Self Study - Python

1) OpenCV Basic
2) Boto3

OpenCV2 Basic
Method of learning

1. Watch Tutorial Videos
2. Solve some exercise questions
3. Google search - How to

Approaches

1. Numpy
2. OpenCV

What is OpenCV

- Related to Computer Vision.
- Result out meaningful data from video / image to set up data for deep learning.

What I learned

- All images can be represented by 0 - 255 matrix
- Contains 3 layers or channel (RGB)

```
#!/usr/bin/env python3

# Task: OPEN the dog_backpack.jpg image and display
import cv2
import numpy as np

def read_image():
    img = cv2.imread('somepath/dog_backpack.jpg')
    print(type(img))
    return img


def flip_image():
    img = cv2.imread('somepath/dog_backpack.jpg')
    flipped_img = cv2.flip(img, 0)
    return flipped_img


def draw_rectangle_dog_face():
    img = cv2.imread('somepath/dog_backpack.jpg')
    return cv2.rectangle(img, pt1=(200, 350), pt2=(600, 750), color=(0,0,255), thickne


# global for mouse event
```

```
img = cv2.imread('somepath/dog_backpack.jpg')


def draw_circle_mouse_event(event, x, y, flags, param):
    if event == cv2.EVENT_RBUTTONDOWN:
        cv2.circle(img, (x,y), 25, (0,0,255), 2)


def main():
    # Mouse event
    cv2.namedWindow(winname="dog_backpage")
    cv2.setMouseCallback("dog_backpage", draw_circle_mouse_event)
    # checked pass
    read_img = read_image()
    # checked pass
    flipped_img = flip_image()
    # checked pass
    draw_rec = draw_rectangle_dog_face()
    while True:
        cv2.imshow("dog_backpage", img)
        if cv2.waitKey(20) & 0xFF == 27:
            break
    cv2.destroyAllWindows()


if __name__ == "__main__":
    main()
```

Blending Same Size Image

```
import cv2
import matplotlib.pyplot as plt
# blending images is nothing more than just adding the images together.
# addWeighted function is being used for blending image
# new_pixel = Alpha * pixel_2 + Beta x Pixel_2 + Y (optional gamma)
img1 = cv2.imread("somepath/dog_backpack.png")
# convert into RGB from BGRimg1 = cv2.cvtColor(img1, cv2.COLOR_BGR2RGB)
img2 = cv2.imread("somepath/watermark_no_copy.png")
# convert into RGB from BGRimg2 = cv2.cvtColor(img2, cv2.COLOR_BGR2RGB)

# check the shape (size)print(img1.shape)
print(img2.shape)

# the size are different need to re-format the size of the image
# BLENDING IMAGES OF THE SAME SIZEimg1 = cv2.resize(img1, (1200,1200))
img2 = cv2.resize(img2, (1200, 1200))

# addWeighted function (formula: Alpha * pixel_2 + Beta x Pixel_2 + Y (optional gamma)
# @https://docs.opencv.org/3.4/d5/dc4/tutorial_adding_images.html
blended_image = cv2.addWeighted(src1=img1, alpha=0.5, src2=img2, beta=0.5, gamma=0)
```

```
while True:
    # simply change the src image
    # cv2.imshow("dog_backpage", img1)
    # cv2.imshow("water-mark", img2)
    cv2.imshow('blended-image', blended_image)
    if cv2.waitKey(20) & 0xFF == 27:
        breakcv2.destroyAllWindows()
```

Result of blending
dog backpage image + do not copy image

Not even close to any kind of image processing and video processing + deep learning to use processed data as an input for deep learning.

Currently studying about threshold to set the matrix value to 0 or max
Will be continue...

Boto3
Method of learning

1. Watch Tutorial Videos
2. Try out myself

3. Google search - How to

Create S3 Bucket using python script

```
def create_s3_bucket():
    aws_keys = retrieve_properties()
    access_key = aws_keys[0].split("=")[1]
    serect_key = aws_keys[1].split("=")[1]
    client = boto3.client("s3", aws_access_key_id=access_key, aws_secret_access_key=se
    # @https://stackoverflow.com/questions/31092056/how-to-create-a-s3-bucket-using-bo
    """ :type : pyboto3.s3 """
    client.create_bucket(Bucket='my-python-s3', CreateBucketConfiguration={'LocationCo
```

Function to list out all existing S3 buckets

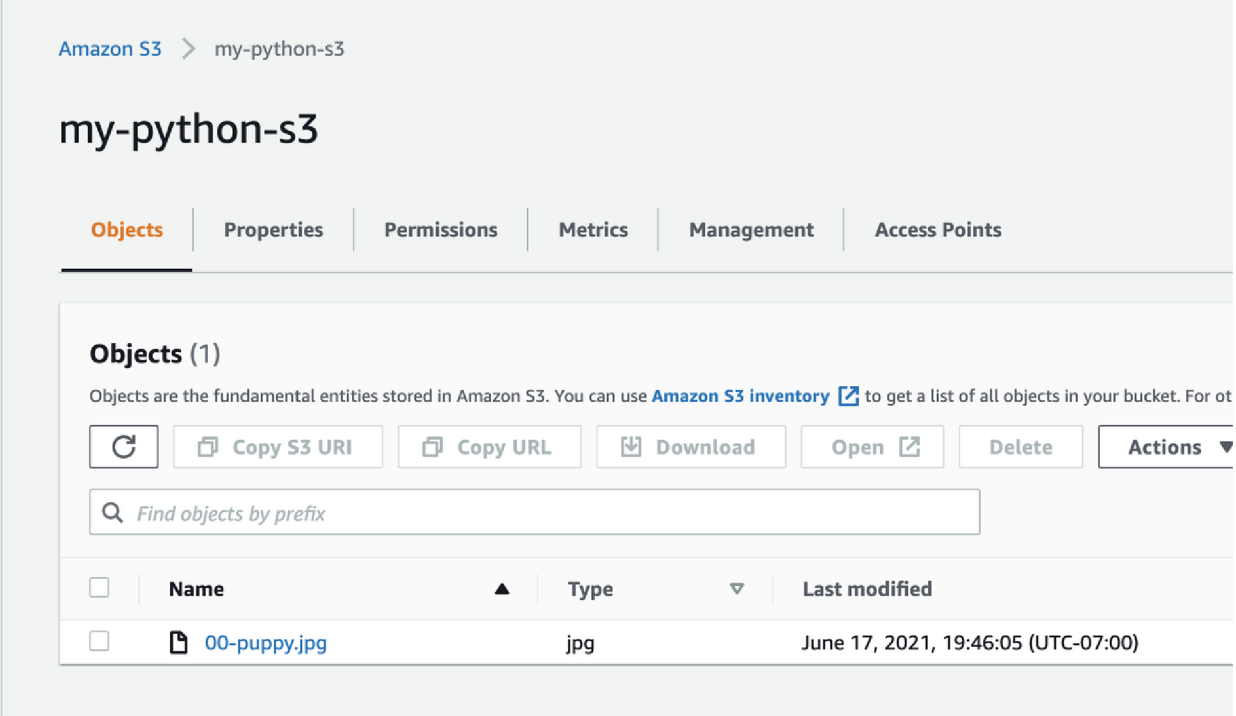
```
def list_s3_bucket():
    aws_keys = retrieve_properties()
    access_key = aws_keys[0].split("=")[1]
    serect_key = aws_keys[1].split("=")[1]
    client = boto3.client("s3", aws_access_key_id=access_key, aws_secret_access_key=se
    """ :type : pyboto3.s3 """
    response = client.list_buckets()
    pprint.pprint(response.get('Buckets'))
```

Upload file into S3 buckets

```
def upload_data_to_s3_bucket(file_name, bucket, object_name=None, args=None):
    """
    :param file_name: name of file on local
    :param bucket: bucket name
    :param object_name: name of file to be in S3 (None) then using local file_name
    :param args: custom args
    :return:
    """
    # set object_name to local file_name if not passed as arg
    if object_name == None:
        object_name = file_name
    aws_keys = retrieve_properties()
    access_key = aws_keys[0].split("=")[1]
    serect_key = aws_keys[1].split("=")[1]
    client = boto3.client("s3", aws_access_key_id=access_key, aws_secret_access_key=se
    """ :type : pyboto3.s3 """
    response = client.upload_file(file_name, bucket, object_name, ExtraArgs= args)
    # respones should be None otherwise exception
    pprint.pprint(response)
```

```
def main():
    # create bucket Checked
    # create_s3_bucket()
```

```
    # list out existing buckets Checked
    # list_s3_bucket() Checked
    # upload files into S3 bucket Checked
    target_file_path = '/Users/junpark/Downloads/Computer-Vision-with-Python/DATA/00-p
    # to make it dynamic use uuid + split("/") and use last data of the list
    upload_data_to_s3_bucket(target_file_path, 'my-python-s3', "00-puppy.jpg")
```



Notice that the file had been stored in the S3 bucket.

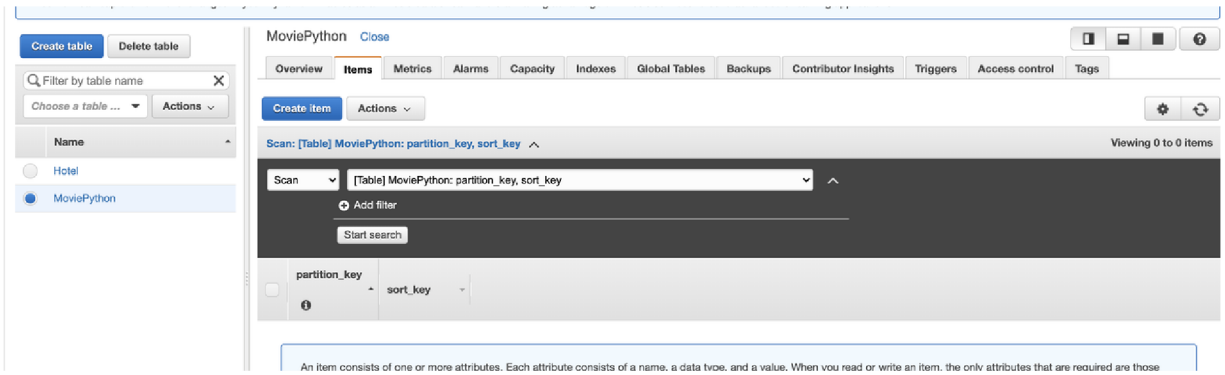
Download from S3

```
def download_file_from_s3():
    aws_keys = retrieve_properties()
    access_key = aws_keys[0].split("=")[1]
    serect_key = aws_keys[1].split("=")[1]
    client = boto3.client("s3", aws_access_key_id=access_key, aws_secret_access_key=se
    """ :type : pyboto3.s3 """
    target_bucket = "my-python-s3"
    # print(client.get_object(Bucket=target_bucket, Key="00-puppy.jpg"))
    client.download_file(Bucket=target_bucket, Key="00-puppy.jpg", Filename="download_
```

DynamoDB

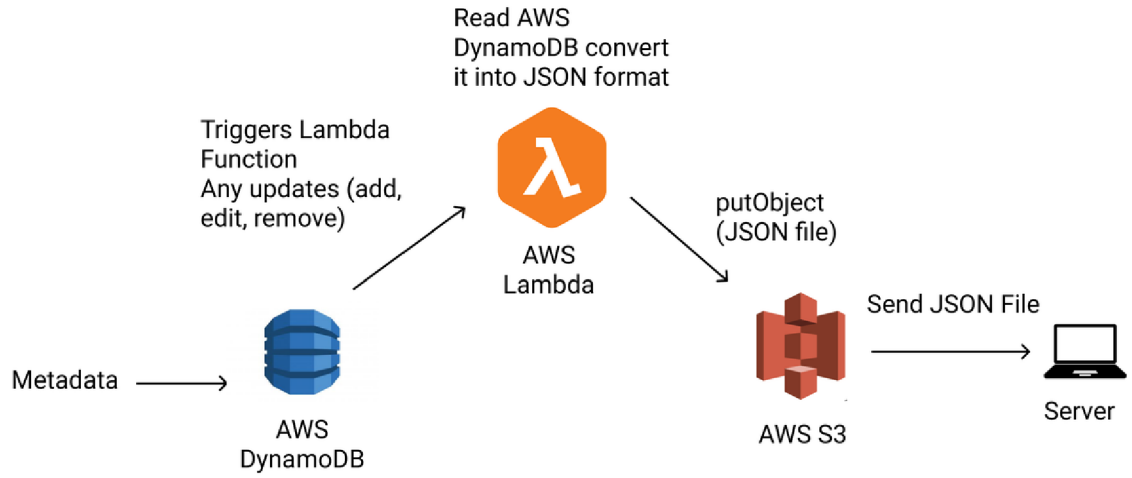
```
def create_dynamodb_table():
    aws_keys = retrieve_properties()
    access_key = aws_keys[0].split("=")[1]
    serect_key = aws_keys[1].split("=")[1]
    boto3_dynamodb = boto3.resource('dynamodb', region_name='us-west-2', aws_access_ke
```

```
                        aws_secret_access_key=serect_key)
    # @https://stackoverflow.com/questions/31092056/how-to-create-a-s3-bucket-using-bo
    """ :type : pyboto3.dynamodb """
    print(list(boto3_dynamodb.tables.all()))
    # param for create table
    table_name = "MoviePython"
    try:
        params = {
            'TableName': table_name,
            # list of dictionary
            'KeySchema' :[
                {'AttributeName': 'partition_key', 'KeyType': 'HASH'},
                {'AttributeName': 'sort_key', 'KeyType': 'RANGE'}
            ],
            'AttributeDefinitions': [
                {'AttributeName': 'partition_key', 'AttributeType': 'N'},
                {'AttributeName': 'sort_key', 'AttributeType': 'N'}
            ],
            'ProvisionedThroughput': {
                'ReadCapacityUnits': 10,
                'WriteCapacityUnits': 10
            }
        }
        table = boto3_dynamodb.create_table(**params)
        print(f"Creating table...")
        table.wait_until_exists()
        return table
    except ClientError:
        print("Table already exist")
```



By running the code on top, I created a Dynamodb table.

Ultimate Goal Design



Constraints:

- BI team cannot directly query DynamoDB, they should only query S3
- DynamoDB → S3 → BI report

Reasons for querying S3 instead of directly use the database.

S3
S3 designed to handle an extremely high number of traffic requests, especially requests for different items

DynamoDB
DynamoDB designed for relatively small items (4KB or less). Each operation is definitely faster than S3 but S3 offers better scalability.

Huge volumes of traffic, it can overwhelmed for a while.

POC (Proof of Concept)

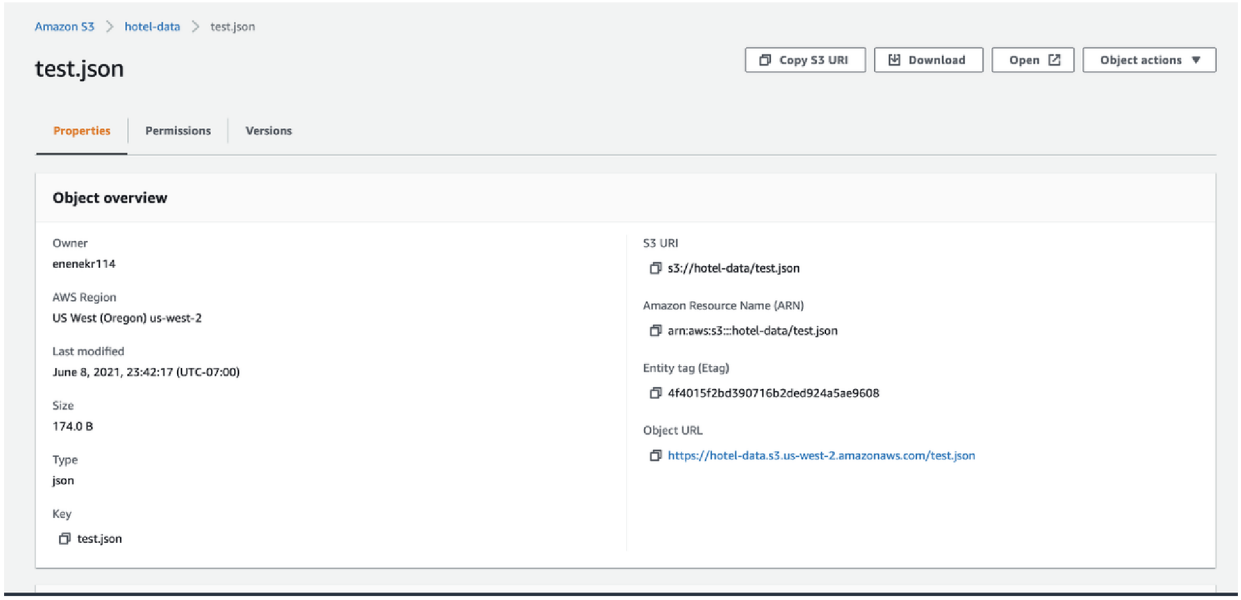
Lambda Function (using javascript) Nothing but works as pipeline in my opinion (pipeline between DynamoDB and S3)

```
const AWS = require("aws-sdk");

// config the dynamodb to get access and read data
const aws_dynamoDB_config = {
    "region": "us-west-2",
    "endpoint": "http://dynamodb.us-west-2.amazonaws.com",
    "accessKeyId": "nope",
    "secretAccessKey": "nope"
};
const s3 = new AWS.S3({
    "accessKeyId": "nope",
    "secretAccessKey": "nope"
});
AWS.config.update(aws_dynamoDB_config);
```

```
exports.handler = async (event) => {
    // grab all data from dynamodb
    let docClient = new AWS.DynamoDB.DocumentClient();
    let params = {
        TableName: "Hotel"
    };
    const scanResult = [];
    let items;
    do{
        items =  await docClient.scan(params).promise();
        items.Items.forEach((item) => scanResult.push(item));
        params.ExclusiveStartKey  = items.LastEvaluatedKey;
    }while(typeof items.LastEvaluatedKey !== "undefined");
    console.log(scanResult);
    let dataToObj = {
        "scanResult": scanResult
    };
    console.log(dataToObj)
    // S3 param
    let param_s3 = {
        Bucket: "hotel-data",
        Key: 'test.json',
        Body: JSON.stringify(dataToObj),
        ContentType: 'application/json'
    }
    await s3.putObject(param_s3).promise();
    console.log("Successfully added in S3");
    return scanResult;
};
```

Updated JSON file in S3



```
app.get("/fetchAllHotel", async (req, res) => {
  // just going to read json file from s3 to test it works
  const getParams = {
    Bucket: "hotel-data",
    Key: "test.json",
  };
  let fetchTestJson = await s3.getObject(getParams).promise();
  console.log(fetchTestJson.Body.toString());
  let jsonData = JSON.parse(fetchTestJson.Body.toString());
  console.log(jsonData);
  res.status(200).json(jsonData);
});
```

public > js > JS fetchAllHotel.js > createTable

```
1   console.log("connected");
2   fetchAllHotel();
3   async function fetchAllHotel() {
4     try {
5       const All_Hotel_Data = await fetch("/fetchAllHotel");
6       const hotel_datas = await All_Hotel_Data.json();
7       // console.log(hotel_datas);
8       createTable(hotel_datas);
9     } catch (error) {
10      console.error(error);
11    }
12  }
13
14  function createTable(hotel_data) {
15    // create tables
16    const hotelArray = hotel_data.scanResult;
17    const unOrderedList = document.createElement("ul");
18    for (let i = 0; i < hotelArray.length; i++) {
19      let liTag = document.createElement("li");
20      liTag.textContent = `Hotel_id: ${hotelArray[i].hotel_id} Hotel_name: $
21      unOrderedList.appendChild(liTag);
22    }
23    // grab the parent node
24    const parentNode = document.querySelector(".table_all_hotel");
25    parentNode.appendChild(unOrderedList);
26  }
```

Since I learned flask, I can re-write these POC (proof of concept) in python.