M = np	image) te a matrix of ones, then multiply it by a scaler of 100 gives a matrix with same dimesions of our image with all values being 100
	.ones(image.shape, dtype = "uint8") * 100  Grayscaled Image
50 - 75 - 100 - 125 -	
150 - 175 - 0 [[ 46 [ 89 1	50 100 150 200 250 97 113 63 79 51] 38 101 80 87 75] 80 26 69 62 73]
 [ 26 [ 42	23 51 71 72 73] 35 73 56 57 58] 17 18 68 68 68]]
[100 1 [100 1  [100 1 [100 1	00 100 100 100 100] 00 100 100 100 100] 00 100 100 100 100] 00 100 100 100 100] 00 100 100 100 100] 00 100 100 100 100] 00 100 100 100 100]
# Notion added: imshow  # Now added2	se this to add this matrix M, to our image ce the increase in brightness = cv2.add(image, M) ("Increasing Brightness", added)  if we just added it, look what happens = image + M
imshow	("Simple Numpy Adding Results in Clipping", added2)  Increasing Brightness
50 - 75 - 100 - 125 -	
150 - 175 - 0	50 100 150 200 250  Simple Numpy Adding Results in Clipping
25 - 50 - 75 -	
100 - 125 - 150 -	
# Notion subtraction imshow	si 100 150 200 250  wise we can also subtract ce the decrease in brightness cted = cv2.subtract(image, M) ("Subtracted", subtracted)
imshow	cted = image - M ("Subtracted 2", subtracted)  Subtracted
50 - 75 - 100 - 125 -	
150 - 175 - 0	50 100 150 200 250  Subtracted 2
25 - 50 -	
100	
# If you # Making square	50 100 150 200 250  ou're wondering why only two dimensions, well this is a grayscale image,  ng a square = np.zeros((300, 300), np.uint8) ctangle(square, (50, 50), (250, 250), 255, -2)
# Makinellipso	("square", square)  ng a ellipse e = np.zeros((300, 300), np.uint8) lipse(ellipse, (150, 150), (150, 150), 30, 0, 180, 255, -1) ("ellipse", ellipse)
0 - 50 -	square
100 -	
150 -	
200 - 250 -	
0	50 100 150 200 250 ellipse
0 - 50 -	
100 -	
150 -	
200 -	
200 -	
250 - 0 7]: # Shows	s only where they intersect cv2. bitwise_and(square, ellipse)  ("AND" And)
250 -  250 -  # Shows hitwise imshow # Shows bitwise imshow bitwise imshow bitwise imshow the shows bitwise imshow the shows bitwise imshow bitwise imshow the shows below the shows by the shows	s only where they intersect cv2.bitwise_and(square, ellipse) ("AND", And)  s where either square or ellipse is e0r = cv2.bitwise_or(square, ellipse) ("bitwise0r", bitwise0r)  s where either exist by itself exor = cv2.bitwise_xor(square, ellipse)
250 -  # Shows And = continued imshow  # Shows bitwise imshow  # Shows bitwise imshow	s only where they intersect cv2.bitwise_and(square, ellipse) ("AND", And)  s where either square or ellipse is e0r = cv2.bitwise_or(square, ellipse) ("bitwise0r", bitwise0r)  s where either exist by itself
250 -  # Shows And = imshow  # Shows bitwise imshow  # Shows bitwise imshow  # Shows bitwise	s only where they intersect cv2.bitwise_and(square, ellipse) ("AND", And)  s where either square or ellipse is eor = cv2.bitwise_or(square, ellipse) ("bitwiseOr", bitwiseOr)  s where either exist by itself eXor = cv2.bitwise_xor(square, ellipse) ("bitwiseXor", bitwiseXor)  s everything that isn't part of the square eNot_sq = cv2.bitwise_not(square) ("bitwiseNot_sq", bitwiseNot_sq)
# Shows And = 0 imshow  # Shows bitwise imshow  # Shows bitwise imshow  # Shows bitwise imshow	s only where they intersect cv2.bitwise_and(square, ellipse) ("AND", And)  s where either square or ellipse is eor = cv2.bitwise_or(square, ellipse) ("bitwiseOr", bitwiseOr)  s where either exist by itself exor = cv2.bitwise_xor(square, ellipse) ("bitwiseXor", bitwiseXor)  s everything that isn't part of the square eNot_sq = cv2.bitwise_not(square) ("bitwiseNot_sq", bitwiseNot_sq)
250 -  # Shows And = Gimshow  # Shows bitwise imshow  # Shows bitwise imshow  # Shows bitwise imshow	s only where they intersect cv2.bitwise_and(square, ellipse) ("AND", And)  s where either square or ellipse is eor = cv2.bitwise_or(square, ellipse) ("bitwiseOr", bitwiseOr)  s where either exist by itself exor = cv2.bitwise_xor(square, ellipse) ("bitwiseXor", bitwiseXor)  s everything that isn't part of the square eNot_sq = cv2.bitwise_not(square) ("bitwiseNot_sq", bitwiseNot_sq)
250 -  # Shows And = dimshow # Shows bitwise imshow # Shows bitwise imshow  # Shows bitwise imshow  100 -	s only where they intersect cv2.bitwise_and(square, ellipse) ("AND", And)  s where either square or ellipse is eor = cv2.bitwise_or(square, ellipse) ("bitwiseOr", bitwiseOr)  s where either exist by itself eXor = cv2.bitwise_xor(square, ellipse) ("bitwiseXor", bitwiseXor)  s everything that isn't part of the square eNot_sq = cv2.bitwise_not(square) ("bitwiseNot_sq", bitwiseNot_sq)
250 -  250 -  # Shows And = 0 imshow  # Shows bitwise imshow  # Shows bitwise imshow  100 -  100 -	s only where they intersect cv2.bitwise_and(square, ellipse) ("AND", And)  s where either square or ellipse is eor = cv2.bitwise_or(square, ellipse) ("bitwiseOr", bitwiseOr)  s where either exist by itself eXor = cv2.bitwise_xor(square, ellipse) ("bitwiseXor", bitwiseXor)  s everything that isn't part of the square eNot_sq = cv2.bitwise_not(square) ("bitwiseNot_sq", bitwiseNot_sq)
250 -  250 -  250 -  250 -  250 -  250 -  250 -  250 -  250 -	s only where they intersect  vv. bitwise.and(square, ellipse)  ("AbU", And)  s where either square or ellipse is  eV = vv. bitwise.or(square, ellipse)  ("intwiseo", bitwiseon)  s where either exist by itself  exur = vv. bitwise.or(square, ellipse)  ("bitwiseon's pitwiseon')  s weverything that isn't part of the square  eNot.Sq = vv. bitwiseon'  ("bitwiseNot.Sq", bitwiseNot.Sq)  AND
250 -  250 -  250 -  250 -  250 -  250 -  250 -  250 -  250 -  200 -  200 -	s only where they intersect  vv2.bitwise and(square, ellipse)  ("NDO", And)  s where either square or ellipse is  our = cv2.bitwise_or(square, ellipse)  ("intrisseon", bitwiseon)  s where either exist by itself  XNOT = cv2.bitwise_or(square, ellipse)  ("intrisseon", bitwiseon)  s verything that isn't part of the square  NHOTO AND  AND  AND
250 -  250 -  250 -  250 -  250 -  250 -  250 -  250 -  250 -  250 -  250 -  250 -  250 -  250 -	s only where they intersect  vv. bitwise.and(square, ellipse)  ("AbU", And)  s where either square or ellipse is  eV = vv. bitwise.or(square, ellipse)  ("intwiseo", bitwiseon)  s where either exist by itself  exur = vv. bitwise.or(square, ellipse)  ("bitwiseon's pitwiseon')  s weverything that isn't part of the square  eNot.Sq = vv. bitwiseon'  ("bitwiseNot.Sq", bitwiseNot.Sq)  AND
250 -  0  # Shows imshow # Shows ims	s only where they intersect  vv. bitwise.and(square, ellipse)  ("AbU", And)  s where either square or ellipse is  eV = vv. bitwise.or(square, ellipse)  ("intwiseo", bitwiseon)  s where either exist by itself  exur = vv. bitwise.or(square, ellipse)  ("bitwiseon's pitwiseon')  s weverything that isn't part of the square  eNot.Sq = vv. bitwiseon'  ("bitwiseNot.Sq", bitwiseNot.Sq)  AND
250 -  250 -  250 -  250 -  250 -  250 -  250 -  250 -  200 -  200 -  200 -  200 -  200 -  200 -  200 -	s only where they intersect  vv2.bitwise and(square, ellipse)  ("NDO", And)  s where either square or ellipse is  our = cv2.bitwise_or(square, ellipse)  ("intrisseon", bitwiseon)  s where either exist by itself  XNOT = cv2.bitwise_or(square, ellipse)  ("intrisseon", bitwiseon)  s verything that isn't part of the square  NHOTO AND  AND  AND
250 -  # Shows imshow # Shows imshow # Shows imshow  # Shows i	soly later toy intersect  FOR THIS and spare, silipse)  FOR THIS AND STORY SPARE OF CITY STORY  SAME STORY SPARE OF STORY  SAME STORY SPARE OF STORY  SAME STORY SPARE OF STORY  (SAME STORY SPARE OF STORY  (SAME STORY SPARE OF STORY SPARE OF STORY  (SAME STORY SPARE OF STORY SPARE OF STORY  (SAME SPARE SPARE OF STORY SPARE OF
250 -  250 -  250 -  3	soly later toy intersect  FOR THIS and spare, silipse)  FOR THIS AND STORY SPARE OF CITY STORY  SAME STORY SPARE OF STORY  SAME STORY SPARE OF STORY  SAME STORY SPARE OF STORY  (SAME STORY SPARE OF STORY  (SAME STORY SPARE OF STORY SPARE OF STORY  (SAME STORY SPARE OF STORY SPARE OF STORY  (SAME SPARE SPARE OF STORY SPARE OF
250 -  25	soly later toy intersect  FOR THIS and spare, silipse)  FOR THIS AND STORY SPARE OF CIPPSE 15  AND STORY SPARE OF STORY SPARE
250 - 250 -	soly later toy intersect  FOR THIS and spare, silipse)  FOR THIS AND STORY SPARE OF CITY STORY  SAME STORY SPARE OF STORY  SAME STORY SPARE OF STORY  SAME STORY SPARE OF STORY  (SAME STORY SPARE OF STORY  (SAME STORY SPARE OF STORY SPARE OF STORY  (SAME STORY SPARE OF STORY SPARE OF STORY  (SAME SPARE SPARE OF STORY SPARE OF
250 -  250 -  250 -  250 -  250 -  250 -  200 -  250 -  200 -  250 -  200 -  250 -  200 -  250 -  200 -  250 -  200 -  250 -  200 -  250 -  200 -  250 -  200 -  250 -  200 -  250 -  200 -  250 -  200 -  250 -  200 -  250 -  200 -  250 -  200 -  250 -  250 -  200 -  25	soly later toy intersect  FOR THIS and spare, silipse)  FOR THIS AND STORY SPARE OF CITY STORY  SAME STORY SPARE OF STORY  SAME STORY SPARE OF STORY  SAME STORY SPARE OF STORY  (SAME STORY SPARE OF STORY  (SAME STORY SPARE OF STORY SPARE OF STORY  (SAME STORY SPARE OF STORY SPARE OF STORY  (SAME SPARE SPARE OF STORY SPARE OF
250 -  250 -	E (1)   Serve (1/4)   10   Serve (1/4)   10   Serve (1/4)   Serve (1/4)
250 -  250 -  250 -  250 -  250 -  250 -  200 -  250 -  200 -  250 -  200 -  250 -  200 -  250 -  200 -  250 -  200 -  250 -  200 -  250 -  200 -  250 -  200 -  250 -  200 -  250 -  200 -  250 -  200 -  250 -  200 -  250 -  200 -  250 -  200 -  250 -  250 -  200 -  25	E (1)   Section
250 -  250 -  3	E (1)   Section
250 -  250 -  250 -  250 -  250 -  250 -  200 -  250 -  200 -  250 -  200 -  250 -  200 -  25	E. Org., party Chry 102 result (1987; party C

In [1]: # Our Setup, Import Libaries, Create our Imshow Function and Download our Images import cv2

import numpy as np
from matplotlib import pyplot as plt

def imshow(title = "Image", image = None, size = 10):
 w, h = image.shape[0], image.shape[1]
 aspect\_ratio = w/h
 plt.figure(figsize=(size \* aspect\_ratio, size))

plt.imshow(cv2.cvtColor(image, cv2.COLOR\_BGR2RGB))

# Define our imshow function

plt.title(title)
plt.show()