def correlation(image, filter\_): # 'filter' is a keyword in python, so is the underscore at the end.

"""

Performs linear filtering on an image. Standardizes image before applying the filter.

Assume image size is W1xW2, filter size is F1xF2.

Arguments:

image -- input image as a numpy array

filter\_ -- linear filter to apply on image as a numpy array.

Returns:

result -- filtered image.

"""

image\_height, image\_width = image.shape

filter\_height, filter\_width = filter\_.shape

# result shape will be of size --> (((W1−F1+2P) / S) + 1) x (((W2−F2+2P) / S) + 1), where 'P' is padding length

# S is stride length, if you don't know about them, don't worry, you will learn in upcoming lectures. For now

# we will use simplest setting P=0,S=1. See the next line.

result\_height, result\_width = (image\_height - filter\_height) + 1, (image\_width - filter\_width) + 1

result = np.zeros((result\_height, result\_width))

for i in range(result\_width):

for j in range(result\_height):

std\_image=standardize(image[j:j+filter\_height,i:i+filter\_width])

result[j,i]=np.sum(filter\_\*std\_image)

return result

# To test your implementation, run the below code.

image = Image.open('./images/car\_road.jpg')

filter\_ = Image.open('./images/car.jpg')

#image = Image.open('./car\_road.jpg')

#filter\_ = Image.open('./car.jpg')

# Converts image to gray scale, so that it is easy to apply a filter

image = np.array(image.convert('L'))

filter\_ = np.array(filter\_.convert('L'))

result = correlation(image, filter\_)