1. Basic Functions

```
# Read in an image
img = cv.imread('../Resources/Photos/park.jpg')
cv.imshow('Park', img)

# Converting to grayscale
gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
cv.imshow('Gray', gray)

# Blur
blur = cv.GaussianBlur(img, (7,7), cv.BORDER_DEFAULT)
cv.imshow('Blur', blur)

# Edge Cascade
canny = cv.Canny(blur, 125, 175)
cv.imshow('Canny Edges', canny)

# Dilating the image
dilated = cv.dilate(canny, (7,7), iterations=3)
cv.imshow('Dilated', dilated)

# Eroding
eroded = cv.erode(dilated, (7,7), iterations=3)
cv.imshow('Eroded', eroded)

# Resize
resized = cv.resize(img, (500,500), interpolation=cv.INTER_CUBIC)
cv.imshow('Resized', resized)

# Cropping
cropped = img[50:200, 200:400]
cv.imshow('Cropped', cropped)
```

2. **Read images and videos**

```
# Read image
img = cv.imread('../Resources/Photos/cats.jpg')
cv.imshow('Cats', img)
cv.waitKey(0)

# Reading Videos
capture = cv.VideoCapture('../Resources/Videos/dog.mp4')
while True:
isTrue, frame = capture.read()
# if cv.waitKey(20) & 0xFF==ord('d'):
# This is the preferred way - if `isTrue` is false (the frame could
# not be read, or we're at the end of the video), we immediately
# break from the loop.
if isTrue:
cv.imshow('Video', frame)
if cv.waitKey(20) & 0xFF==ord('d'):
break
else:
break
capture.release()
cv.destroyAllWindows()
```

3. Draw on text:

```
import cv2 as cv
import numpy as np
blank = np.zeros((500,500,3), dtype='uint8')
cv.imshow('Blank', blank)
# 1. Paint the image a certain
Colour
blank[200:300, 300:400] = 0,0,255
cv.imshow('Green', blank)
# 2. Draw a Rectangle
cv.rectangle(blank, (0,0), (blank.shape[1]//2, blank.shape[0]//2), (0,255,0), thickness=-1)
cv.imshow('Rectangle', blank)
# 3. Draw A circle
cv.circle(blank, (blank.shape[1]//2, blank.shape[0]//2), 40, (0,0,255), thickness=-1)
cv.imshow('Circle', blank)
# 4. Draw a line
cv.line(blank, (100,250), (300,400), (255,255,255), thickness=3)
cv.imshow('Line', blank)
# 5. Write text
cv.putText(blank, 'Hello, my name is Jason!!!', (0,225), cv.FONT_HERSHEY_TRIPLEX, 1.0,
(0,255,0), 2)
cv.imshow('Text', blank)
```

4. **Threshold**

```
import cv2 as cv

img = cv.imread('../Resources/Photos/cats.jpg')
cv.imshow('Cats', img)

gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
cv.imshow('Gray', gray)

# Simple
Thresholding
threshold, thresh = cv.threshold(gray, 150, 255, cv.THRESH_BINARY )
cv.imshow('Simple Thresholded', thresh)

threshold, thresh_inv = cv.threshold(gray, 150, 255, cv.THRESH_BINARY_INV )
cv.imshow('Simple Thresholded Inverse', thresh_inv)

# Adaptive
Thresholding
adaptive_thresh = cv.adaptiveThreshold(gray, 255, cv.ADAPTIVE_THRESH_GAUSSIAN_C,
cv.THRESH_BINARY_INV, 11, 9)
cv.imshow('Adaptive Thresholding', adaptive_thresh)

cv.waitKey(0)
```

5. **Contour**

```
import cv2 as cv
import numpy as np

img = cv.imread('../Resources/Photos/cats.jpg')
cv.imshow('Cats', img)

blank = np.zeros(img.shape, dtype='uint8')
cv.imshow('Blank', blank)

gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
cv.imshow('Gray', gray)

blur = cv.GaussianBlur(gray, (5,5), cv.BORDER_DEFAULT)
cv.imshow('Blur', blur)

canny = cv.Canny(blur, 125, 175)
cv.imshow('Canny Edges', canny)



contours, hierarchies = cv.findContours(canny, cv.RETR_LIST, cv.CHAIN_APPROX_SIMPLE)
print(f'{len(contours)} contour(s) found!')

cv.drawContours(blank, contours, -1, (0,0,255), 1)
cv.imshow('Contours Drawn', blank)
```

6. **Transformation**

```
import cv2 as cv
import numpy as np

img = cv.imread('../Resources/Photos/park.jpg')
cv.imshow('Park', img)

# Translation
def translate(img, x, y):
transMat = np.float32([[1,0,x],[0,1,y]])
dimensions = (img.shape[1], img.shape[0])
return cv.warpAffine(img, transMat, dimensions)

# -x --> Left
# -y --> Up
# x --> Right
# y --> Down

translated = translate(img, -100, 100)
cv.imshow('Translated', translated)

# Rotation
def rotate(img, angle, rotPoint=None):
(height,width) = img.shape[:2]

if rotPoint is None:
rotPoint = (width//2,height//2)

rotMat = cv.getRotationMatrix2D(rotPoint, angle, 1.0)
dimensions = (width,height)

return cv.warpAffine(img, rotMat, dimensions)

rotated = rotate(img, -45)
cv.imshow('Rotated', rotated)

rotated_rotated = rotate(img, -90)
cv.imshow('Rotated Rotated', rotated_rotated)

# Resizing
resized = cv.resize(img, (500,500), interpolation=cv.INTER_CUBIC)
cv.imshow('Resized', resized)

# Flipping
flip = cv.flip(img, -1)
cv.imshow('Flip', flip)

# Cropping
cropped = img[200:400, 300:400]
cv.imshow('Cropped', cropped)

cv.waitKey(0)
```

Face detection

```
import cv2 as cv

img = cv.imread('../Resources/Photos/group 1.jpg')
cv.imshow('Group of 5 people', img)

gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
cv.imshow('Gray People', gray)

haar_cascade = cv.CascadeClassifier('haar_face.xml')

faces_rect = haar_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=1)

print(f'Number of faces found = {len(faces_rect)}')

for (x,y,w,h) in faces_rect:
cv.rectangle(img, (x,y), (x+w,y+h), (0,255,0), thickness=2)

cv.imshow('Detected Faces', img)

cv.waitKey(0)
```