Python 데이터 분석

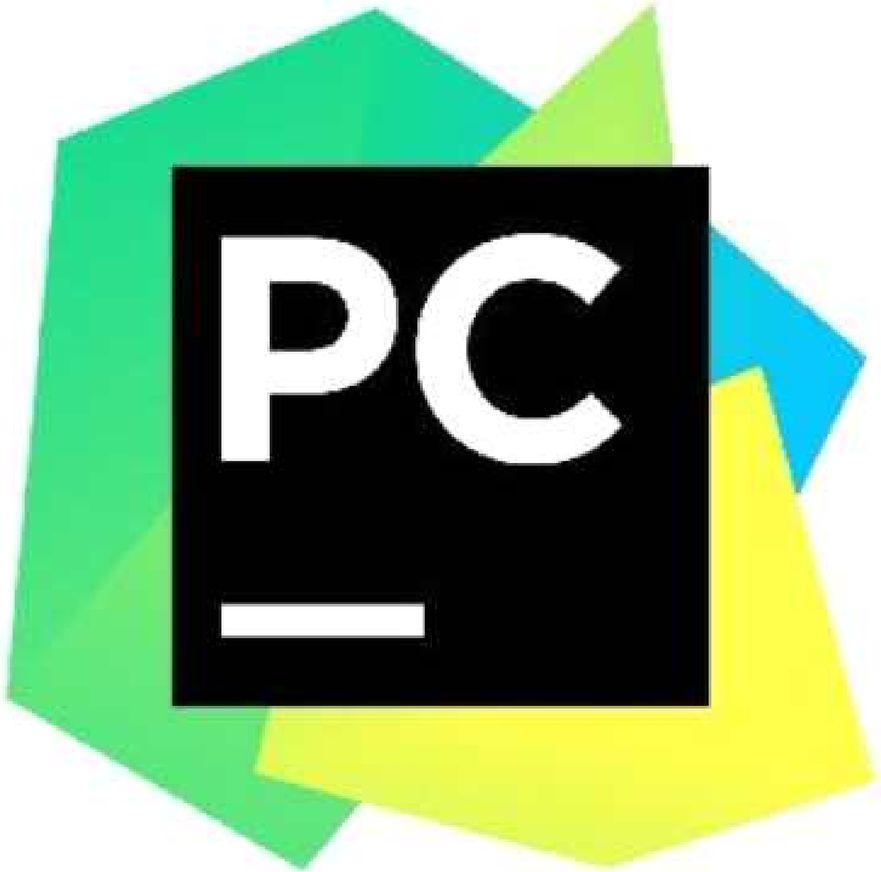
OpenCV 를 이용한 이미지 처리

한이진 Han Yi Jin

사용자 환경

PyCharm IDE, Python 3.9

OpenCV **설치**
pip install opencv-pythion



데 이 터 분 석 과 정

문 제 해 결 방 법

데이터 수집 및 분석하기

특정 수식 이미지을 OpenCV로 처리
이미지 픽셀값, (RGB) 배열 값 분석하여 특정
유의미한 패턴을 확인

데이터 정제하기

필요한 이미지 추출
테스트 데이터 만들기

KNN 모델 학습

정제된 데이터 학습
학습된 KNN 모델로 테스트

테스트 결과 검증

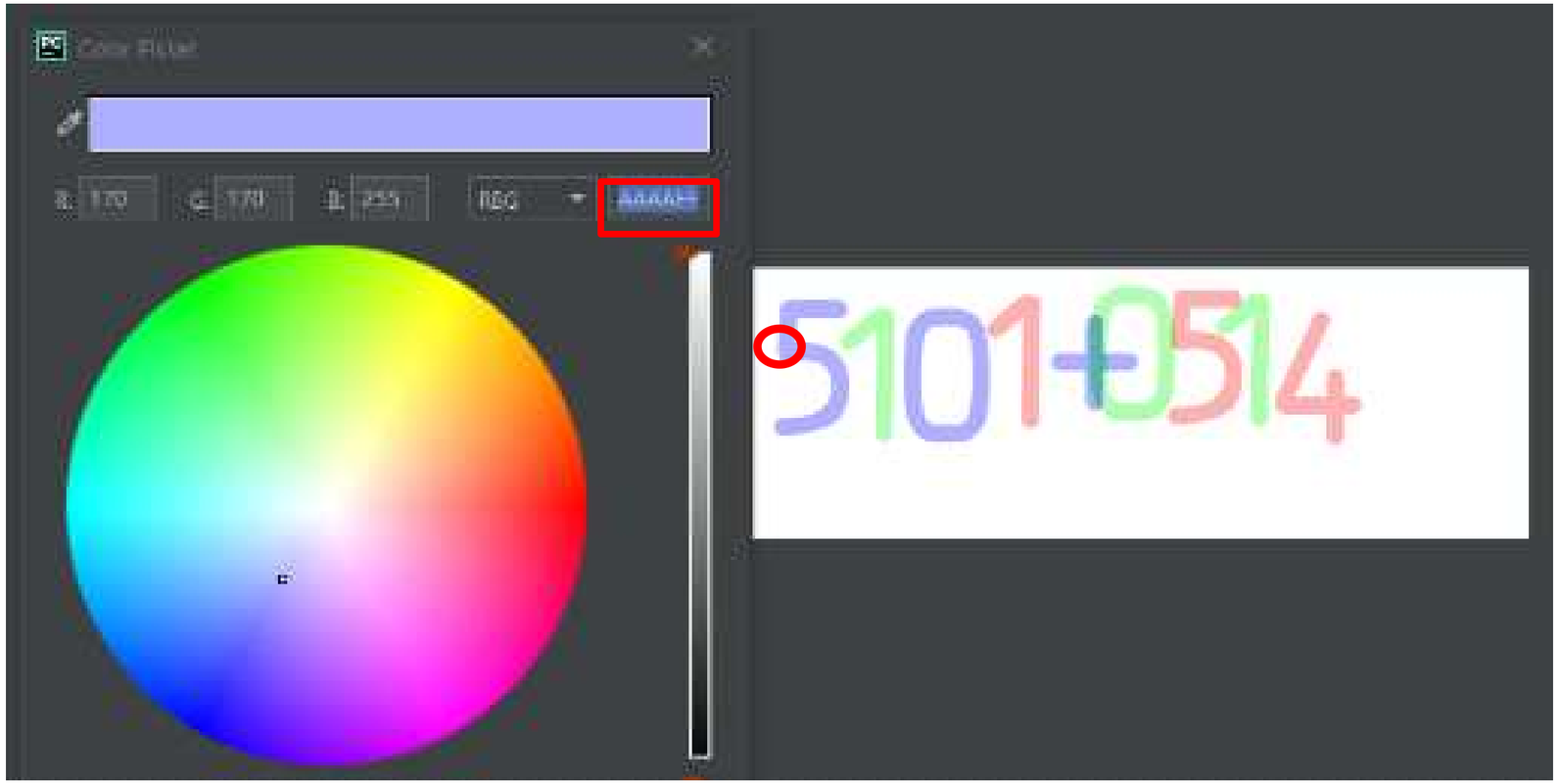
OpenCV

- 실시간 이미지/ 영상 처리에 사용하는 오픈 소스 라이브러리
- python, C++, Java와 같은 다양한 개발 환경을 지원
- TensorFlow , PyTorch의 딥러닝 프레임워크를 지원
- 물체 인식, 안면인식, 제스처 인식, 모바일 로보틱스등 기술에 응용되고 있다

KNN 알고리즘

- 지도학습에 한 종류로 거리기반 (k-최근접 이웃) 분류분석 모델이다
- 데이터로부터 거리가 가까운 'K'개의 다른 데이터의 레이블을 참조하여 분류하는 알고리즘
- 이미지 처리, 영상에서 글자 인식과 얼굴인식, 상품 추천에 대한 개별 선호 예측 등 많은 분야에서 응용되고 있다.
- 물체 인식, 안면인식, 제스처 인식, 모바일 로보틱스등 기술에 응용되고 있다
- 알고리즘이 간단하여 구현하기가 쉽다는 장점이 있다.
- 모델을 생성하지 않아 특징과 클래스 간 관계를 이해하는 데 제한적이며 데이터가 많아지면 분류 단계가 느려지는 단점이 있다.

데 이 터 수 집 및 분 석



색상 추출기를 활용하여 각 문자의 색상을 분석

- 파랑색 : RGB 에서 AAAAFF 픽셀값 (170, 170, 255)
- 초록색 : RGB 에서 AAFFAA 픽셀값 (170, 255, 170)
- 빨간색 : RGB 에서 FFAAAA 픽셀값 (255, 170, 170)

구성되어 있음을 파악한다

데 이 터 정 제 하 기


```
import cv2
img = cv2.imread('images/d2.png', cv2.IMREAD_COLOR)
```

OpenCV는 이미지(영상) 데이터를 numpy.ndarry으로 표현한다

cv2.imread(filename,flag)

flag: 이미지 파일을 읽는 방법(옵션)

- cv2.IMREAD_COLOR : 이미지를 color로 읽어 (행(y축), 열(x축), (RGB)) 3차원 배열로 반환된다.
  ex) img.shape => (206, 207, 3) img[: , : , 1] => Green 색상 , img[: , : , 2] => Blue 색상

- cv2.IMREAD_GRAYSCALE : 이미지를 Grayscale로 읽어 (행(y축), 열(x축) 2차원 행렬 배열로 반환된다.

```
print(img[50,50])
>img[50,500]=[255,255,255] # 이미지[y축,x축]의 픽셀값 출력
```

데이터 정제하기

픽셀(pixel,화소)는 디지털이미지를 구성하는 색상이나 밝기를 표시하는 값이다

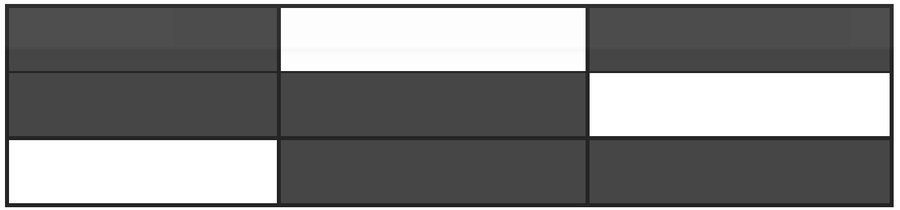
흑백의 이미지의 경우 각 픽셀의 밝기를 지정하여 이미지를 형성, 각각의 픽셀은 그 지점의 밝기를 나타낸다.픽셀값은 $2^8 = 256$ 즉, 0과 255 사이의 값들 중 하나의 값이 가진다.

0은 가장 어두운 검은 색을 나타내고, 255는 가장 밝은 상태 흰색을 나타낸다.

컬러 이미지의 경우 화소의 색을 지정하여 이미지를 형성, Red, Green, Blue 3원색(color channel)을 조합하여 표현한다

각 픽셀은3원색 각각의 밝기를 나타내는 값(R,G,B 각 1개씩 3개의 값)을 가진다. 0과 255사이의 값들 중 하나의 값을 가지며, 0은 검은색, 255는 원색을 나타낸다.

0	255	0
0	0	255
255	0	0



(255, 0, 0)	(0, 0, 0)	(0, 255, 0)
(0, 255, 255)	(0, 0, 255)	(255, 0, 255)
(255, 127, 0)	(255, 255, 255)	(255, 255, 0)

데이터 정제하기 utis.py

```
import cv2
img = cv2.imread('images/d2.png', cv2.IMREAD_COLOR)

GREEN = 0
BLUE = 1
RED = 2

def getcolors(img, color):#특정색상을 지닌 단어를 이미지로 추출
    other_1 = (color + 1) % 3 #추출하고자하는단어의 색상외의변수
    other_2 = (color + 2) % 3
    # 불리언 인덱싱
    indexes = img[:, :, other_1] == 255
    img[indexes] = [0, 0, 0]

    indexes = img[:, :, other_2] == 255 #지정 색상외 변수의 픽셀값변경 [0,0,0]
    img[indexes] = [0, 0, 0]                즉, 검은색으로 변경한다

    indexes = img[:, :, color] < 170
    img[indexes] = [0, 0, 0]

    indexes = img[:, :, color] != 0  #추출하고자 하는 색상의 단어
    img[indexes] = [255, 255, 255][255,255,255] 즉, 흰색으로 변경한다

    return img
```

ex) BLUE 색상의 단어를 추출
other_1=GREEN
Other_2=RED

데이터 정제하기 utlis.py

```
def extract_chars(img): #전체 이미지에서 왼쪽부터 이미지를 윤관석 기준으로 추출
    chars = []
    colors = [BLUE,GREEN,RED]

    for color in colors:
        imgs = getcolors(img.copy(),color)

        gray_imgs = cv2.cvtColor(imgs,cv2.COLOR_BGR2GRAY)
        ret, thre_imgs = cv2.threshold(gray_imgs,127,255,cv2.THRESH_BINARY)

        contours, _ =


cv2.findContours(thre_imgs,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)

        for contour in contours:
            area = cv2.contourArea(contour)

            if area > 50:
                x,y,width,height = cv2.boundingRect(contour)
                roi = gray_imgs[y:y+height, x:x+width]
                chars.append((x,roi))

    chars = sorted(chars, key = lambda char:char[0])
    return chars
```

데이터 정제하기 utlis.py
def extract_chars(img)

```
gray_imgs = cv2.cvtColor(imgs,cv2.COLOR_BGR2GRAY)#흑백처리

#윤곽선(findContours)를 검출하는 주된 요소는 하얀색의 객체를 검출한다
#배경은 검은색이며 검출하려는 물체는 하얀색의 성질을 띄게끔 변형한다.

ret, thre_imgs = cv2.threshold(gray_imgs,127,255,cv2.THRESH_BINARY)

# (원본이미지, 임계값, 임계값 이상일 경우 바꿀 최대값(흰색=255로
지정),THRESH_BINARY)
#THRESH_BINARY는 픽셀값이 임계값보다 클 경우 최대값으로 작을 경우 0(검은색)으로
이진화
# 127이상은 255으로 127미만은 0으로 처리

contours, _ = cv2.findContours(thre_imgs,cv2.RETR_EXTERNAL,
                               cv2.CHAIN_APPROX_SIMPLE)


# cv2.findContours(이진화 이미지, 검색 방법, 근사화 방법)를 이용하여 이진화 이미지에서 윤곽선(컨투어)를 검출
# 반환값으로 윤곽선, 계층 구조를 반환
# 윤곽선은 Numpy 구조의 배열로 검출된 윤곽선의 지점들이 담겨있다.


# 검색방법: cv2.RETR_EXTERNAL : 외곽 윤곽선만 검출하며, 계층 구조를 구성하지 않습니다.
# 근사화 방법: cv2.CHAIN_APPROX_SIMPLE : 윤곽점들 단순화 수평, 수직 및 대각선 요소를 압축하고 끝점만 남겨 둡니다.
```

데이터 정제하기 utis.py

def extract_chars(img)

```
for contour in contours:

    #cv2.contourArea() 외곽선이 감싸는 영역의 면적을 반환한다.
        area = cv2.contourArea(contour)

        if area > 50:#추출된 이미지 크기가 50인 경우만 데이터로 파악 (점 요소 제거)

    #cv2.boundingRect(외곽선 좌표) 주어진 점을 감싸는 최소 크기
                                        사각형(바운딩 박스)를 반환합니다.
    #반환값: 사각형 정보. (x, y, w, h) 튜플
            x,y,width,height = cv2.boundingRect(contour)

            roi = gray_imgs[y:y+height, x:x+width] # 해당 단어영역만추출
            chars.append((x,roi))

chars = sorted(chars, key = lambda char:char[0])
return chars  # x축 기준으로 단어가 나열되어 하나의 수식형태로
              저장
                  ex) 5101+0514
```

테스트데이터만들기 make_train_data.py

```
def resized20(img):
    resized = cv2.resize(img,(20,20))     #동일한 크기(20 * 20)의 이미지로 변환
    return resized.reshape(-1,400).astype(np.float32)
                                  #KNN알고리즘에 적용하기 위해 1차원 배열로 나열 재구조화
for char in chars:
    cv2.imshow('char[1]',char[1])
    input = cv2.waitKey(0)
    resized= utils.resized20(char[1])
```

#추출되어 보여지는 이미지 해당 숫자를 키보드로 입력
0~9까지 입력된 숫자를 폴더명으로 지정한다.

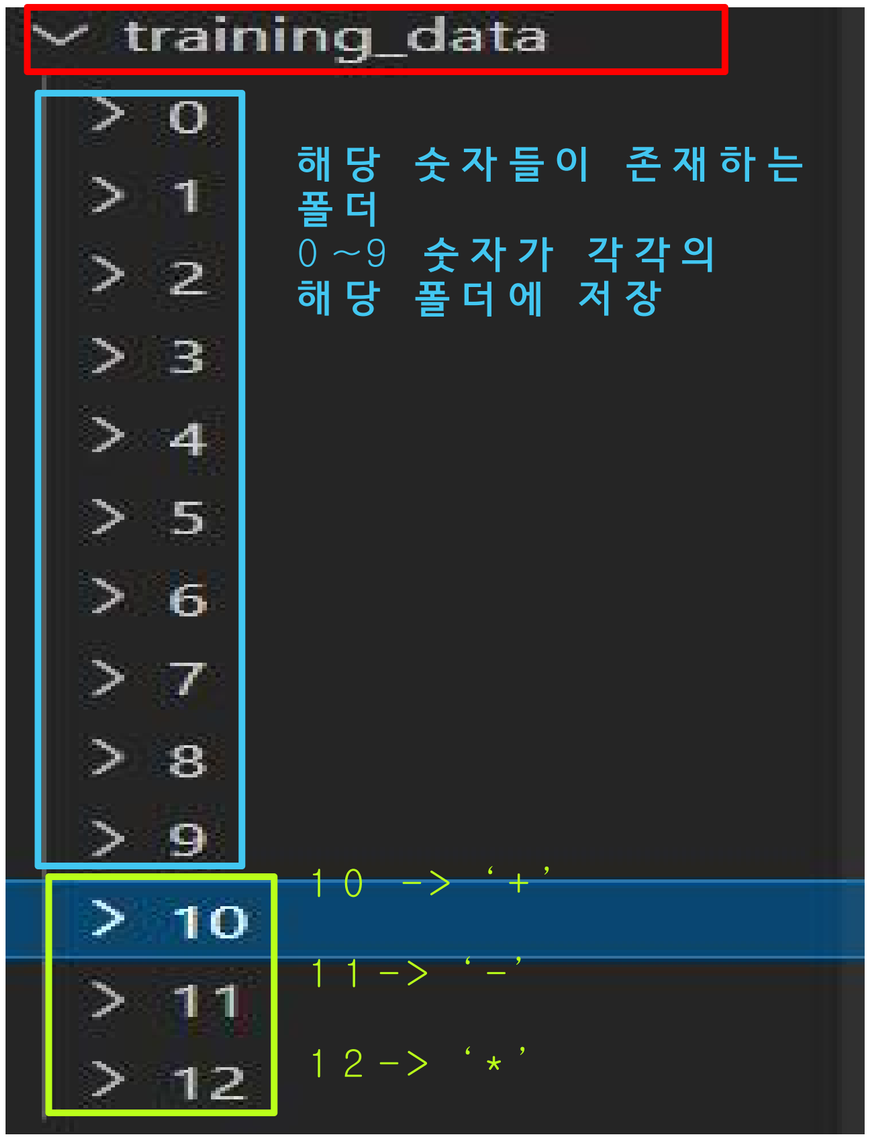
```
    if input >= 48 and input <= 57: #유니코드 값 48=0, 57=9
        name = str(input - 48)#입력된 숫자가 파일명으로 지정된다

        if not os.path.isdir('./training_data'):
            os.makedirs('./training_data')    #해당 디렉토리가 존재하지 않을 시 생성하여 저장
        if not os.path.isdir('./training_data/' + name):
            os.makedirs('./training_data/' + name)
        file_count = len(next(os.walk('./training_data/' + name + '/'))[2])
        cv2.imwrite('./training_data/' + str(input - 48) + '/' +
                    str(file_count + 1) + '.png', resized)
#수식 + = a / - = b / * = c로 입력받는다.
    elif input == ord('a') or input == ord('b') or input == ord('c'):

        #파이썬 내장함수 ord('a')는 문자의 유니코드 값97을 돌려주는 함수이다.
        name = str(input - ord('a') + 10)
        if not os.path.isdir('./training_data'):
            os.makedirs('./training_data')
        if not os.path.isdir('./training_data/' + name):
            os.makedirs('./training_data/' + name)
        file_count = len(next(os.walk('./training_data/' + name + '/'))[2])
        cv2.imwrite('./training_data/' + name + '/' +
                    str(file_count + 1) + '.png', resized)
```

테스트데이터만들기

정제된 테스트 데이터가 저장됨



KNN모델학습 Run.py

```
filenames= list(range(0,13))
train =[]  #테스트할 데이터 배열
train_labels=[]  # 테스트 데이터가 참조하는 레이블 배열
                                    이미지 데이터가 해당하는 숫자배열

for filename in filenames:
    path ='./training_data/'+str(filename)+'/'
    print(path)
    file_count = len(next(os.walk(path))[2])
    for i in range(1,file_count+1):
        img= cv2.imread(path+str(i)+'.png')
        gray= cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

        train.append(gray)  #테스트할 데이터 추가
        train_labels.append(filename) #각 이미지 데이터(숫자) 레이블 정보 추가

x= np.array(train)
train=x[:,:].reshape(-1,400).astype(np.float32)
#2차원 배열인 이미지 (20*20) => 길이가 400인 1차원 배열로 재배열
#KNN알고리즘 적용을 위한 데이터 1차원으로 재배열

train_labels=np.array(train_labels)[:,np.newaxis]

np.savez("trained.npz", train=train, train_labels=train_labels) #학습된 데이터
                                                                정보를 파일로 저장
```

테스트데이터검증 Run.py

```
filename= 'trained.npz' # 학습된 KNN모델 불러오기

with np.load(filename) as data: #file open close 자동
    train= data['train'] #학습데이터
    train_labes = data['train_labels'] #데이터 레이블

knn=cv2.ml.KNearest_create() #KNN객체 생성
knn.train(train,cv2.ml.ROW_SAMPLE,train_labes) # KNN객체에 학습시킴

def check(test): #test 하나의 이미지를 주어졌을 때
    #가장 가까운 K(=1 하나)개를 찾아, 어떤 숫자에 해당하는지 찾는다
    ret,result,neibours,dist= knn.findNearest(test,k=1) #하이퍼 파라미터
    return result #데이터 레이블을 반환
```

#하이퍼 파라미터

모델링할 때 사용자가 직접 세팅해주는 값

KNN 알고리즘(K-최근접 이웃)에서는 K값을 직접 조정하여 이웃(최근접)하는 개수를 지정할 수 있다

테스트데이터검증 Run.py

```
def get_result(file_name):
    img = cv2.imread(file_name)
    chars=utils.extract_chars(img) # 테스트할 이미지가 문자로 추출

    result_string=''
    for char in chars:
        matched = check(utils.resized20(char[1]))
                                                    # 테스트 데이터 크기, 1차원 배열로
변환
        if matched <10: # 0~9 인 숫자
            result_string+= str(int(matched))
            continue

        if matched==10:
            matched ='+'
        elif matched == 11:
            matched ='-'
        elif matched == 12:
            matched ='*'

        result_string += matched # 숫자와 수식을 문자열로 변환
    return result_string
```

1433+2028

테스트데이터검증 Run.py

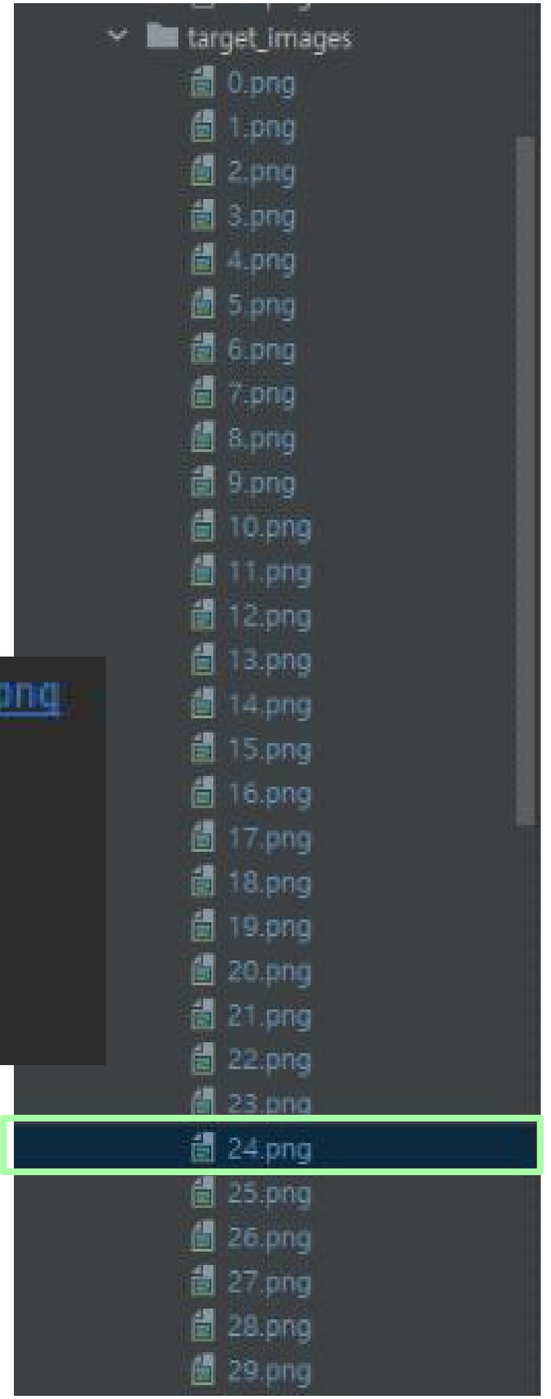
```
def remove_first_0(string): #수식정제
    temp=[]
    for i in string:
        if i == '+' or i== '-' or i=='*':
            temp.append(i) #수식 문자
    import re
    arr=re.split('₩*|₩+|₩-',string)  #수식을 기준으로 문자열 나누기
    return str(int(arr[0]))+temp[0]+str(int(arr[1]))
                #숫자 문자열의 앞자리 0을 제거하기 위해 int형 변환하여 제거후 다시 문자열로
변환

# 특정한 폴더에 이미지 파일을 다운로드 받습니다.
response = s.get(image_url, stream=True)
target_image = './target_images/' + str(i) + '.png'
with open(target_image, 'wb') as out_file:
    shutil.copyfileobj(response.raw, out_file)
del response

# 다운로드 받은 이미지 파일을 분석하여 답을 도출합니다.
answer_string = get_result(target_image)
print('String: ' + answer_string)
answer_string = remove_first_0(answer_string)
answer = str(eval(answer_string))
                #ex)eval("7+5") = 12, 문자열로 이루어진 수식을 계산하는 함수
print('Answer: ' + answer)
```

테스트데이터검증 Run.py

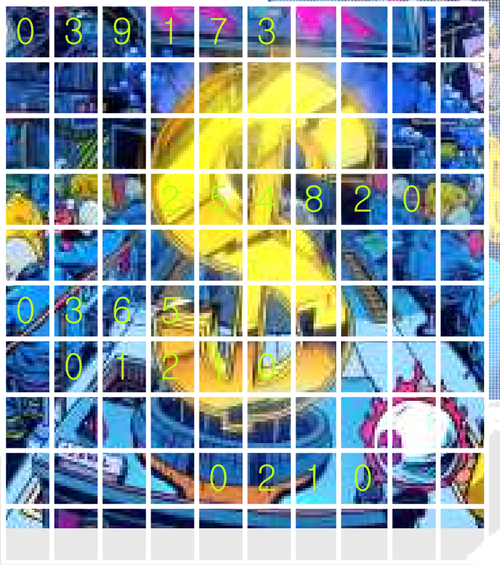

```
Problem 24: http://192.168.0.89:10000/images/b147934c27a2adcbc602dea2e41f51d0.png
String: 2291068690-2454916193  문자열 변환
split
['2291068690', '2454916193']  수식 기준으로 분리
Answer: -163847503  eval 함수로 결과 값 반환
--- 0.06245851516723633 seconds ---
```

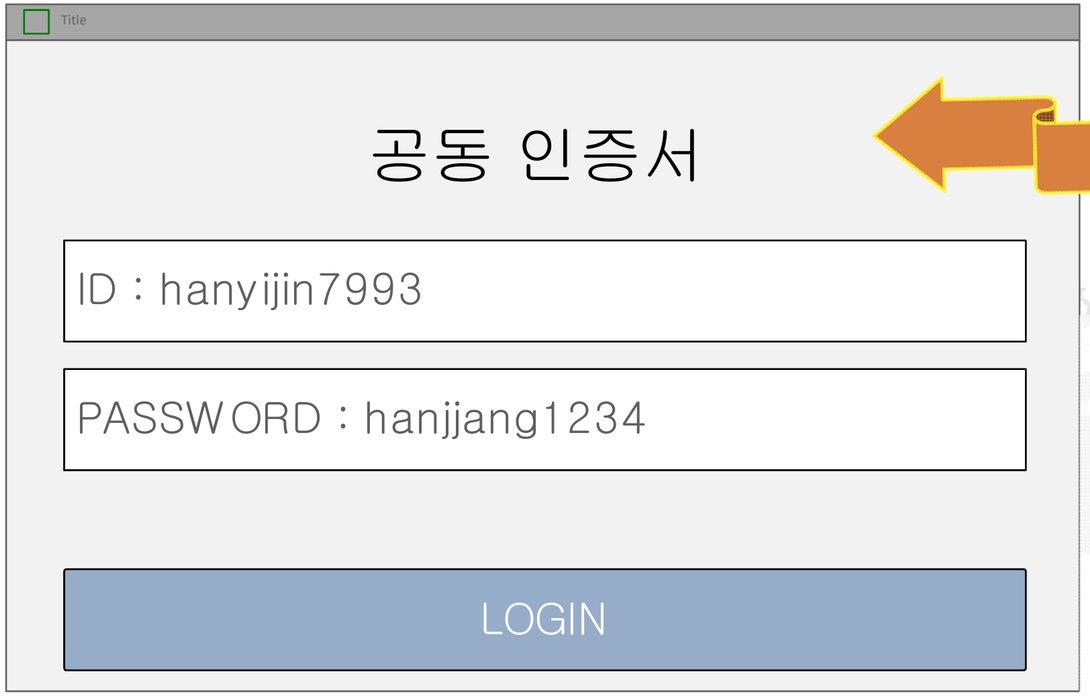


사고의 전환 개인 정보(자산) 이미지 토큰화

NFT (Non-fungible token, 대체불가능토근)

블록체인 (Block Chain)





블록체인의 시스템에서 활용되고 있는 SHA-256

해시 함수 중에 하나, 단방향 암호화로 256bit의 형태를 지닌다
암호화된 결과를 픽셀값으로 변환하여 이미지 생성,
개인 정보, 인증서 등을 생성된 이미지로 대체하여
데이터 관리 및 보안 문제 해결

기술의
품질을
높이는
ㅁㄲㅅ