6.10 홍채 인식 실습

■ 목표

- 홍채 영상을 이용한 신원 인증 알고리즘 생성
- 등록 -> 해시(Iris Code) -> 매칭

■ 원리

- 홍채(Iris):동공 주위의 무늬, 사람마다 고유
- 응용 분야 : 보안 게이트, 여권, 스마트폰 인증

생체인식	장점	단점
지문	빠름	접촉 필요
얼굴	비접촉	조명/표정 영향
홍채	높은 정확도, 위변조 어려움	IR 카메라 필요

6.10 홍채 인식 실습

MMU IRIS DATASET

- https://www.kaggle.com/datasets/naureenmohammad/mmu-iris-dataset?select=MMU-Iris-Database

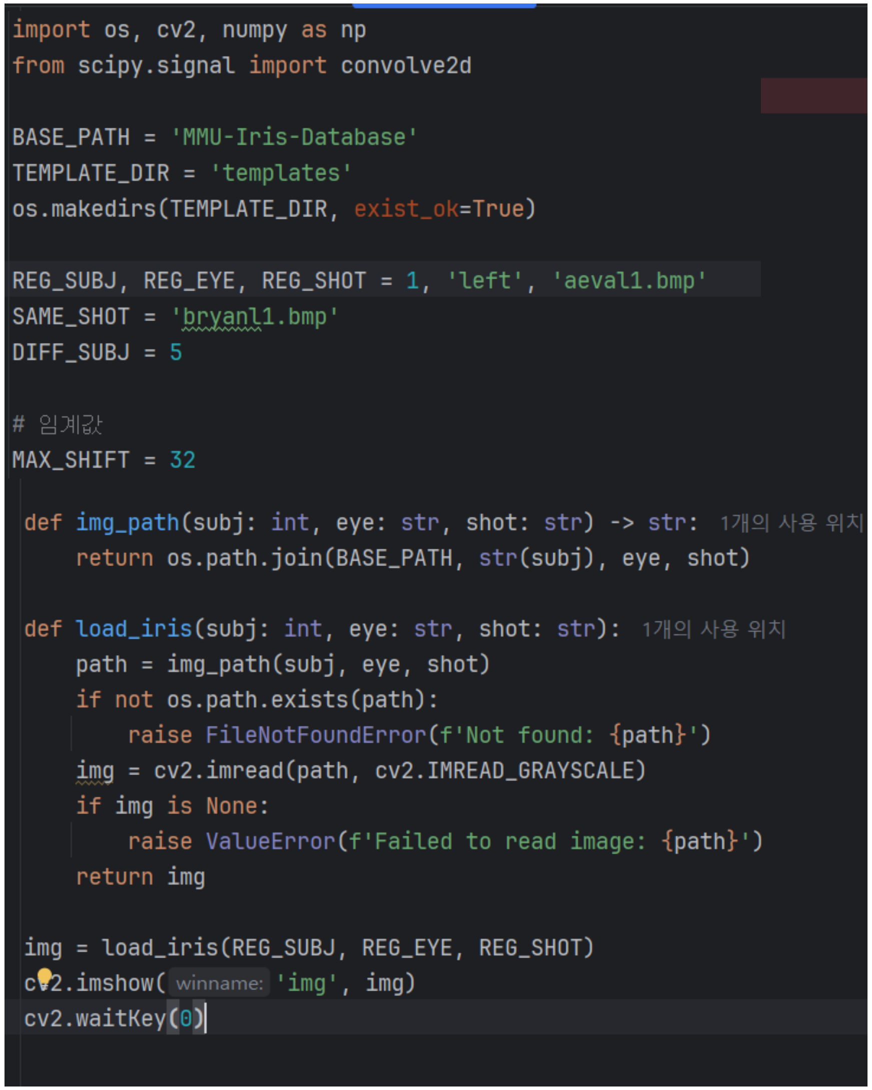
```
MMU-Iris/
├── 1/
│     ├── left/
│     │     ├── aeval1.bmp
│     │     ├── aeval2.bmp
│     └── right/
│           ├── aeval1.bmp
│           ├── aeval2.bmp
├── 2/
│     ├── left/
│     └── right/
└── 3/
      ├── left/
      └── right/
```

5단계 프로세스

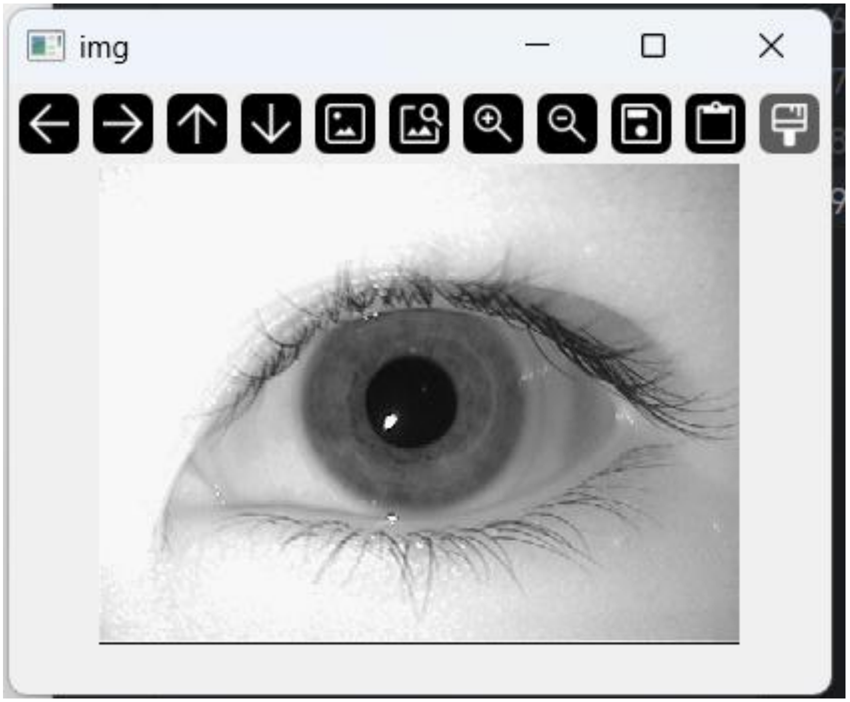
- 획득
- 분할
- 정규화
- 해시 매칭

6.10 홍채 인식 실습

■ 등록 – 이미지 로드

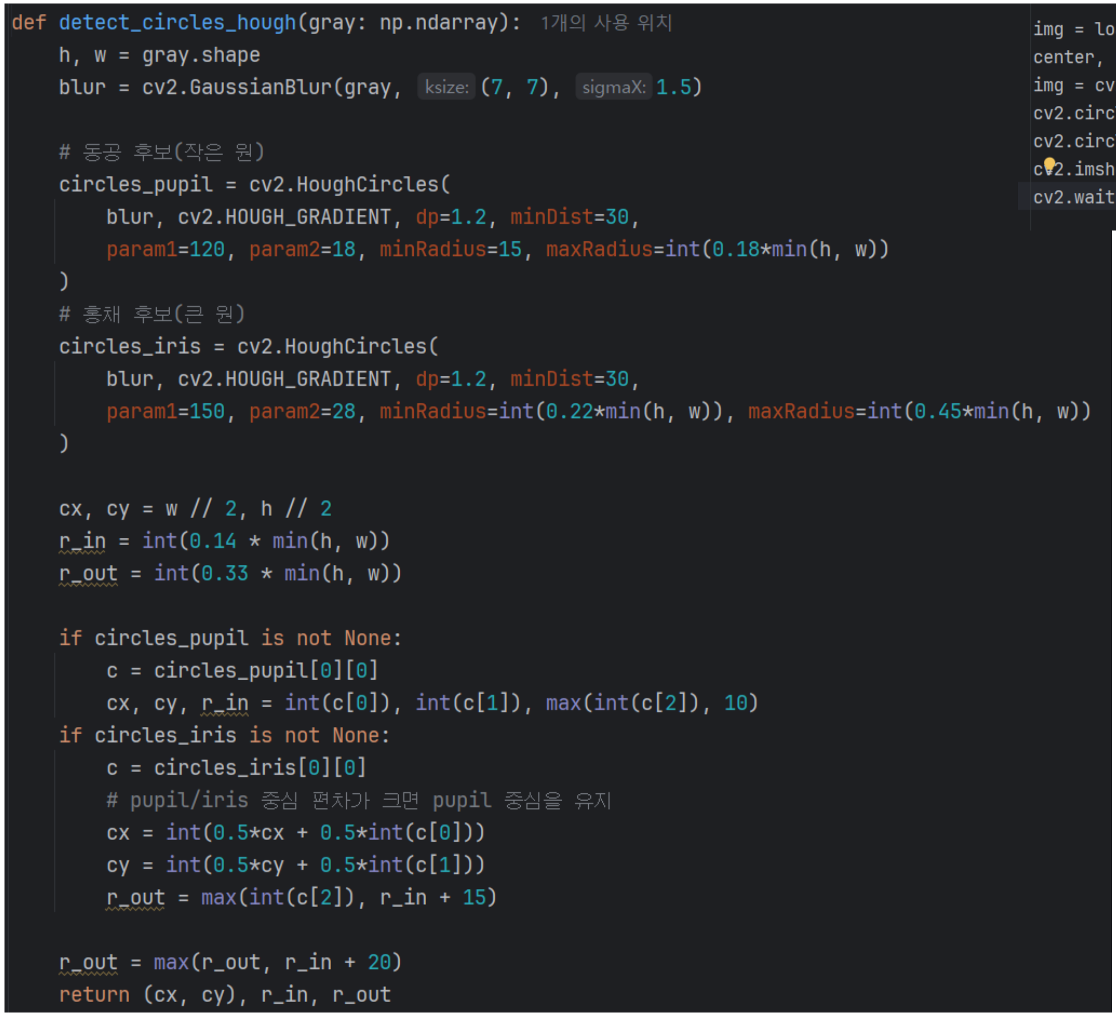


```python
import os, cv2, numpy as np
from scipy.signal import convolve2d

BASE_PATH = 'MMU-Iris-Database'
TEMPLATE_DIR = 'templates'
os.makedirs(TEMPLATE_DIR, exist_ok=True)

REG_SUBJ, REG_EYE, REG_SHOT = 1, 'left', 'aeval1.bmp'
SAME_SHOT = 'bryanl1.bmp'
DIFF_SUBJ = 5

# 임계값
MAX_SHIFT = 32

def img_path(subj: int, eye: str, shot: str) -> str:  1개의 사용 위치
    return os.path.join(BASE_PATH, str(subj), eye, shot)

def load_iris(subj: int, eye: str, shot: str):  1개의 사용 위치
    path = img_path(subj, eye, shot)
    if not os.path.exists(path):
        raise FileNotFoundError(f'Not found: {path}')
    img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
    if img is None:
        raise ValueError(f'Failed to read image: {path}')
    return img

img = load_iris(REG_SUBJ, REG_EYE, REG_SHOT)
cv2.imshow( winname: 'img', img)
cv2.waitKey(0)
```

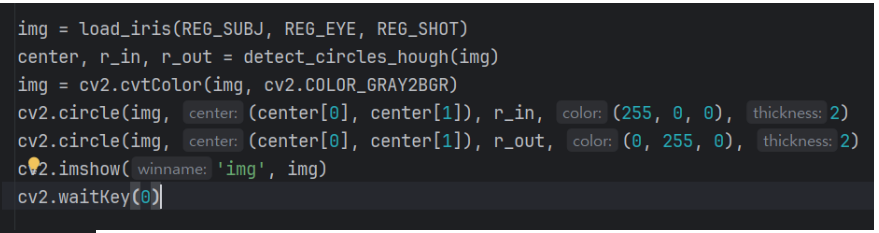


6.10 홍채 인식 실습

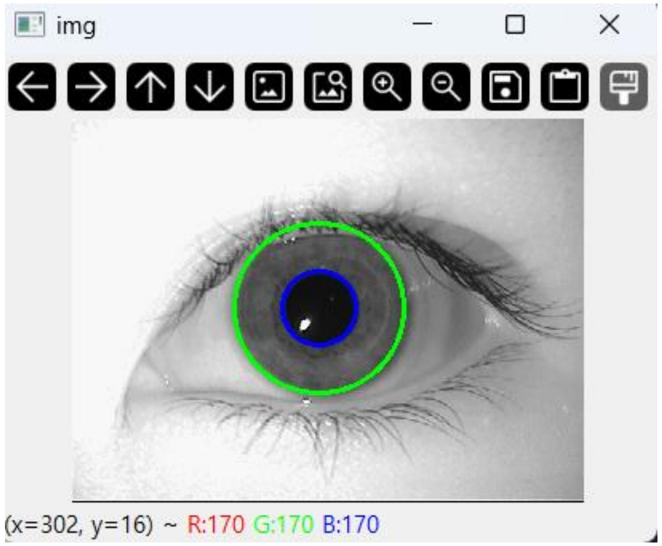
■ 등록 – pupil과 Iris 찾기



```python
def detect_circles_hough(gray: np.ndarray):  1개의 사용 위치
    h, w = gray.shape
    blur = cv2.GaussianBlur(gray, ksize: (7, 7), sigmaX: 1.5)

    # 동공 후보(작은 원)
    circles_pupil = cv2.HoughCircles(
        blur, cv2.HOUGH_GRADIENT, dp=1.2, minDist=30,
        param1=120, param2=18, minRadius=15, maxRadius=int(0.18*min(h, w))
    )
    # 홍채 후보(큰 원)
    circles_iris = cv2.HoughCircles(
        blur, cv2.HOUGH_GRADIENT, dp=1.2, minDist=30,
        param1=150, param2=28, minRadius=int(0.22*min(h, w)), maxRadius=int(0.45*min(h, w))
    )

    cx, cy = w // 2, h // 2
    r_in = int(0.14 * min(h, w))
    r_out = int(0.33 * min(h, w))

    if circles_pupil is not None:
        c = circles_pupil[0][0]
        cx, cy, r_in = int(c[0]), int(c[1]), max(int(c[2]), 10)
    if circles_iris is not None:
        c = circles_iris[0][0]
        # pupil/iris 중심 편차가 크면 pupil 중심을 유지
        cx = int(0.5*cx + 0.5*int(c[0]))
        cy = int(0.5*cy + 0.5*int(c[1]))
        r_out = max(int(c[2]), r_in + 15)

    r_out = max(r_out, r_in + 20)
    return (cx, cy), r_in, r_out
```

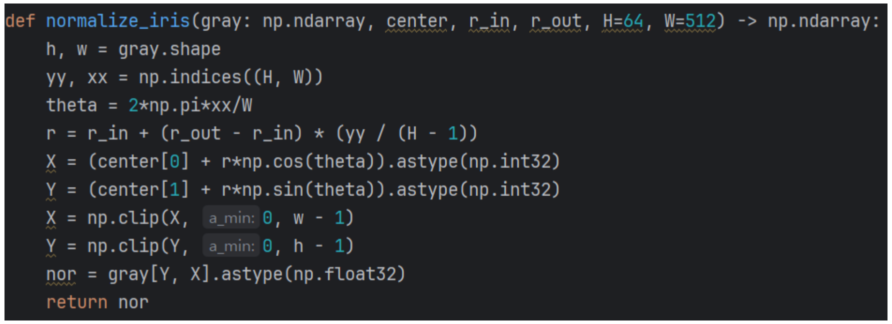


```python
img = load_iris(REG_SUBJ, REG_EYE, REG_SHOT)
center, r_in, r_out = detect_circles_hough(img)
img = cv2.cvtColor(img, cv2.COLOR_GRAY2BGR)
cv2.circle(img, center: (center[0], center[1]), r_in, color: (255, 0, 0), thickness: 2)
cv2.circle(img, center: (center[0], center[1]), r_out, color: (0, 255, 0), thickness: 2)
cv2.imshow( winname: 'img', img)
cv2.waitKey(0)
```

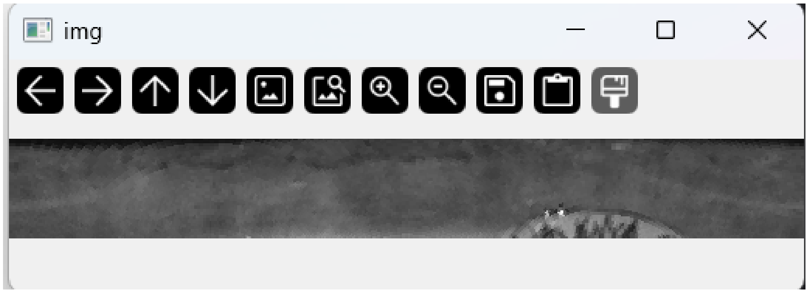


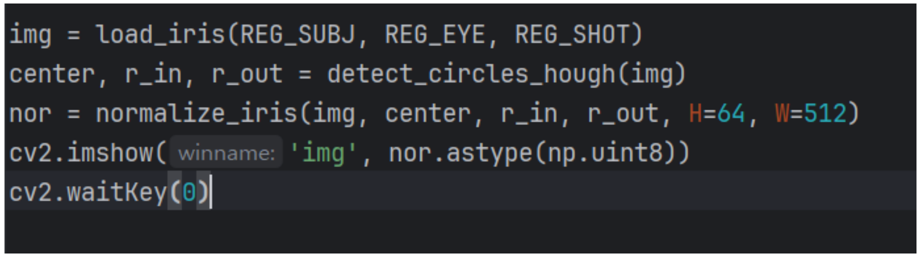
6.10 홍채 인식 실습

■ 등록 – 홍채 펼치기



```
def normalize_iris(gray: np.ndarray, center, r_in, r_out, H=64, W=512) -> np.ndarray:
    h, w = gray.shape
    yy, xx = np.indices((H, W))
    theta = 2*np.pi*xx/W
    r = r_in + (r_out - r_in) * (yy / (H - 1))
    X = (center[0] + r*np.cos(theta)).astype(np.int32)
    Y = (center[1] + r*np.sin(theta)).astype(np.int32)
    X = np.clip(X,  a_min: 0, w - 1)
    Y = np.clip(Y,  a_min: 0, h - 1)
    nor = gray[Y, X].astype(np.float32)
    return nor
```

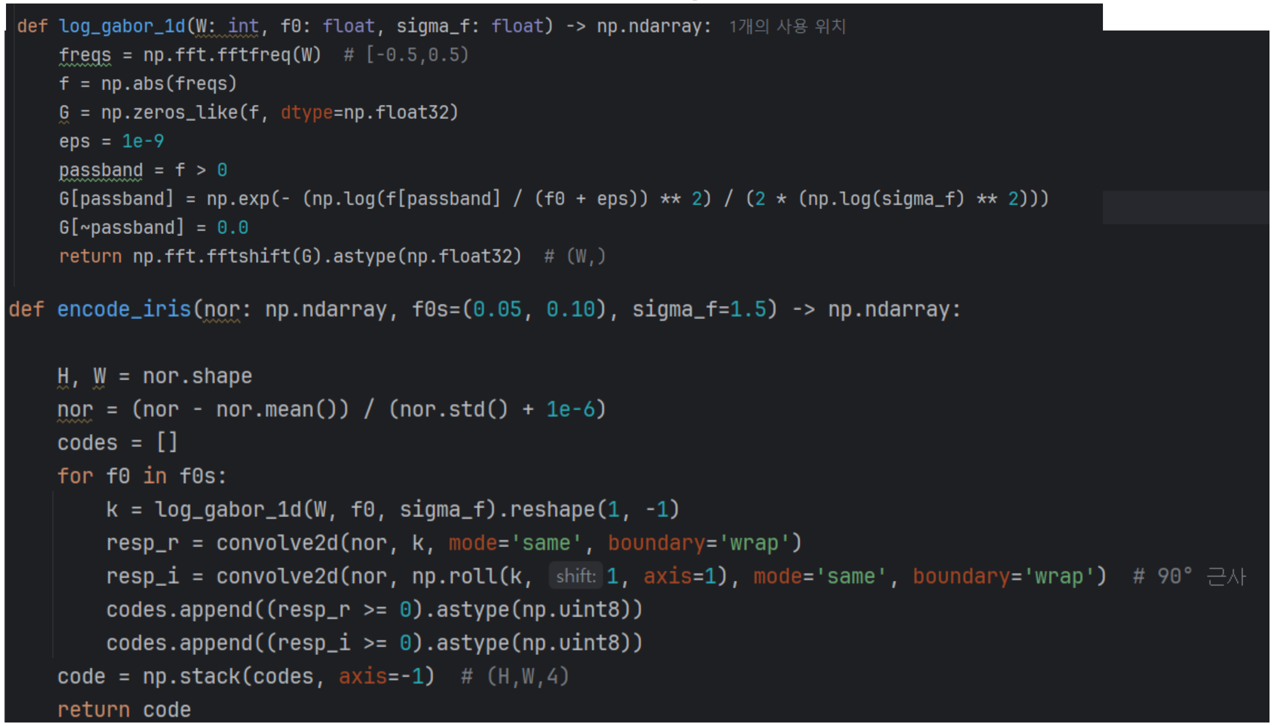




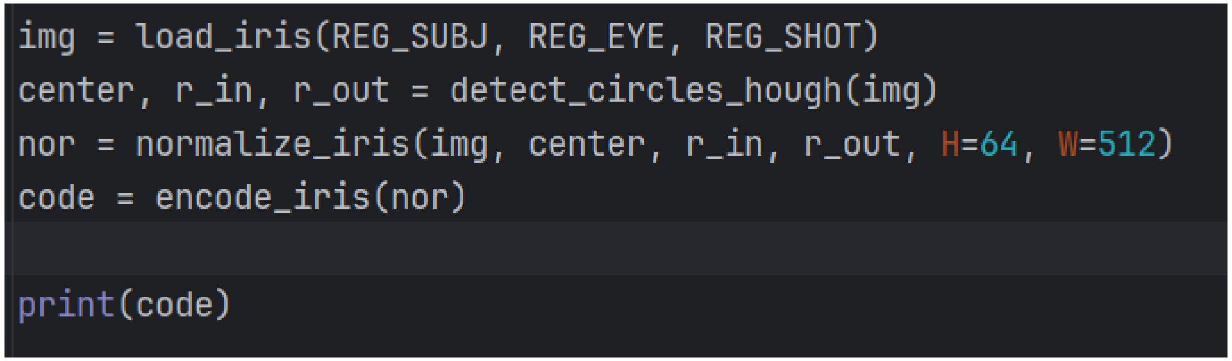
```
img = load_iris(REG_SUBJ, REG_EYE, REG_SHOT)
center, r_in, r_out = detect_circles_hough(img)
nor = normalize_iris(img, center, r_in, r_out, H=64, W=512)
cv2.imshow( winname: 'img', nor.astype(np.uint8))
cv2.waitKey(0)
```

6.10 홍채 인식 실습

■ 등록 – 홍채정보 Encoding



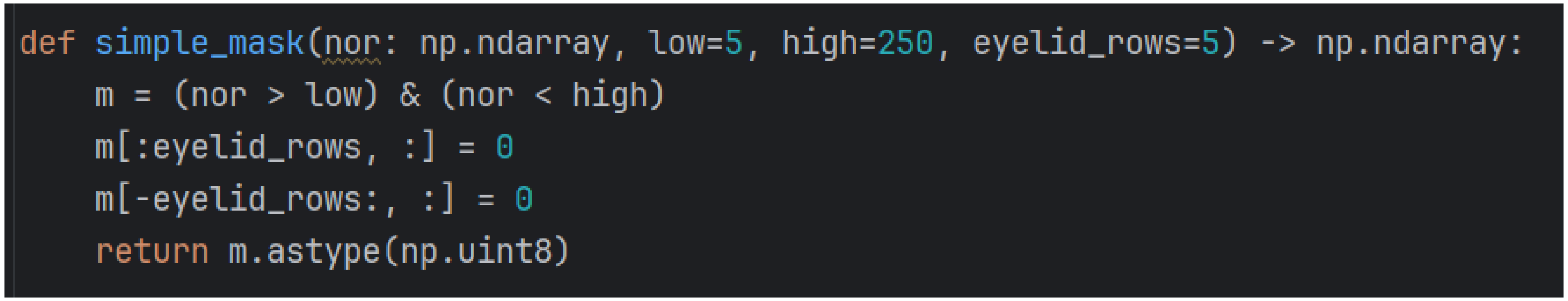
```
def log_gabor_1d(W: int, f0: float, sigma_f: float) -> np.ndarray:  1개의 사용 위치
    freqs = np.fft.fftfreq(W)  # [-0.5,0.5)
    f = np.abs(freqs)
    G = np.zeros_like(f, dtype=np.float32)
    eps = 1e-9
    passband = f > 0
    G[passband] = np.exp(- (np.log(f[passband] / (f0 + eps)) ** 2) / (2 * (np.log(sigma_f) ** 2)))
    G[~passband] = 0.0
    return np.fft.fftshift(G).astype(np.float32)  # (W,)

def encode_iris(nor: np.ndarray, f0s=(0.05, 0.10), sigma_f=1.5) -> np.ndarray:

    H, W = nor.shape
    nor = (nor - nor.mean()) / (nor.std() + 1e-6)
    codes = []
    for f0 in f0s:
        k = log_gabor_1d(W, f0, sigma_f).reshape(1, -1)
        resp_r = convolve2d(nor, k, mode='same', boundary='wrap')
        resp_i = convolve2d(nor, np.roll(k, shift: 1, axis=1), mode='same', boundary='wrap')  # 90° 근사
        codes.append((resp_r >= 0).astype(np.uint8))
        codes.append((resp_i >= 0).astype(np.uint8))
    code = np.stack(codes, axis=-1)  # (H,W,4)
    return code
```



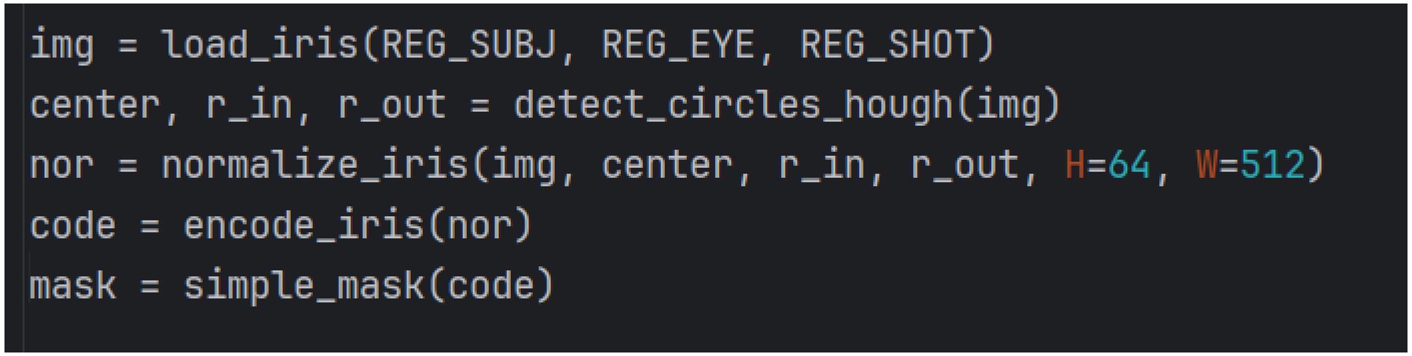
```
img = load_iris(REG_SUBJ, REG_EYE, REG_SHOT)
center, r_in, r_out = detect_circles_hough(img)
nor = normalize_iris(img, center, r_in, r_out, H=64, W=512)
code = encode_iris(nor)

print(code)
```

6.10 홍채 인식 실습

■ 등록 – 오차 예상 영역 masking



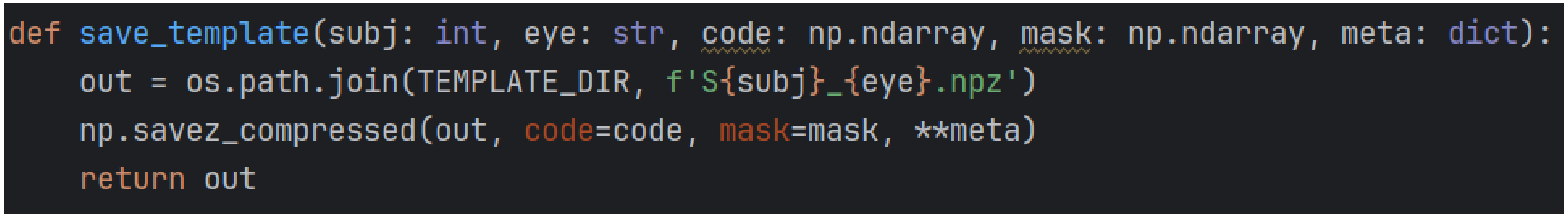
```python
def simple_mask(nor: np.ndarray, low=5, high=250, eyelid_rows=5) -> np.ndarray:
    m = (nor > low) & (nor < high)
    m[:eyelid_rows, :] = 0
    m[-eyelid_rows:, :] = 0
    return m.astype(np.uint8)
```



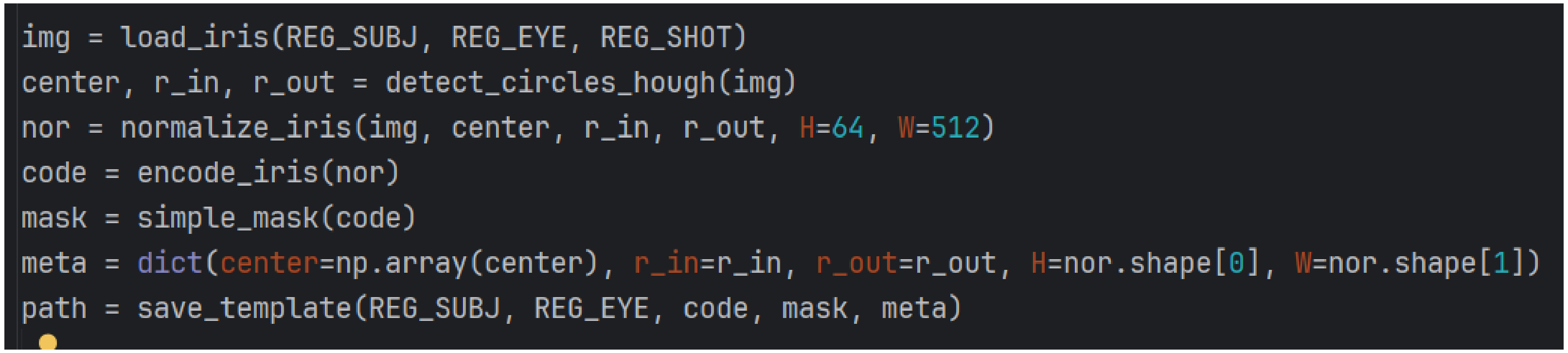
```python
img = load_iris(REG_SUBJ, REG_EYE, REG_SHOT)
center, r_in, r_out = detect_circles_hough(img)
nor = normalize_iris(img, center, r_in, r_out, H=64, W=512)
code = encode_iris(nor)
mask = simple_mask(code)
```

6.10 홍채 인식 실습

■ 등록 – 등록 정보 저장



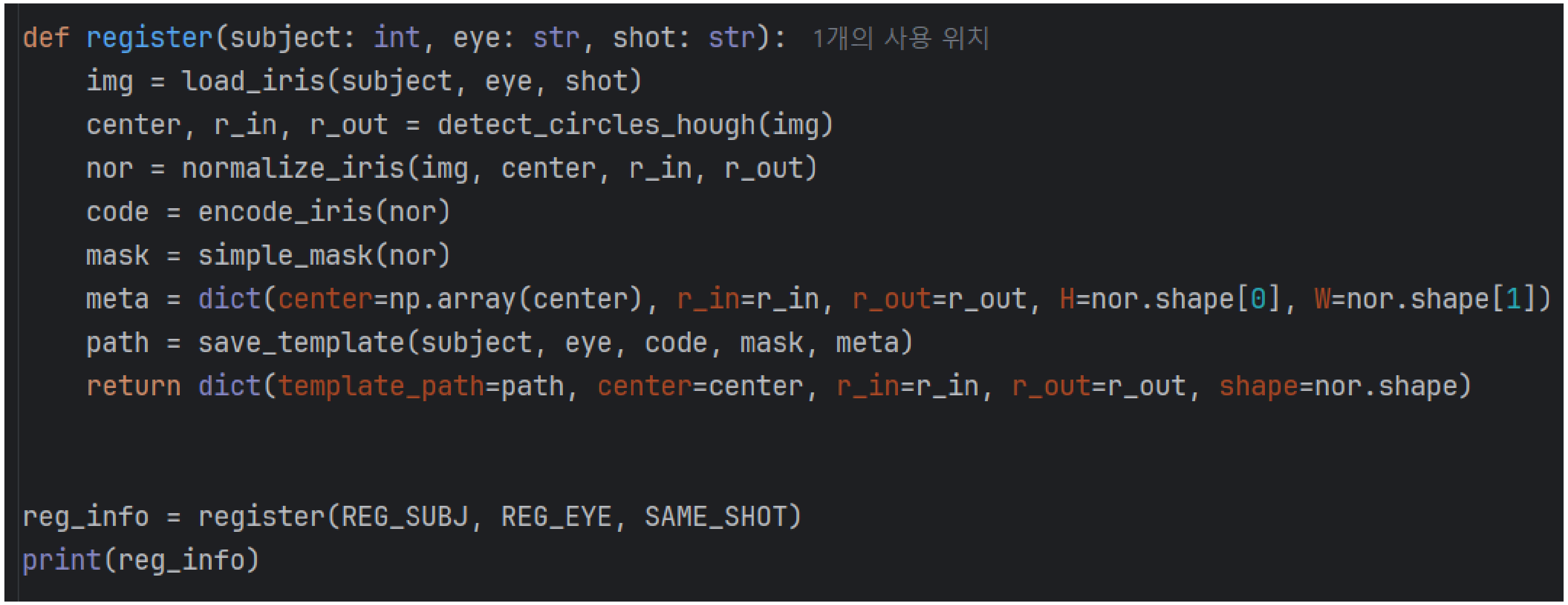
```python
def save_template(subj: int, eye: str, code: np.ndarray, mask: np.ndarray, meta: dict):
    out = os.path.join(TEMPLATE_DIR, f'S{subj}_{eye}.npz')
    np.savez_compressed(out, code=code, mask=mask, **meta)
    return out
```



```python
img = load_iris(REG_SUBJ, REG_EYE, REG_SHOT)
center, r_in, r_out = detect_circles_hough(img)
nor = normalize_iris(img, center, r_in, r_out, H=64, W=512)
code = encode_iris(nor)
mask = simple_mask(code)
meta = dict(center=np.array(center), r_in=r_in, r_out=r_out, H=nor.shape[0], W=nor.shape[1])
path = save_template(REG_SUBJ, REG_EYE, code, mask, meta)
```

6.10 홍채 인식 실습

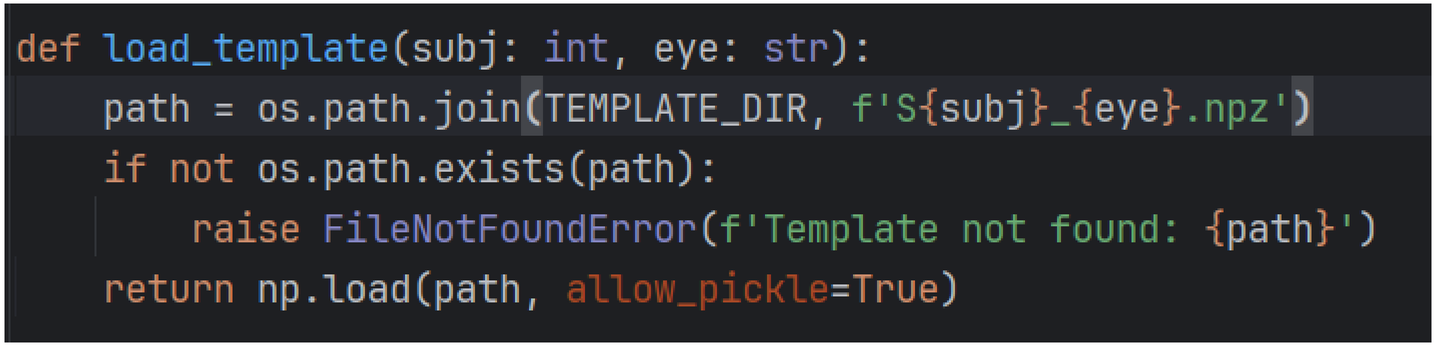
■ 등록 – 함수화



```python
def register(subject: int, eye: str, shot: str):  1개의 사용 위치
    img = load_iris(subject, eye, shot)
    center, r_in, r_out = detect_circles_hough(img)
    nor = normalize_iris(img, center, r_in, r_out)
    code = encode_iris(nor)
    mask = simple_mask(nor)
    meta = dict(center=np.array(center), r_in=r_in, r_out=r_out, H=nor.shape[0], W=nor.shape[1])
    path = save_template(subject, eye, code, mask, meta)
    return dict(template_path=path, center=center, r_in=r_in, r_out=r_out, shape=nor.shape)


reg_info = register(REG_SUBJ, REG_EYE, SAME_SHOT)
print(reg_info)
```

6.10 홍채 인식 실습

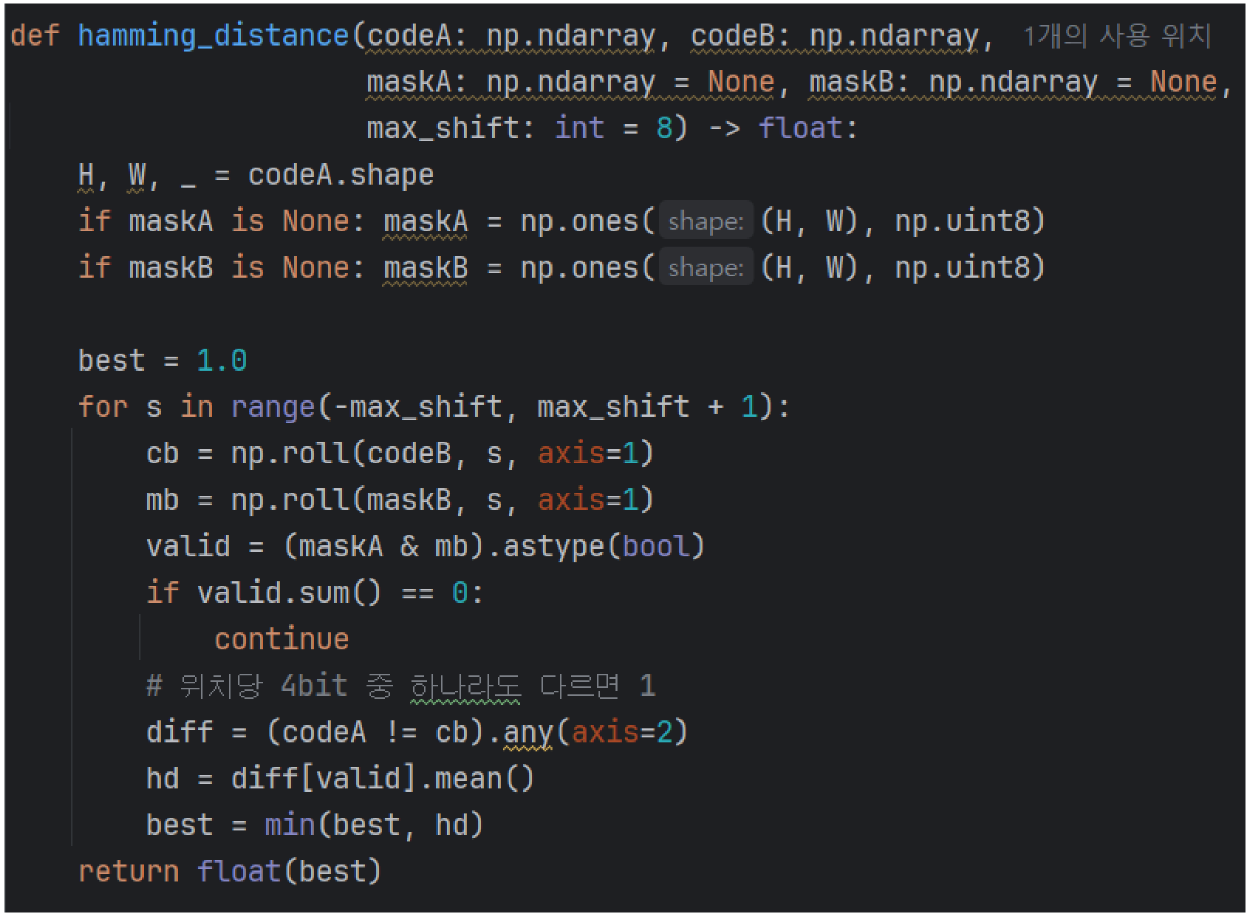
■ 신원 인증 – 저장정보 load



```python
def load_template(subj: int, eye: str):
    path = os.path.join(TEMPLATE_DIR, f'S{subj}_{eye}.npz')
    if not os.path.exists(path):
        raise FileNotFoundError(f'Template not found: {path}')
    return np.load(path, allow_pickle=True)
```

6.10 홍채 인식 실습

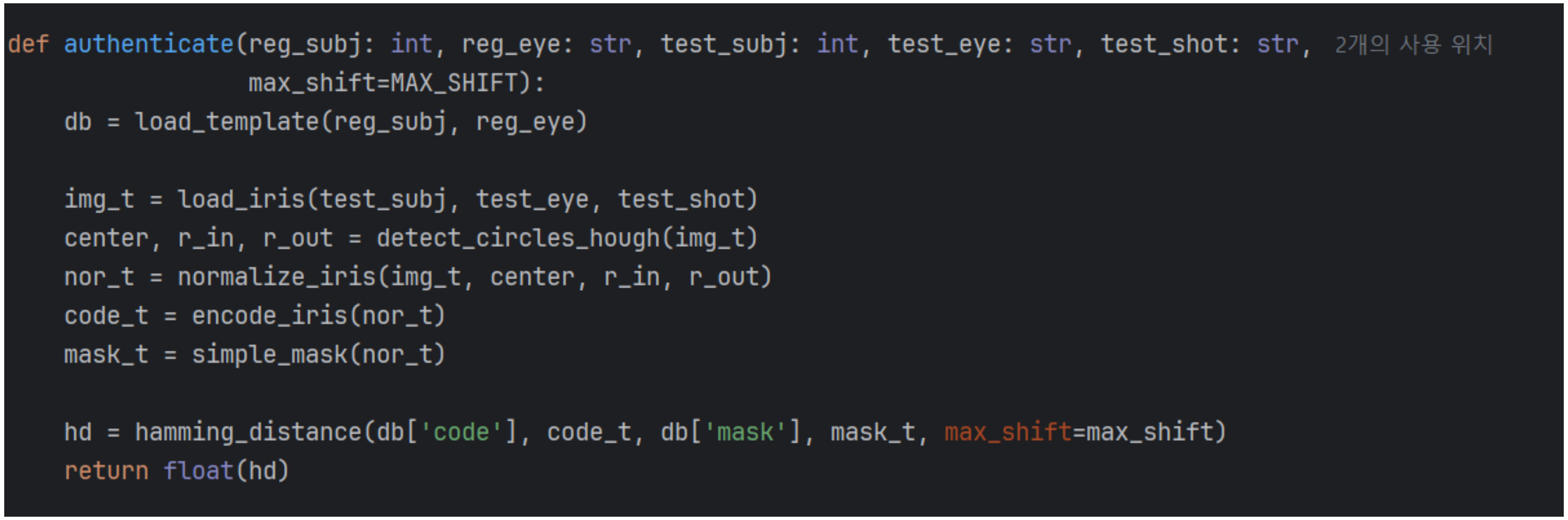
■ 신원 인증 – 해밍거리



```
def hamming_distance(codeA: np.ndarray, codeB: np.ndarray,  1개의 사용 위치
                     maskA: np.ndarray = None, maskB: np.ndarray = None,
                     max_shift: int = 8) -> float:
    H, W, _ = codeA.shape
    if maskA is None: maskA = np.ones( shape: (H, W), np.uint8)
    if maskB is None: maskB = np.ones( shape: (H, W), np.uint8)

    best = 1.0
    for s in range(-max_shift, max_shift + 1):
        cb = np.roll(codeB, s, axis=1)
        mb = np.roll(maskB, s, axis=1)
        valid = (maskA & mb).astype(bool)
        if valid.sum() == 0:
            continue
        # 위치당 4bit 중 하나라도 다르면 1
        diff = (codeA != cb).any(axis=2)
        hd = diff[valid].mean()
        best = min(best, hd)
    return float(best)
```

6.10 홍채 인식 실습

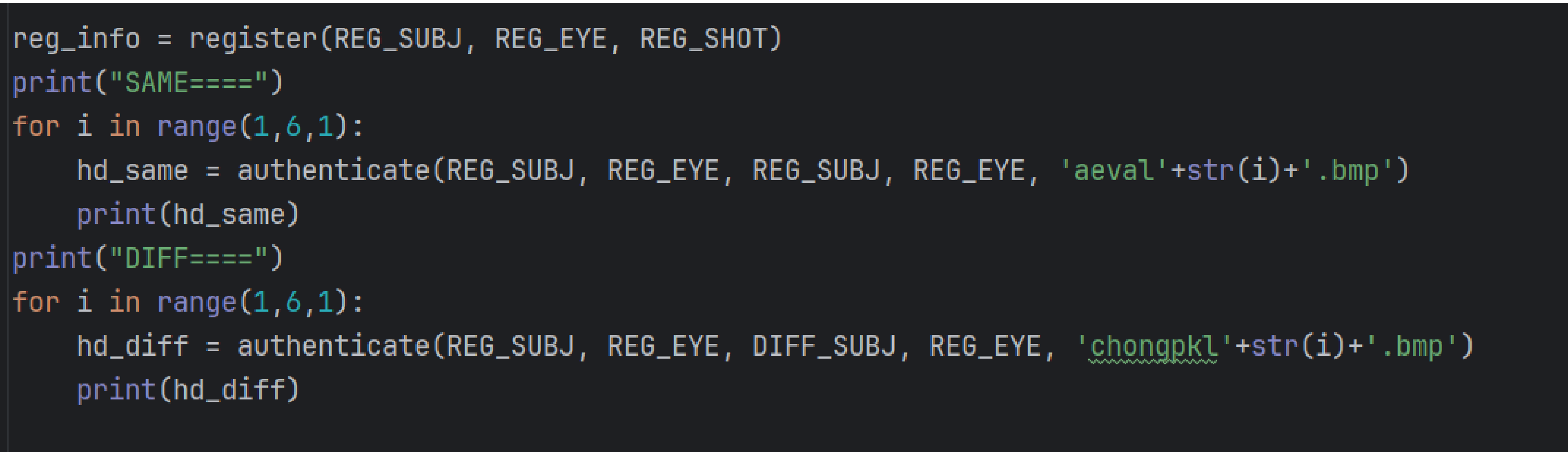
■ 신원 인증



```python
def authenticate(reg_subj: int, reg_eye: str, test_subj: int, test_eye: str, test_shot: str,
                 max_shift=MAX_SHIFT):
    db = load_template(reg_subj, reg_eye)

    img_t = load_iris(test_subj, test_eye, test_shot)
    center, r_in, r_out = detect_circles_hough(img_t)
    nor_t = normalize_iris(img_t, center, r_in, r_out)
    code_t = encode_iris(nor_t)
    mask_t = simple_mask(nor_t)

    hd = hamming_distance(db['code'], code_t, db['mask'], mask_t, max_shift=max_shift)
    return float(hd)
```

6.10 홍채 인식 실습

■ 신원 인증 -test



```python
reg_info = register(REG_SUBJ, REG_EYE, REG_SHOT)
print("SAME====")
for i in range(1,6,1):
    hd_same = authenticate(REG_SUBJ, REG_EYE, REG_SUBJ, REG_EYE, 'aeval'+str(i)+'.bmp')
    print(hd_same)
print("DIFF====")
for i in range(1,6,1):
    hd_diff = authenticate(REG_SUBJ, REG_EYE, DIFF_SUBJ, REG_EYE, 'chongpkl'+str(i)+'.bmp')
    print(hd_diff)
```