Card - type Identification

The project focuses on utilizing **OpenCV** for image processing tasks to identify the type of a card based on the **location of black blocks** along its edges. Assuming the card is upright, the algorithm employes **connected component analysis** to recognize the blocks and maps their positions to specific card types.
The method focuses on the **spatial arrangement** of blocks rather than card symbols or numbers.

Approach

The primary dependency used for the programming code is **NumPy**, which is utilized for all the numerical operations carried out in the program.
Here, OpenCV is used solely for image processing tasks such as loading, thresholding, and connected component analysis.

The implementation of the program is quoted below,

1. **Load and preprocess the image**: The image is loaded in grayscale and thresholded to create a binary image.

2. **Extract connected components**: Connected components are extracted from the inverted binary image, representing individual blocks on the card.

3. **Calculate component areas**: The **areas** of each connected component are calculated.

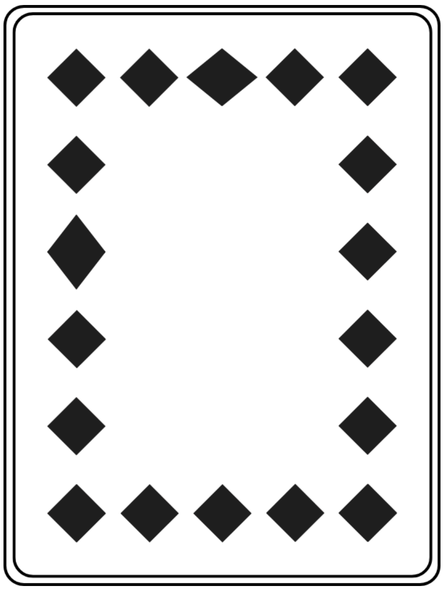
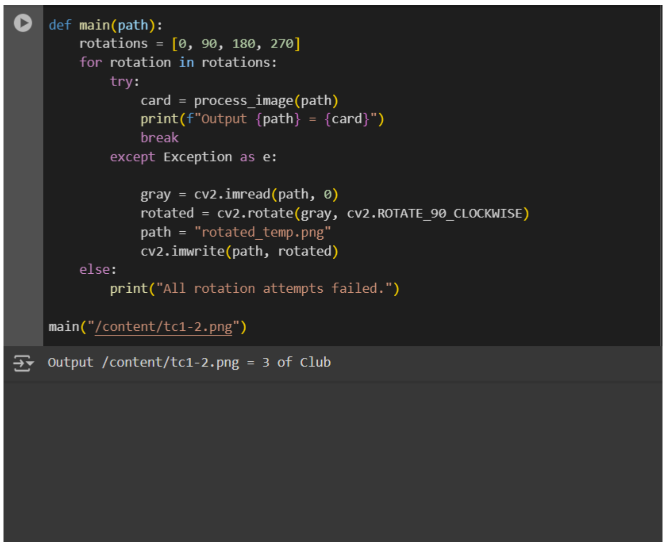
4. **Identify largest components**: The two largest components are identified based on their areas.

5. **Map components to card typ**e: The identified components are mapped to card numbers (based on their positions) and card suits (based on their relative positions).

6. **Handle rotation**: If the initial processing fails, the image is rotated 90 degrees and the process is repeated until a valid card type is identified or all rotations have been tried.

Output

Output for the 6 card images along with the cards,

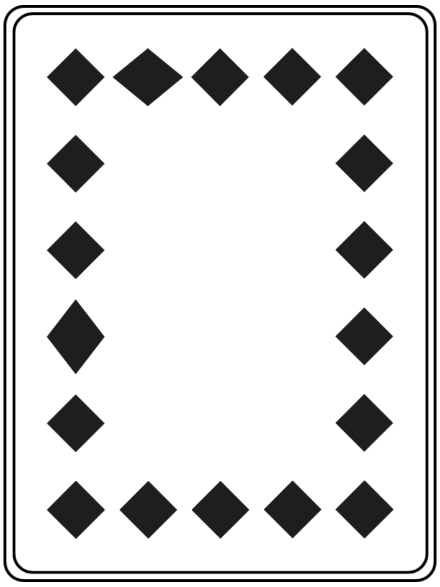
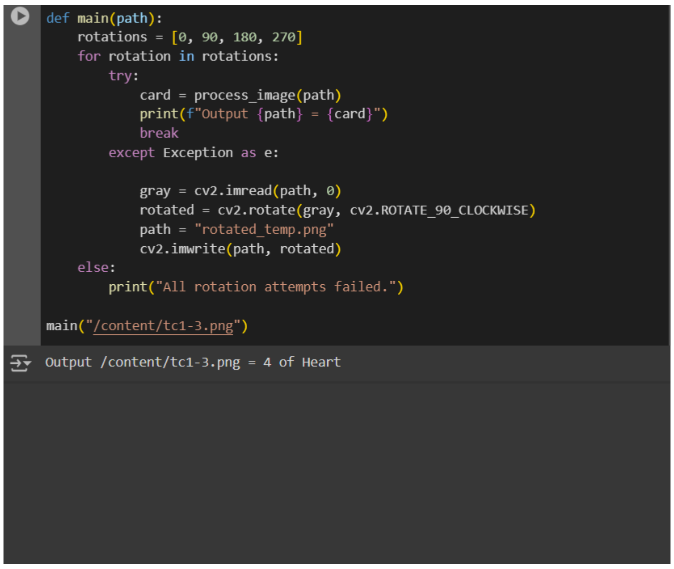



```
def main(path):
    rotations = [0, 90, 180, 270]
    for rotation in rotations:
        try:
            card = process_image(path)
            print(f"Output {path} = {card}")
            break
        except Exception as e:

            gray = cv2.imread(path, 0)
            rotated = cv2.rotate(gray, cv2.ROTATE_90_CLOCKWISE)
            path = "rotated_temp.png"
            cv2.imwrite(path, rotated)
    else:
        print("All rotation attempts failed.")

main("/content/tc1-2.png")
```

```
Output /content/tc1-2.png = 3 of Club
```

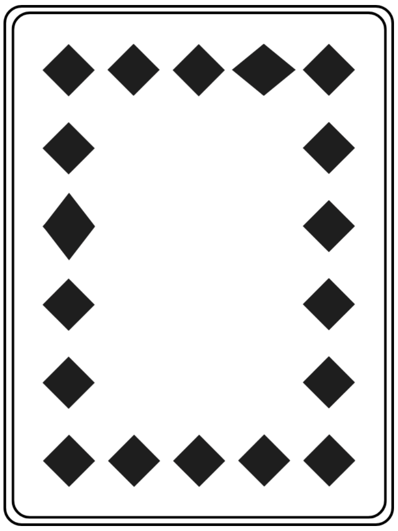



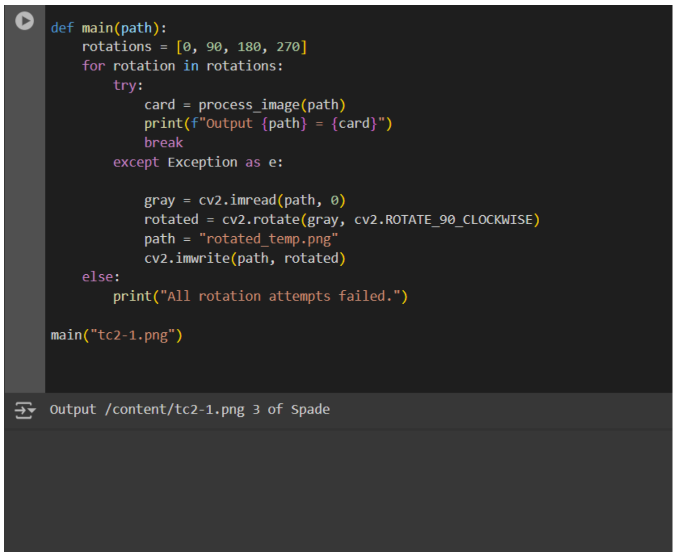
```
def main(path):
    rotations = [0, 90, 180, 270]
    for rotation in rotations:
        try:
            card = process_image(path)
            print(f"Output {path} = {card}")
            break
        except Exception as e:

            gray = cv2.imread(path, 0)
            rotated = cv2.rotate(gray, cv2.ROTATE_90_CLOCKWISE)
            path = "rotated_temp.png"
            cv2.imwrite(path, rotated)
    else:
        print("All rotation attempts failed.")

main("/content/tc1-3.png")
```

```
Output /content/tc1-3.png = 4 of Heart
```



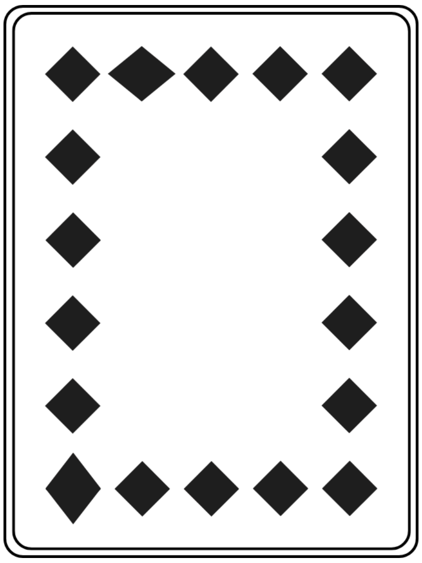


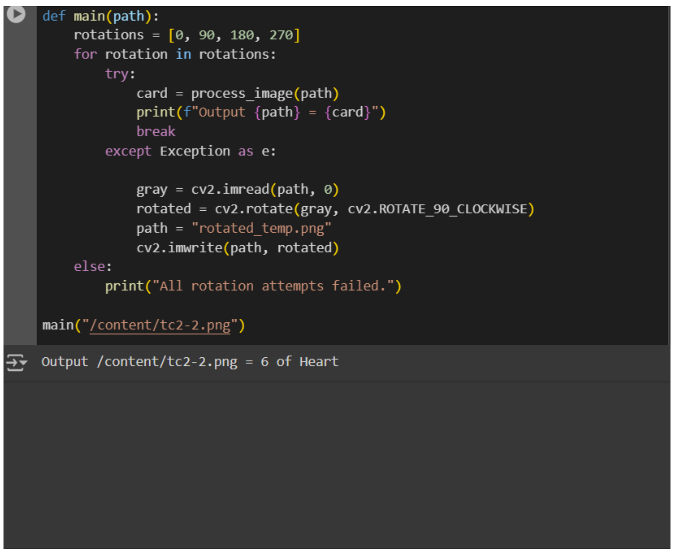
```
def main(path):
    rotations = [0, 90, 180, 270]
    for rotation in rotations:
        try:
            card = process_image(path)
            print(f"Output {path} = {card}")
            break
        except Exception as e:

            gray = cv2.imread(path, 0)
            rotated = cv2.rotate(gray, cv2.ROTATE_90_CLOCKWISE)
            path = "rotated_temp.png"
            cv2.imwrite(path, rotated)
    else:
        print("All rotation attempts failed.")

main("tc2-1.png")
```

```
Output /content/tc2-1.png 3 of Spade
```



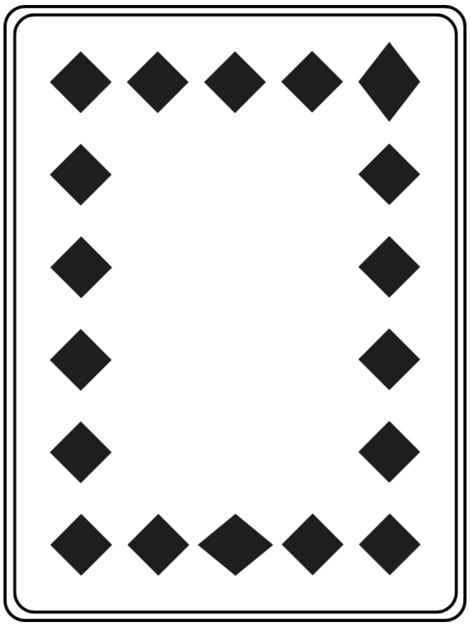


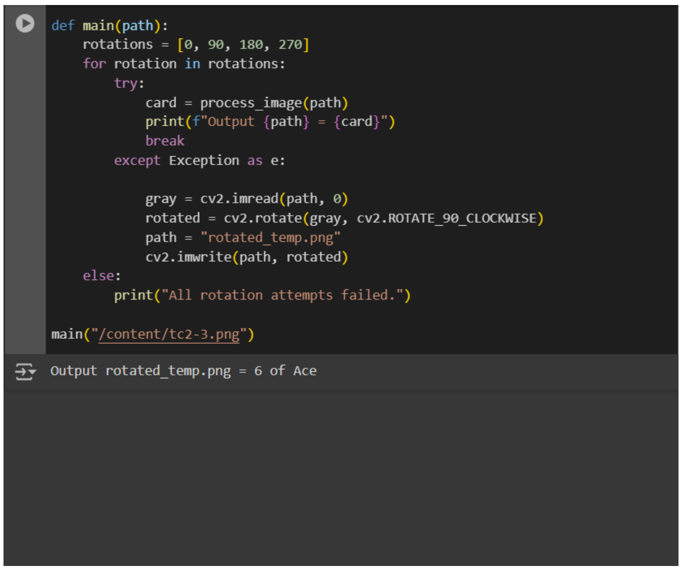
```
def main(path):
    rotations = [0, 90, 180, 270]
    for rotation in rotations:
        try:
            card = process_image(path)
            print(f"Output {path} = {card}")
            break
        except Exception as e:

            gray = cv2.imread(path, 0)
            rotated = cv2.rotate(gray, cv2.ROTATE_90_CLOCKWISE)
            path = "rotated_temp.png"
            cv2.imwrite(path, rotated)
    else:
        print("All rotation attempts failed.")

main("/content/tc2-2.png")
```

```
Output /content/tc2-2.png = 6 of Heart
```



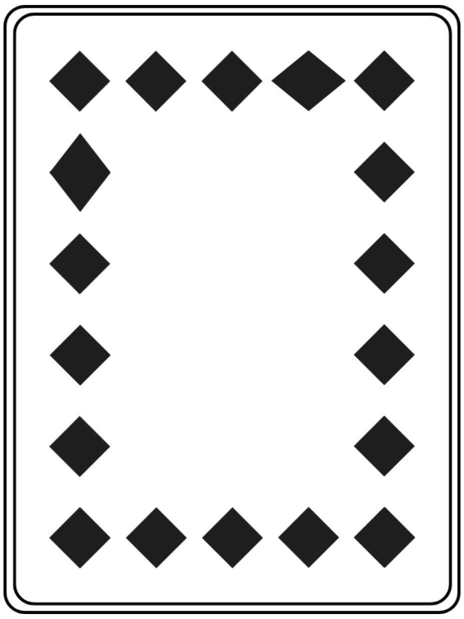


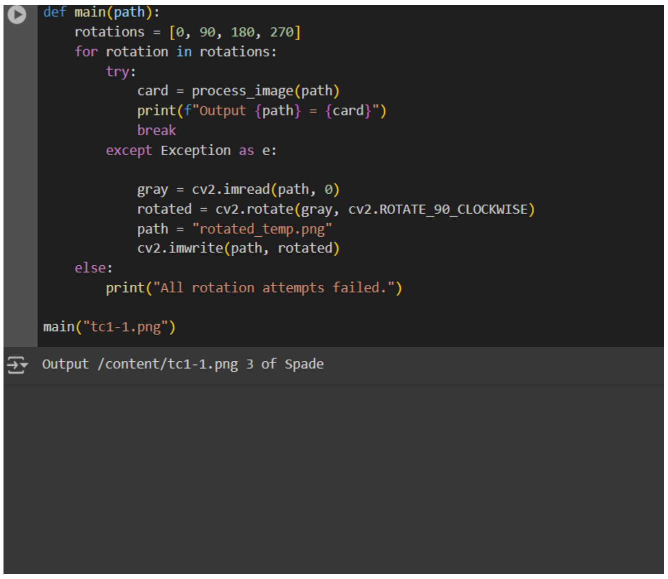
```
def main(path):
    rotations = [0, 90, 180, 270]
    for rotation in rotations:
        try:
            card = process_image(path)
            print(f"Output {path} = {card}")
            break
        except Exception as e:

            gray = cv2.imread(path, 0)
            rotated = cv2.rotate(gray, cv2.ROTATE_90_CLOCKWISE)
            path = "rotated_temp.png"
            cv2.imwrite(path, rotated)
    else:
        print("All rotation attempts failed.")

main("/content/tc2-3.png")
```

```
Output rotated_temp.png = 6 of Ace
```





```
def main(path):
    rotations = [0, 90, 180, 270]
    for rotation in rotations:
        try:
            card = process_image(path)
            print(f"Output {path} = {card}")
            break
        except Exception as e:

            gray = cv2.imread(path, 0)
            rotated = cv2.rotate(gray, cv2.ROTATE_90_CLOCKWISE)
            path = "rotated_temp.png"
            cv2.imwrite(path, rotated)
    else:
        print("All rotation attempts failed.")

main("tc1-1.png")
```

```
Output /content/tc1-1.png 3 of Spade
```