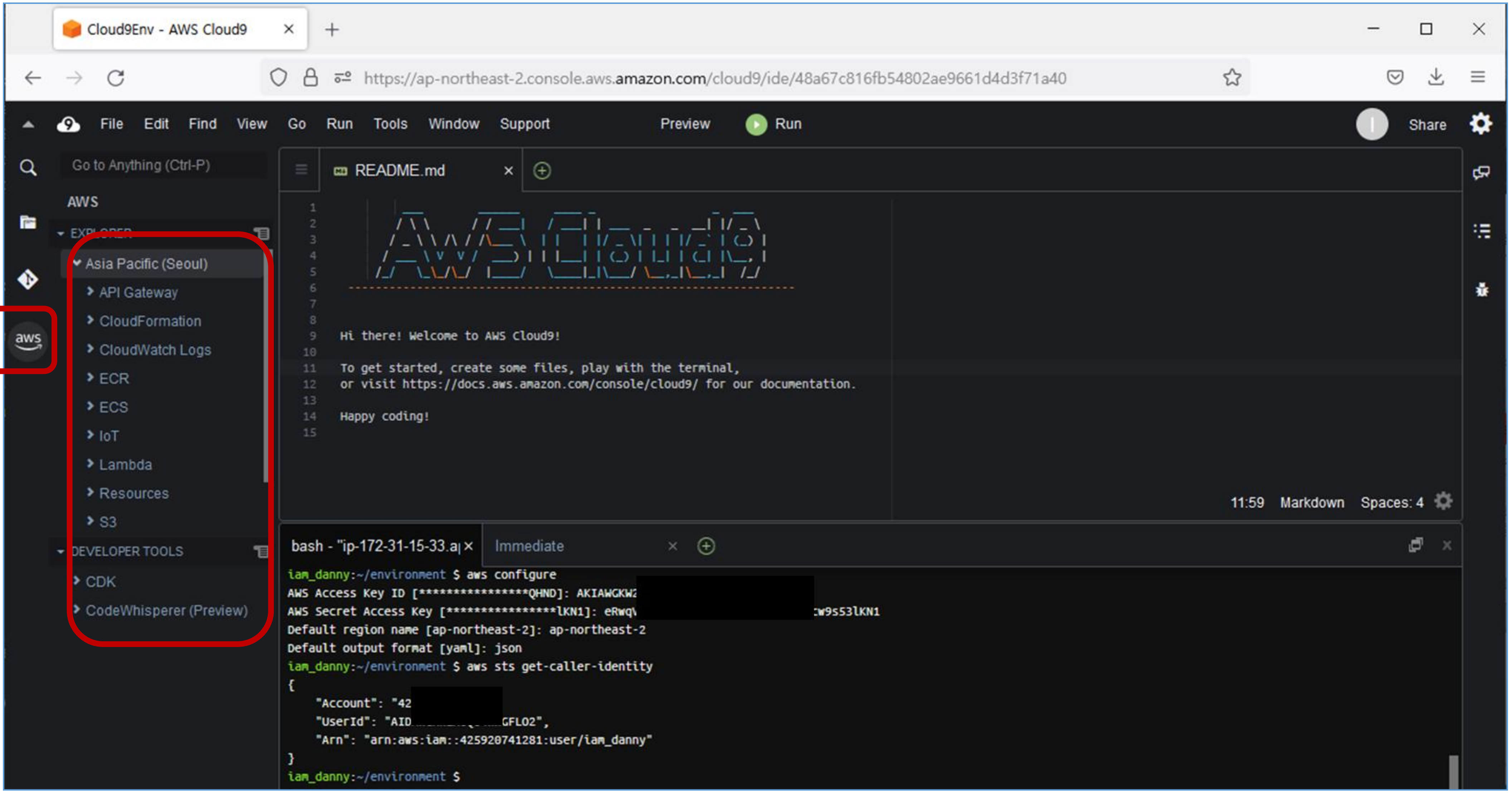

File Edit Selection View Go Run Terminal Help
createTable.py - Untitled (Workspace) - Visual Studio Code
AWS
EXPLORER
Asia Pacific (Seoul)
API Gateway
CloudFormation
CloudWatch Logs
ECR
ECS
IoT
Lambda
S3
Schemas
Step Functions
Systems Manager
Resources
DEVELOPER TOOLS
CDK
CodeWhisperer (Preview)
createTable.py 2, M
Lab3 > environment > labRepo > createTable.py > waitForTableCreation
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER CODEWHISPERER REFERENCE LOG
main* 0 2 Live Share AWS: profile:default Ln 68, Col 1 Spaces: 4 UTF-8 LF Python 3.10.5 64-bit Prettier

```python
    waiter = ddbClient.get_waiter('table_exists')

    waiter.wait(
        TableName=tableName
    )

    ## End TODO 3


def getTableInfo(ddbClient, tableName):
    response = ddbClient.describe_table(
        TableName=tableName
    )

    return response

## Utility methods
def readConfig():
    config = configparser.ConfigParser()
    config.read('./labRepo/config.ini')
```

```
D:\AWS\AWS2022\0905_DevelopingOnAWS\day2\Lab3\environment>
D:\AWS\AWS2022\0905_DevelopingOnAWS\day2\Lab3\environment>
D:\AWS\AWS2022\0905_DevelopingOnAWS\day2\Lab3\environment>
D:\AWS\AWS2022\0905_DevelopingOnAWS\day2\Lab3\environment>
D:\AWS\AWS2022\0905_DevelopingOnAWS\day2\Lab3\environment>python labRepo\queryData.py

************
Querying for notes that belong to user student...

{"Note": "DynamoDB is NoSQL", "NoteId": "1"}
{"Note": "A DynamoDB table is schemaless", "NoteId": "2"}
{"Note": "PartiQL is a SQL compatible language for DynamoDB", "NoteId": "3"}
{"Note": "I love DyDB", "NoteId": "4"}
{"Note": "The maximum item size in DynamoDB is 400 KB", "NoteId": "5"}

D:\AWS\AWS2022\0905_DevelopingOnAWS\day2\Lab3\environment>
```

# 클라우드 환경 : AWS Cloud9

AWS Explorer를 사용하여 다양한 AWS 서비스의 리소스를 확인하고, 생성하고, 업데이트하고, 삭제할 수 있습니다.



```
Hi there! Welcome to AWS Cloud9!

To get started, create some files, play with the terminal,
or visit https://docs.aws.amazon.com/console/cloud9/ for our documentation.

Happy coding!
```

```
iam_danny:~/environment $ aws configure
AWS Access Key ID [****************QHND]: AKIAWGKW2[redacted]
AWS Secret Access Key [****************lKN1]: eRwq[redacted]w9s53lKN1
Default region name [ap-northeast-2]: ap-northeast-2
Default output format [yaml]: json
iam_danny:~/environment $ aws sts get-caller-identity
{
    "Account": "42[redacted]
    "UserId": "AID[redacted]GFLO2",
    "Arn": "arn:aws:iam::425920741281:user/iam_danny"
}
iam_danny:~/environment $
```

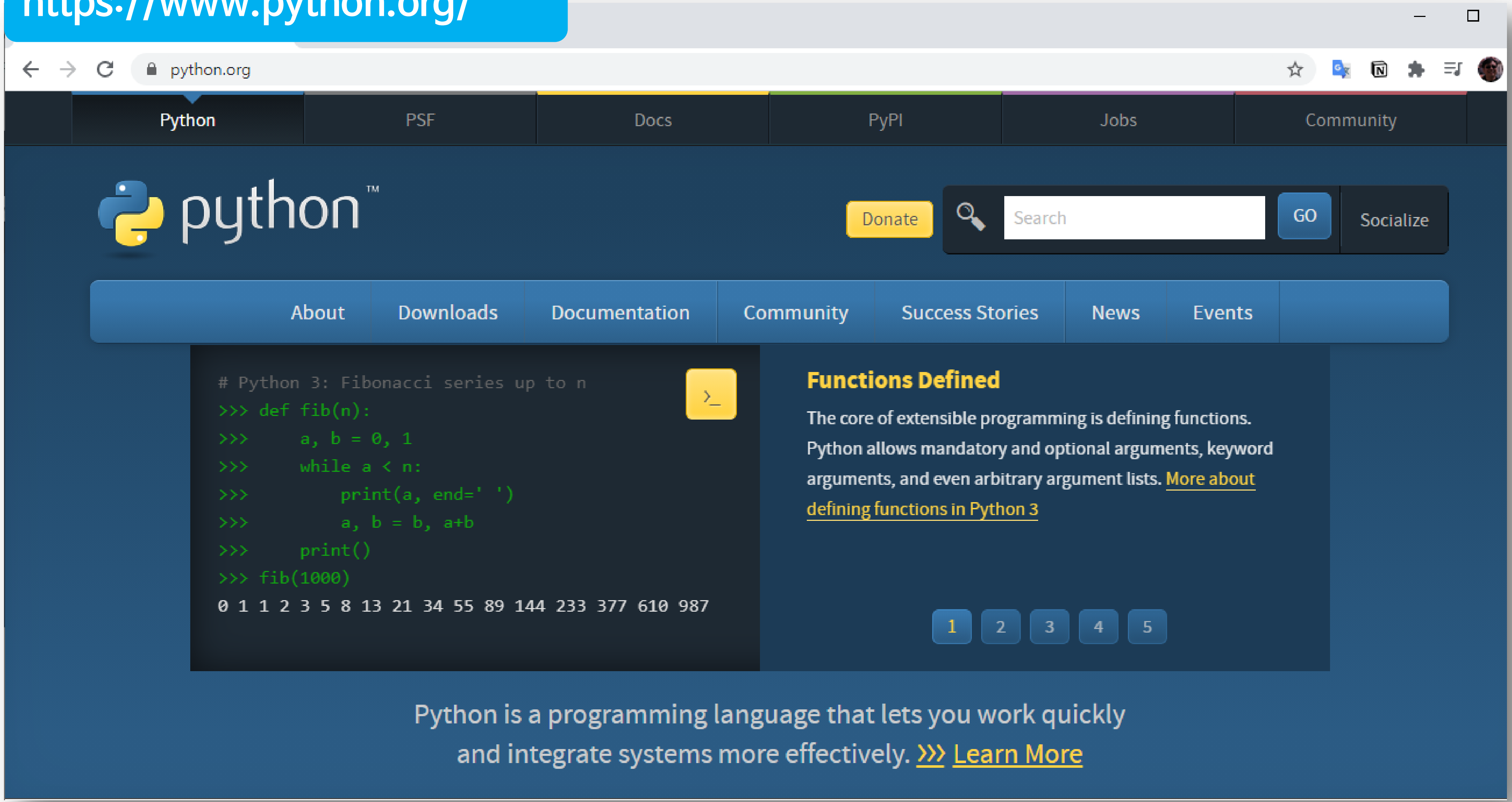
# 2. 로컬 개발 환경

파이썬(Python)설치

VS Code 설치

# Python 사이트

https://www.python.org/



```
# Python 3: Fibonacci series up to n
>>> def fib(n):
>>>     a, b = 0, 1
>>>     while a < n:
>>>         print(a, end=' ')
>>>         a, b = b, a+b
>>>     print()
>>> fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

**Functions Defined**

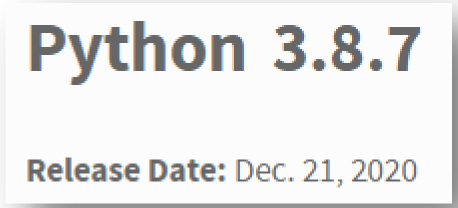
The core of extensible programming is defining functions. Python allows mandatory and optional arguments, keyword arguments, and even arbitrary argument lists. More about defining functions in Python 3

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

# Python 설치파일 다운로드

## ■ 파이썬 설치

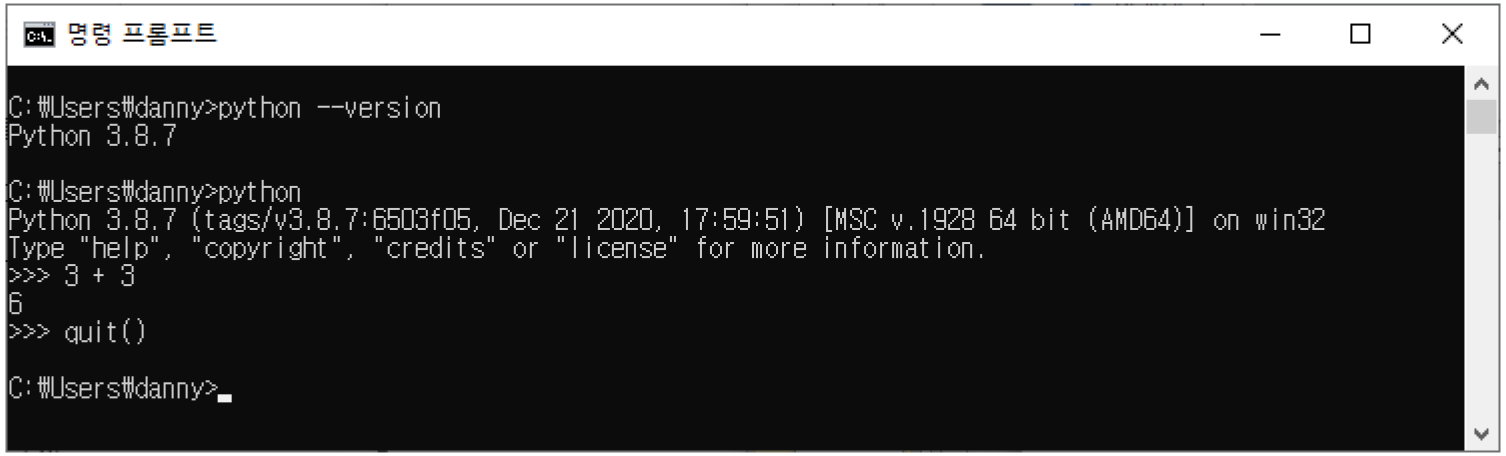
https://www.python.org/downloads/


Python 3.8.7

**Release Date:** Dec. 21, 2020

## ■ 파이썬 실행

- 버전 확인 : python --version
- 실행 : python
- 종료 : quit()



```
C:₩Users₩danny>python --version
Python 3.8.7

C:₩Users₩danny>python
Python 3.8.7 (tags/v3.8.7:6503f05, Dec 21 2020, 17:59:51) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 3 + 3
6
>>> quit()

C:₩Users₩danny>_
```

## Python 가상환경 설치

**가상 환경(virtual environment)으로 프로젝트별로 독립된 파이썬 실행 환경을 사용할 수 있습니다.**

- 가상환경 생성 : python -m venv venv

- 가상환경 실행
  windows : venv₩Scripts₩activate.bat
  Linux, macOS : source venv/bin/activate

- JupyterLab/ Jupyter Notebook 설치
  pip install jupyterlab
  pip install notebook

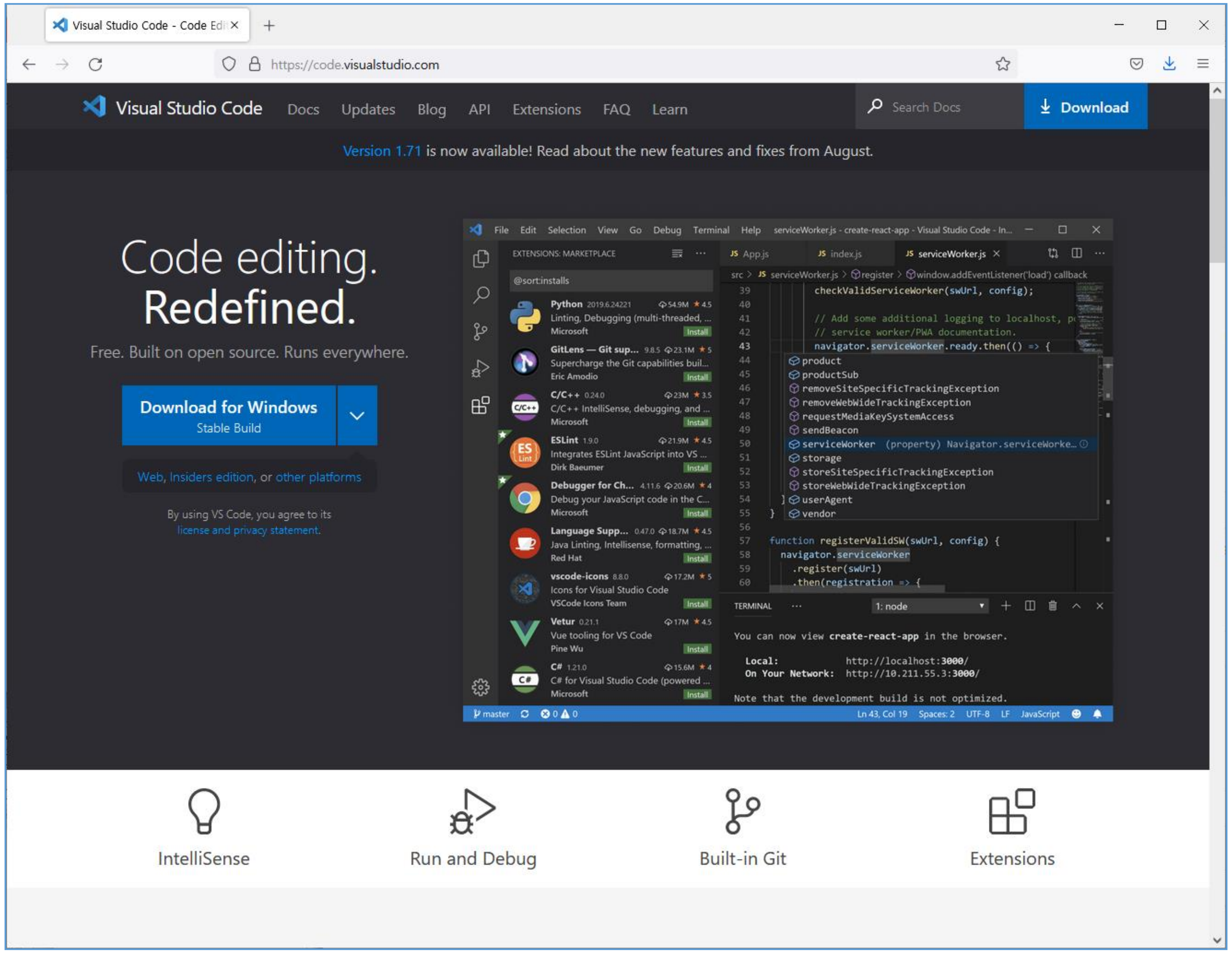
- JupyterLab/ Jupyter Notebook 실행
  ```
  jupyter-lab          (또는 jupyter lab)
  jupyter-notebook     (또는 jupyter notebook)
  ```

- 패키지 목록 관리
  pip freeze > requirements.txt
  pip install -r requirements.txt

# VS Code 설치

## ■ VS Code 설치

**https://code.visualstudio.com/**



Visual Studio Code - Code Edi
https://code.visualstudio.com
Visual Studio Code
Docs
Updates
Blog
API
Extensions
FAQ
Learn
Search Docs
Download

Version 1.71 is now available! Read about the new features and fixes from August.

Code editing.
Redefined.

Free. Built on open source. Runs everywhere.

Download for Windows
Stable Build

Web, Insiders edition, or other platforms

By using VS Code, you agree to its
license and privacy statement.

File
Edit
Selection
View
Go
Debug
Terminal
Help
serviceWorker.js - create-react-app - Visual Studio Code - In...
EXTENSIONS: MARKETPLACE
@sort:installs
App.js
index.js
serviceWorker.js
src > serviceWorker.js > register > window.addEventListener('load') callback

Python 2019.6.24221 54.9M ★4.5
Linting, Debugging (multi-threaded, ...
Microsoft Install

GitLens — Git sup... 9.8.5 23.1M ★5
Supercharge the Git capabilities buil...
Eric Amodio Install

C/C++ 0.24.0 23M ★3.5
C/C++ IntelliSense, debugging, and ...
Microsoft Install

ESLint 1.9.0 21.9M ★4.5
Integrates ESLint JavaScript into VS ...
Dirk Baeumer Install

Debugger for Ch... 4.11.6 20.6M ★4
Debug your JavaScript code in the C...
Microsoft Install

Language Supp... 0.47.0 18.7M ★4.5
Java Linting, Intellisense, formatting, ...
Red Hat Install

vscode-icons 8.8.0 17.2M ★5
Icons for Visual Studio Code
VSCode Icons Team Install

Vetur 0.21.1 17M ★4.5
Vue tooling for VS Code
Pine Wu Install

C# 1.21.0 15.6M ★4
C# for Visual Studio Code (powered ...
Microsoft Install

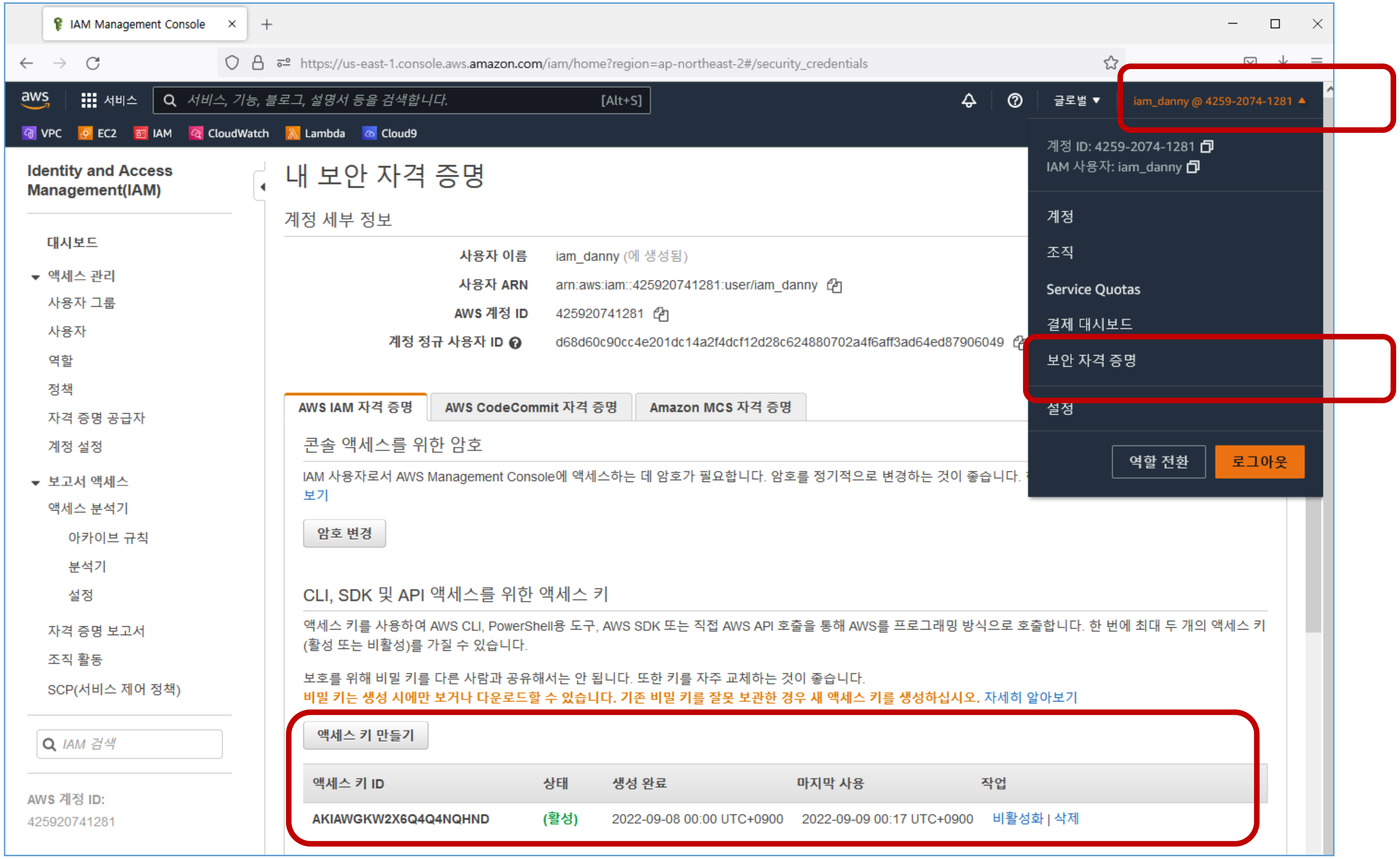
```
          checkValidServiceWorker(swUrl, config);

          // Add some additional logging to localhost, p
          // service worker/PWA documentation.
          navigator.serviceWorker.ready.then(() => {
             product
             productSub
             removeSiteSpecificTrackingException
             removeWebWideTrackingException
             requestMediaKeySystemAccess
             sendBeacon
             serviceWorker  (property) Navigator.serviceWorke...
             storage
             storeSiteSpecificTrackingException
             storeWebWideTrackingException
        ]    userAgent
    }        vendor

function registerValidSW(swUrl, config) {
  navigator.serviceWorker
    .register(swUrl)
    .then(registration => {
```

```
You can now view create-react-app in the browser.

  Local:            http://localhost:3000/
  On Your Network:  http://10.211.55.3:3000/

Note that the development build is not optimized.
```

IntelliSense
Run and Debug
Built-in Git
Extensions

## 3. AWS CLI 설치

# AWS Console : 액세스 키 만들기



IAM Management Console

https://us-east-1.console.aws.amazon.com/iam/home?region=ap-northeast-2#/security_credentials

aws 서비스 서비스, 기능, 블로그, 설명서 등을 검색합니다. [Alt+S] 글로벌 iam_danny @ 4259-2074-1281

VPC EC2 IAM CloudWatch Lambda Cloud9

계정 ID: 4259-2074-1281
IAM 사용자: iam_danny

계정
조직
Service Quotas
결제 대시보드
보안 자격 증명
설정

역할 전환 로그아웃

Identity and Access Management(IAM)

대시보드
액세스 관리
사용자 그룹
사용자
역할
정책
자격 증명 공급자
계정 설정
보고서 액세스
액세스 분석기
아카이브 규칙
분석기
설정
자격 증명 보고서
조직 활동
SCP(서비스 제어 정책)

IAM 검색

AWS 계정 ID:
425920741281

내 보안 자격 증명

계정 세부 정보

| | |
|---|---|
| 사용자 이름 | iam_danny (에 생성됨) |
| 사용자 ARN | arn:aws:iam::425920741281:user/iam_danny |
| AWS 계정 ID | 425920741281 |
| 계정 정규 사용자 ID | d68d60c90cc4e201dc14a2f4dcf12d28c624880702a4f6aff3ad64ed87906049 |

AWS IAM 자격 증명 | AWS CodeCommit 자격 증명 | Amazon MCS 자격 증명

콘솔 액세스를 위한 암호

IAM 사용자로서 AWS Management Console에 액세스하는 데 암호가 필요합니다. 암호를 정기적으로 변경하는 것이 좋습니다.
보기

암호 변경

CLI, SDK 및 API 액세스를 위한 액세스 키

액세스 키를 사용하여 AWS CLI, PowerShell용 도구, AWS SDK 또는 직접 AWS API 호출을 통해 AWS를 프로그래밍 방식으로 호출합니다. 한 번에 최대 두 개의 액세스 키(활성 또는 비활성)를 가질 수 있습니다.

보호를 위해 비밀 키를 다른 사람과 공유해서는 안 됩니다. 또한 키를 자주 교체하는 것이 좋습니다.
비밀 키는 생성 시에만 보거나 다운로드할 수 있습니다. 기존 비밀 키를 잘못 보관한 경우 새 액세스 키를 생성하십시오. 자세히 알아보기

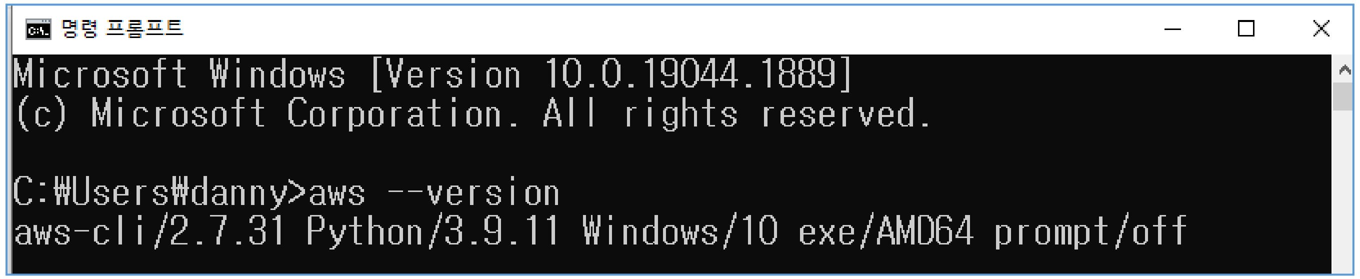
액세스 키 만들기

| 액세스 키 ID | 상태 | 생성 완료 | 마지막 사용 | 작업 |
|---|---|---|---|---|
| AKIAWGKW2X6Q4Q4NQHND | (활성) | 2022-09-08 00:00 UTC+0900 | 2022-09-09 00:17 UTC+0900 | 비활성화 \| 삭제 |

# AWS CLI 설치

## AWS CLI 설치

- https://docs.aws.amazon.com/ko_kr/cli/latest/userguide/getting-started-install.html
- aws --version



명령 프롬프트

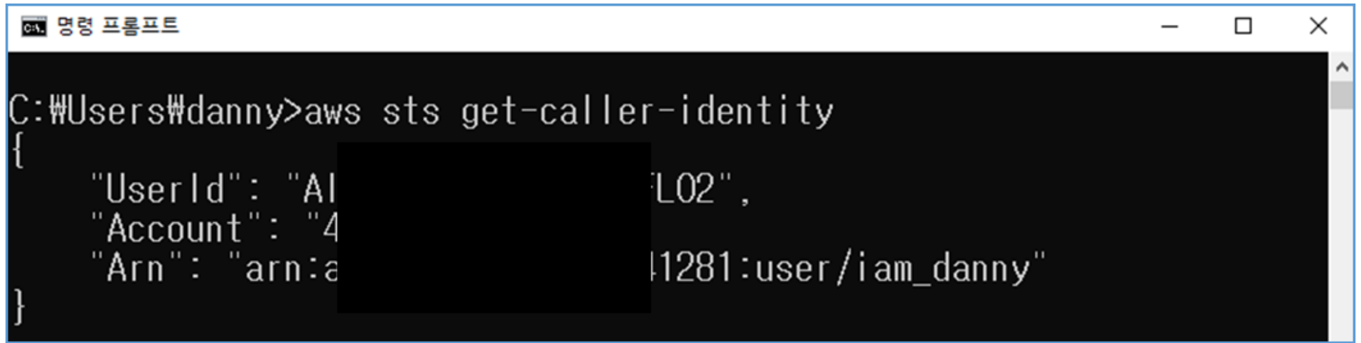
```
Microsoft Windows [Version 10.0.19044.1889]
(c) Microsoft Corporation. All rights reserved.

C:₩Users₩danny>aws --version
aws-cli/2.7.31 Python/3.9.11 Windows/10 exe/AMD64 prompt/off
```

## aws configure

- AWS Access Key ID [비워 둠]: ENTER 키를 누릅니다.
- AWS Secret Access Key [비워 둠]: ENTER 키를 누릅니다.
- Default region name [적절한 리전으로 업데이트]: REGION
- Default output format [yaml으로 업데이트]: yaml

[redacted]

## aws sts get-caller-identity

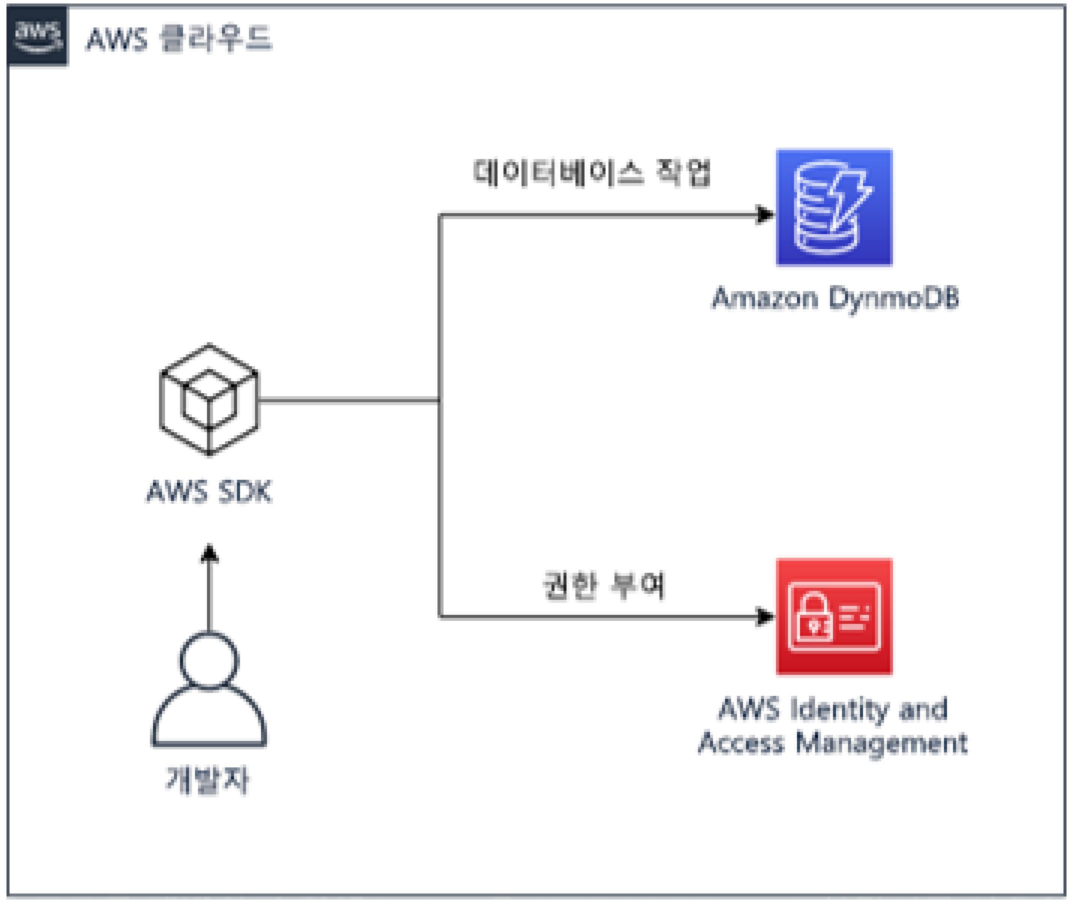


명령 프롬프트

```
C:₩Users₩danny>aws sts get-caller-identity
{
    "UserId": "AI[redacted]L02",
    "Account": "4
    "Arn": "arn:a1281:user/iam_danny"
}
```

## 4. DynamoDB

# DynamoDB

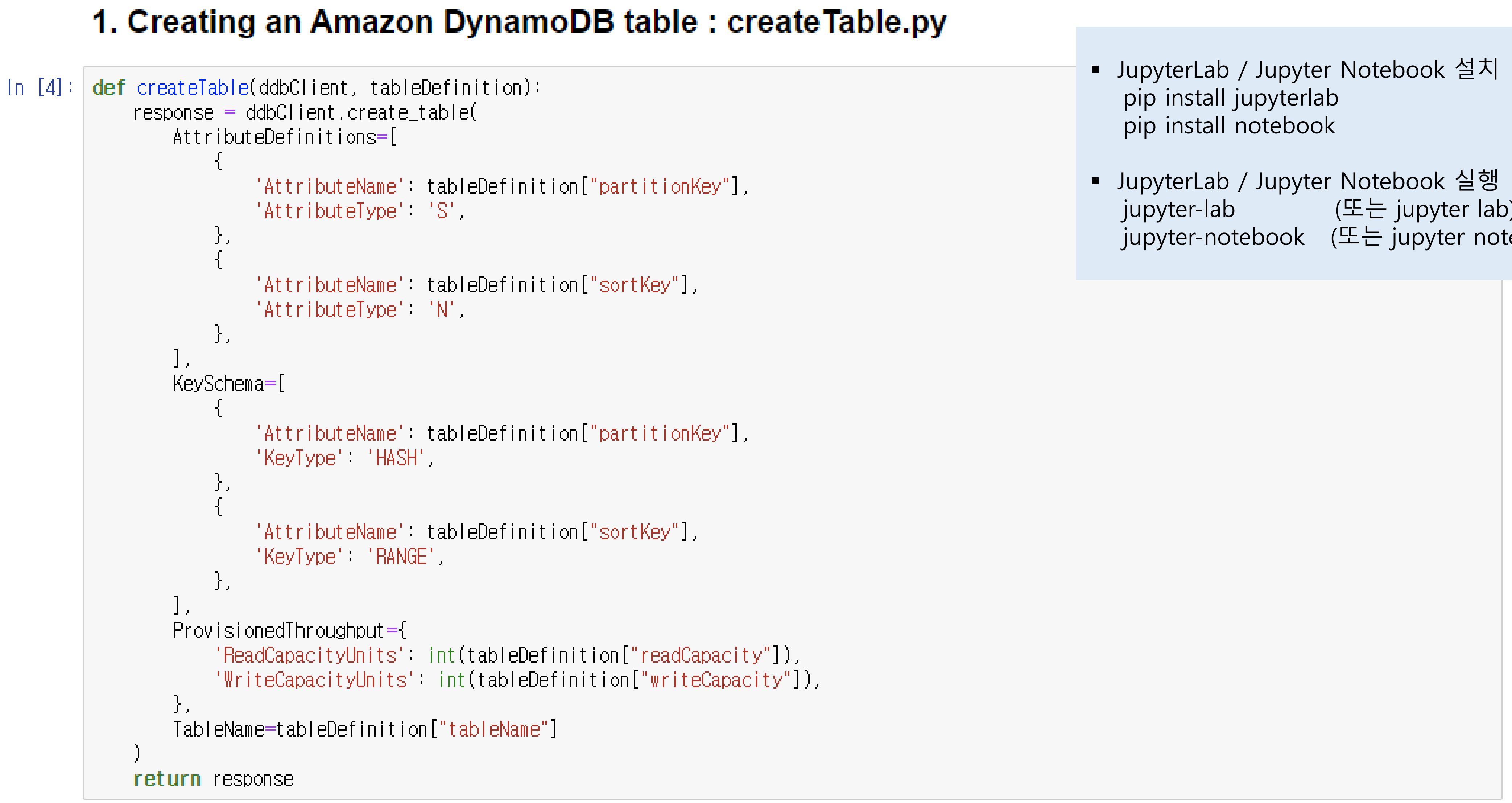


## config.ini

```
[DynamoDB]
tableName = Notes
partitionKey = UserId
sortKey = NoteId
readCapacity = 1
writeCapacity = 1
sourcenotes = ./labRepo/notes.json
queryUserId = student
pageSize = 3
queryNoteId = 5
notePrefix = The maximum item size in DynamoDB is
```

# DynamoDB

https://github.com/kgpark88/cloud-native/blob/main/DynamoDB/DevelopingOnAWS-DynomoDB.ipynb



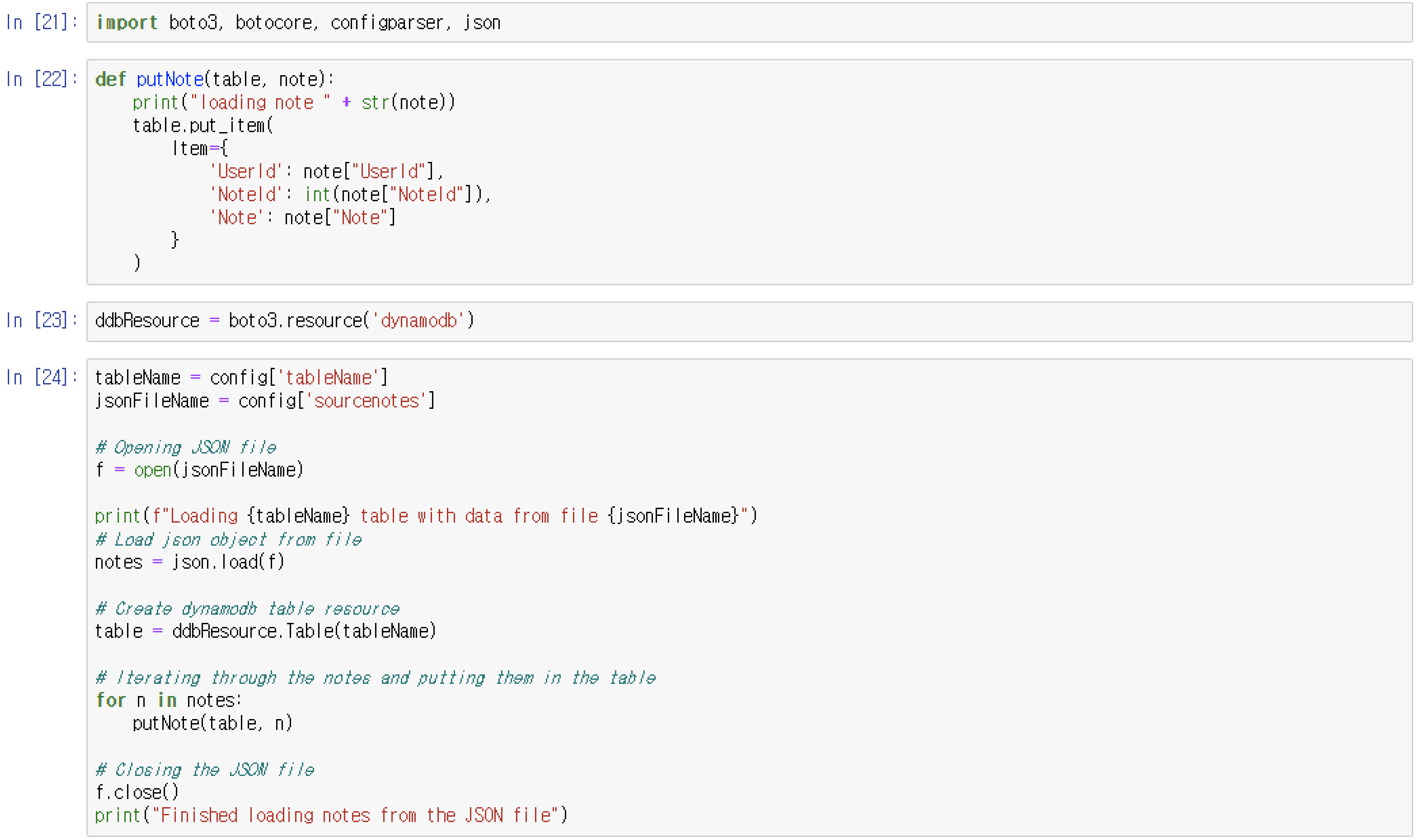
## 1. Creating an Amazon DynamoDB table : createTable.py

In [4]:
```
def createTable(ddbClient, tableDefinition):
    response = ddbClient.create_table(
        AttributeDefinitions=[
            {
                'AttributeName': tableDefinition["partitionKey"],
                'AttributeType': 'S',
            },
            {
                'AttributeName': tableDefinition["sortKey"],
                'AttributeType': 'N',
            },
        ],
        KeySchema=[
            {
                'AttributeName': tableDefinition["partitionKey"],
                'KeyType': 'HASH',
            },
            {
                'AttributeName': tableDefinition["sortKey"],
                'KeyType': 'RANGE',
            },
        ],
        ProvisionedThroughput={
            'ReadCapacityUnits': int(tableDefinition["readCapacity"]),
            'WriteCapacityUnits': int(tableDefinition["writeCapacity"]),
        },
        TableName=tableDefinition["tableName"]
    )
    return response
```

- JupyterLab / Jupyter Notebook 설치
  pip install jupyterlab
  pip install notebook

- JupyterLab / Jupyter Notebook 실행
  jupyter-lab (또는 jupyter lab)
  jupyter-notebook (또는 jupyter notebook)

## DynamoDB 2. 테이블에 데이터 로드 : loadData.py



In [21]:
```python
import boto3, botocore, configparser, json
```

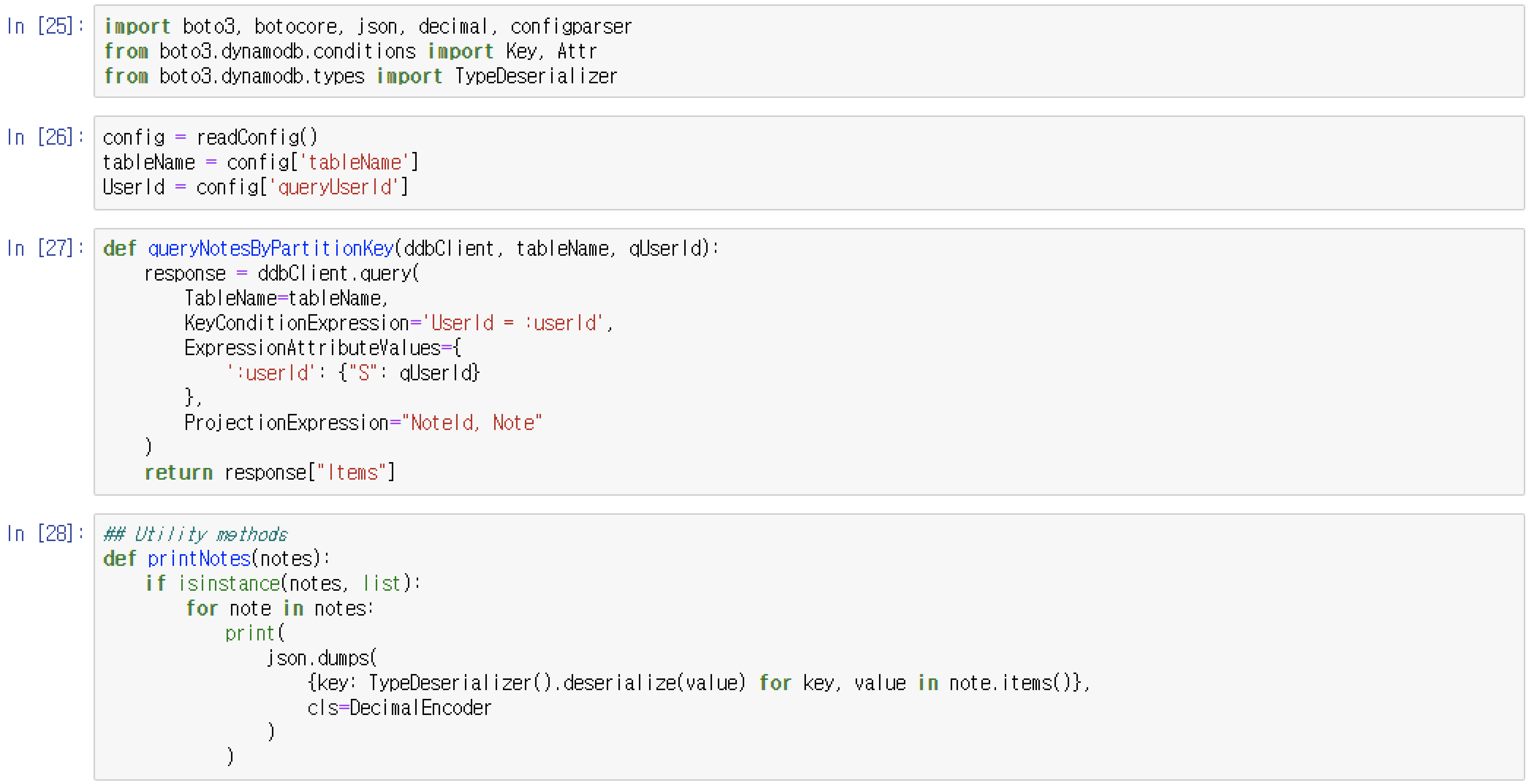
In [22]:
```python
def putNote(table, note):
    print("loading note " + str(note))
    table.put_item(
        Item={
            'UserId': note["UserId"],
            'NoteId': int(note["NoteId"]),
            'Note': note["Note"]
        }
    )
```

In [23]:
```python
ddbResource = boto3.resource('dynamodb')
```

In [24]:
```python
tableName = config['tableName']
jsonFileName = config['sourcenotes']

# Opening JSON file
f = open(jsonFileName)

print(f"Loading {tableName} table with data from file {jsonFileName}")
# Load json object from file
notes = json.load(f)

# Create dynamodb table resource
table = ddbResource.Table(tableName)

# Iterating through the notes and putting them in the table
for n in notes:
    putNote(table, n)

# Closing the JSON file
f.close()
print("Finished loading notes from the JSON file")
```

# DynamoDB

## 3. 파티션 키 및 프로젝션을 사용하여 데이터 쿼리 : loadData.py



In [25]:
```
import boto3, botocore, json, decimal, configparser
from boto3.dynamodb.conditions import Key, Attr
from boto3.dynamodb.types import TypeDeserializer
```

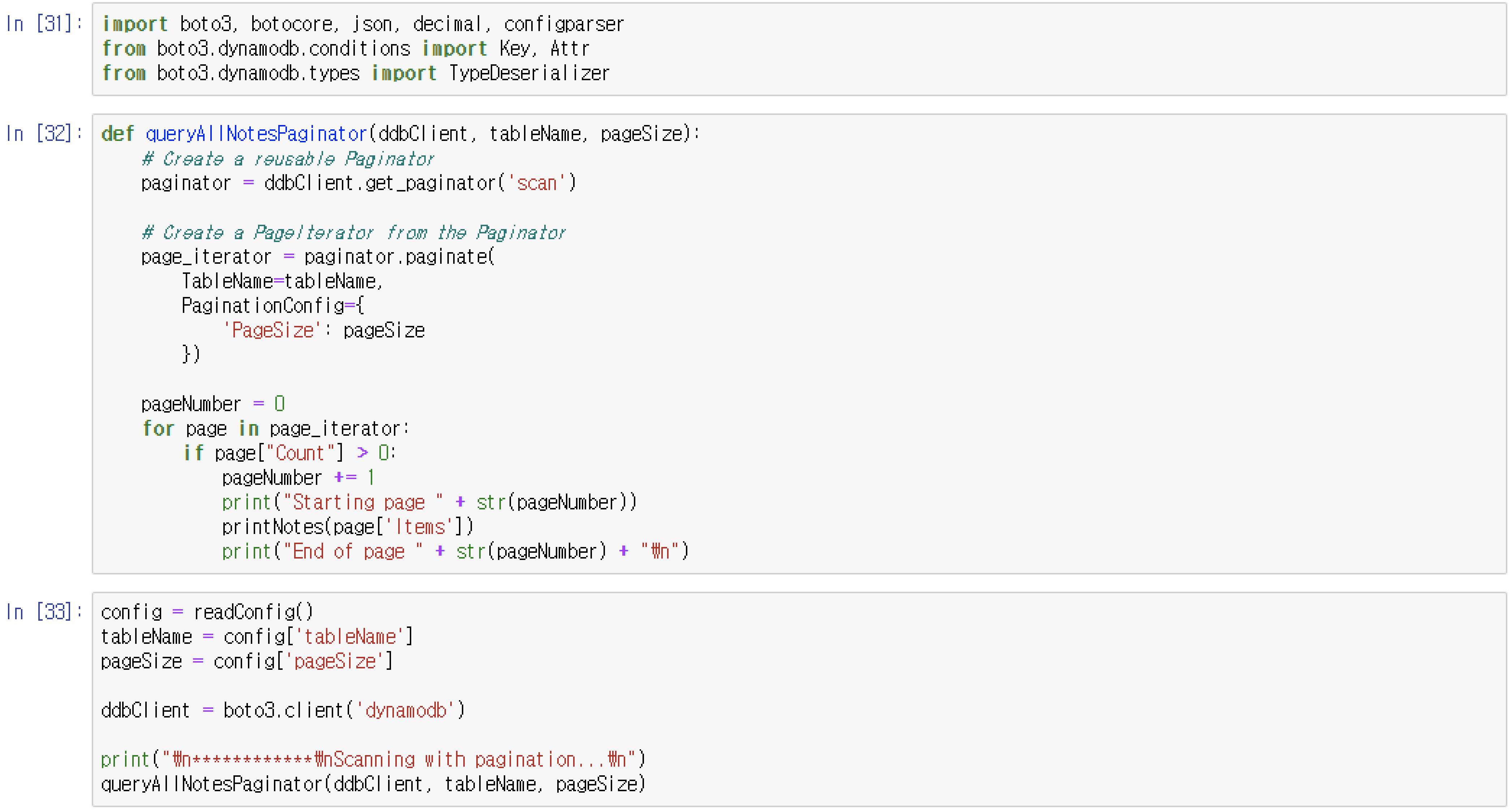
In [26]:
```
config = readConfig()
tableName = config['tableName']
UserId = config['queryUserId']
```

In [27]:
```
def queryNotesByPartitionKey(ddbClient, tableName, qUserId):
    response = ddbClient.query(
        TableName=tableName,
        KeyConditionExpression='UserId = :userId',
        ExpressionAttributeValues={
            ':userId': {"S": qUserId}
        },
        ProjectionExpression="NoteId, Note"
    )
    return response["Items"]
```

In [28]:
```
## Utility methods
def printNotes(notes):
    if isinstance(notes, list):
        for note in notes:
            print(
                json.dumps(
                    {key: TypeDeserializer().deserialize(value) for key, value in note.items()},
                    cls=DecimalEncoder
                )
            )
```

# DynamoDB

## 4: Paginator를 사용하여 테이블 스캔 : paginateData.py

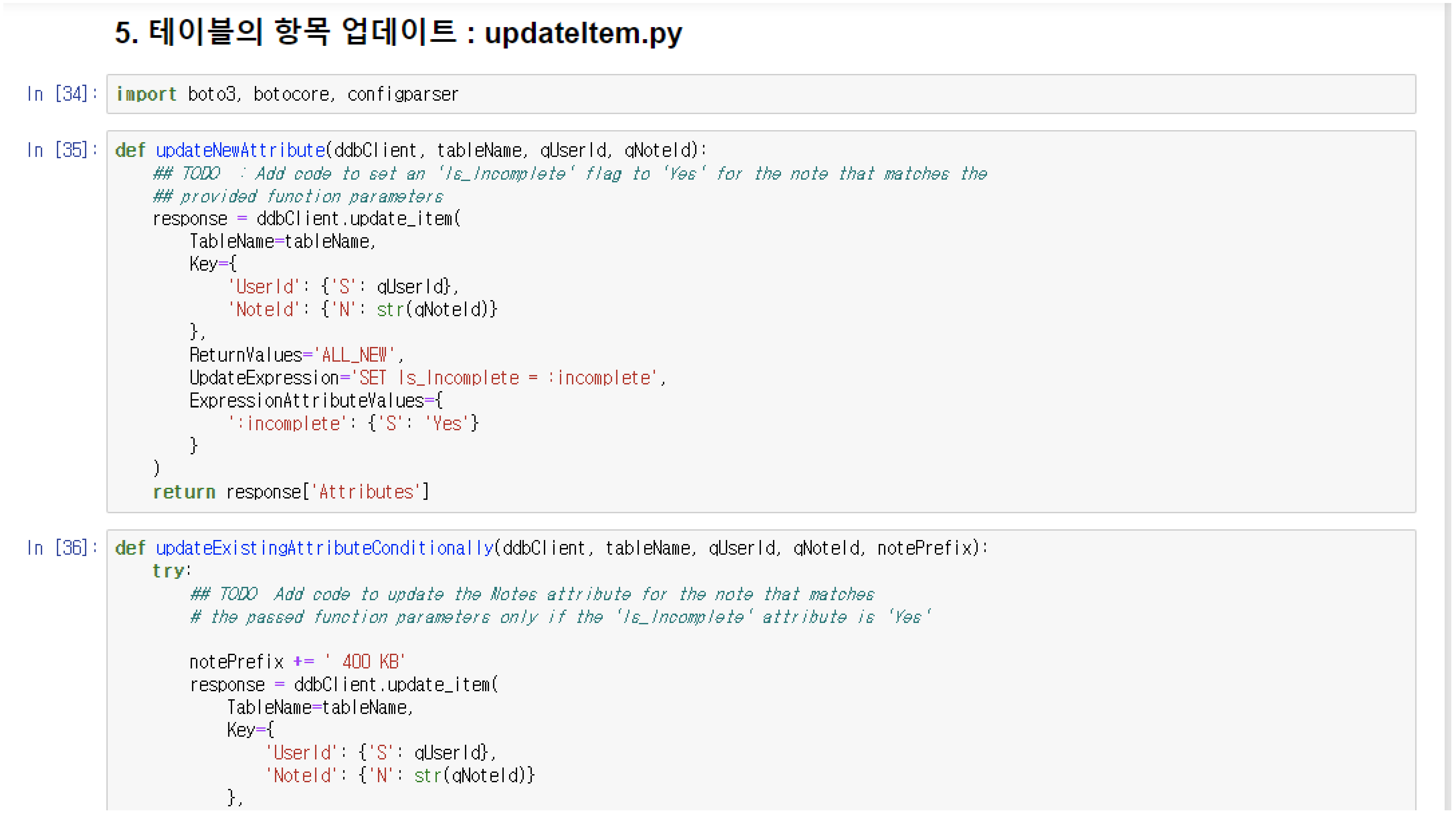


```
In [31]: import boto3, botocore, json, decimal, configparser
         from boto3.dynamodb.conditions import Key, Attr
         from boto3.dynamodb.types import TypeDeserializer
```

```
In [32]: def queryAllNotesPaginator(ddbClient, tableName, pageSize):
             # Create a reusable Paginator
             paginator = ddbClient.get_paginator('scan')

             # Create a PageIterator from the Paginator
             page_iterator = paginator.paginate(
                 TableName=tableName,
                 PaginationConfig={
                     'PageSize': pageSize
                 })

             pageNumber = 0
             for page in page_iterator:
                 if page["Count"] > 0:
                     pageNumber += 1
                     print("Starting page " + str(pageNumber))
                     printNotes(page['Items'])
                     print("End of page " + str(pageNumber) + "₩n")
```

```
In [33]: config = readConfig()
         tableName = config['tableName']
         pageSize = config['pageSize']

         ddbClient = boto3.client('dynamodb')

         print("₩n************₩nScanning with pagination...₩n")
         queryAllNotesPaginator(ddbClient, tableName, pageSize)
```

# DynamoDB



## 5. 테이블의 항목 업데이트 : updateItem.py

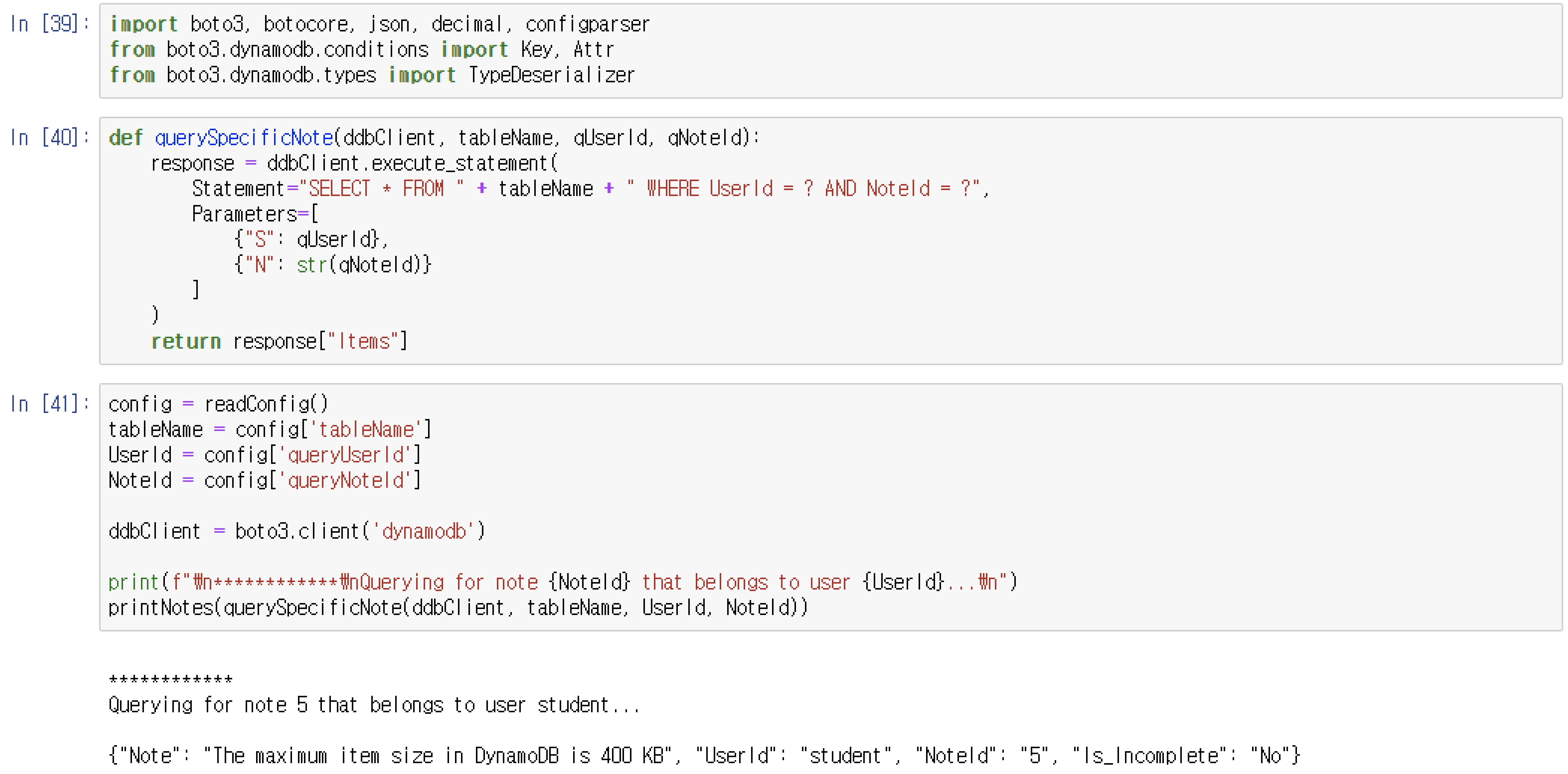
```
In [34]: import boto3, botocore, configparser
```

```
In [35]: def updateNewAttribute(ddbClient, tableName, qUserId, qNoteId):
             ## TODO  : Add code to set an 'Is_Incomplete' flag to 'Yes' for the note that matches the
             ## provided function parameters
             response = ddbClient.update_item(
                 TableName=tableName,
                 Key={
                     'UserId': {'S': qUserId},
                     'NoteId': {'N': str(qNoteId)}
                 },
                 ReturnValues='ALL_NEW',
                 UpdateExpression='SET Is_Incomplete = :incomplete',
                 ExpressionAttributeValues={
                     ':incomplete': {'S': 'Yes'}
                 }
             )
             return response['Attributes']
```

```
In [36]: def updateExistingAttributeConditionally(ddbClient, tableName, qUserId, qNoteId, notePrefix):
             try:
                 ## TODO  Add code to update the Notes attribute for the note that matches
                 # the passed function parameters only if the 'Is_Incomplete' attribute is 'Yes'

                 notePrefix += ' 400 KB'
                 response = ddbClient.update_item(
                     TableName=tableName,
                     Key={
                         'UserId': {'S': qUserId},
                         'NoteId': {'N': str(qNoteId)}
                     },
```

# DynamoDB

## 6. DynamoDB용 PartiQL(SQL 호환 쿼리 언어) 사용 : partiQL.py



```
In [39]: import boto3, botocore, json, decimal, configparser
         from boto3.dynamodb.conditions import Key, Attr
         from boto3.dynamodb.types import TypeDeserializer
```

```
In [40]: def querySpecificNote(ddbClient, tableName, qUserId, qNoteId):
             response = ddbClient.execute_statement(
                 Statement="SELECT * FROM " + tableName + " WHERE UserId = ? AND NoteId = ?",
                 Parameters=[
                     {"S": qUserId},
                     {"N": str(qNoteId)}
                 ]
             )
             return response["Items"]
```

```
In [41]: config = readConfig()
         tableName = config['tableName']
         UserId = config['queryUserId']
         NoteId = config['queryNoteId']

         ddbClient = boto3.client('dynamodb')

         print(f"₩n************₩nQuerying for note {NoteId} that belongs to user {UserId}...₩n")
         printNotes(querySpecificNote(ddbClient, tableName, UserId, NoteId))
```

```
************
Querying for note 5 that belongs to user student...

{"Note": "The maximum item size in DynamoDB is 400 KB", "UserId": "student", "NoteId": "5", "Is_Incomplete": "No"}
```

Thank you