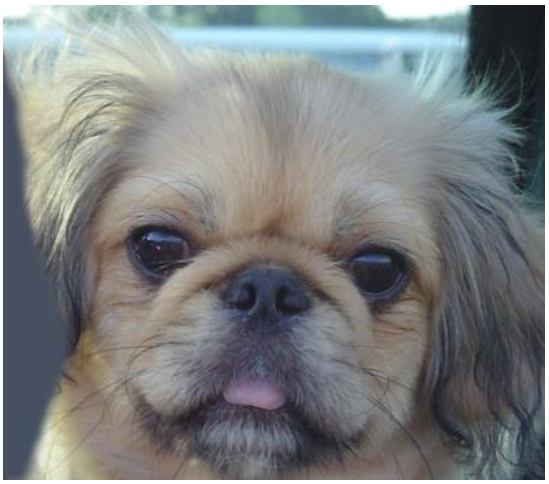
```
Python 3.6.2 (v3.6.2:5fd33b5926, Jul 16 2017, 20:11:06)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> WARNING: The version of Tcl/Tk (8.5.9) in use may be unstable.
Visit http://www.python.org/download/mac/tcltk/ for current information.

===== RESTART: /Users/gautamsoni/Desktop/CPSC 425/assignment1/a1_code.py =====
============================== Part 1.1======================================
>>> boxfilter(3)
array([[0.11111111, 0.11111111, 0.11111111],
       [0.11111111, 0.11111111, 0.11111111],
       [0.11111111, 0.11111111, 0.11111111]])

>>> boxfilter(4)
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    boxfilter(4)
  File "/Users/gautamsoni/Desktop/CPSC 425/assignment1/a1_code.py", line 25, in boxfilter
    raise exceptions.ArgumentError("Dimensions must be odd")
conda.exceptions.ArgumentError: Dimensions must be odd

>>> boxfilter(5)
array([[0.04, 0.04, 0.04, 0.04, 0.04],
       [0.04, 0.04, 0.04, 0.04, 0.04],
       [0.04, 0.04, 0.04, 0.04, 0.04],
       [0.04, 0.04, 0.04, 0.04, 0.04],
       [0.04, 0.04, 0.04, 0.04, 0.04]])

============================== Part 1.2======================================

>>> gauss1d(0.3)
array([0.00383626, 0.99232748, 0.00383626])

>>> gauss1d(0.5)
array([0.10650698, 0.78698604, 0.10650698])

>>> gauss1d(1)
array([0.00443305, 0.05400558, 0.24203623, 0.39905028, 0.24203623,
       0.05400558, 0.00443305])

>>> gauss1d(2)
array([0.0022182 , 0.00877313, 0.02702316, 0.06482519, 0.12110939,
       0.17621312, 0.19967563, 0.17621312, 0.12110939, 0.06482519,
       0.02702316, 0.00877313, 0.0022182 ])
============================== Part 1.3======================================
>>> gauss2d(0.5)
```

```
array([[0.01134374, 0.08381951, 0.01134374],
       [0.08381951, 0.61934703, 0.08381951],
       [0.01134374, 0.08381951, 0.01134374]])

>>> gauss2d(1)
array([[1.96519161e-05, 2.39409349e-04, 1.07295826e-03, 1.76900911e-03,
        1.07295826e-03, 2.39409349e-04, 1.96519161e-05],
       [2.39409349e-04, 2.91660295e-03, 1.30713076e-02, 2.15509428e-02,
        1.30713076e-02, 2.91660295e-03, 2.39409349e-04],
       [1.07295826e-03, 1.30713076e-02, 5.85815363e-02, 9.65846250e-02,
        5.85815363e-02, 1.30713076e-02, 1.07295826e-03],
       [1.76900911e-03, 2.15509428e-02, 9.65846250e-02, 1.59241126e-01,
        9.65846250e-02, 2.15509428e-02, 1.76900911e-03],
       [1.07295826e-03, 1.30713076e-02, 5.85815363e-02, 9.65846250e-02,
        5.85815363e-02, 1.30713076e-02, 1.07295826e-03],
       [2.39409349e-04, 2.91660295e-03, 1.30713076e-02, 2.15509428e-02,
        1.30713076e-02, 2.91660295e-03, 2.39409349e-04],
       [1.96519161e-05, 2.39409349e-04, 1.07295826e-03, 1.76900911e-03,
        1.07295826e-03, 2.39409349e-04, 1.96519161e-05]])

================================ Part 1.4=====================================
>>> dog_image = Image.open('/Users/gautamsoni/Desktop/CPSC 425/assignment1/dog.jpg')
>>> dog_image.show()
>>> dog_image_grey = dog_image.convert("L")
>>> dog_image_array = np.asarray(dog_image_grey)
>>> gauss_dog = gaussconvolve2d(dog_image_array, 7)
>>> new_dog = Image.fromarray(gauss_dog.astype('uint8'))
>>> new_dog.save('/Users/gautamsoni/Desktop/CPSC
425/assignment1/new_dog_coloured.jpg')
>>> new_dog.show()
```

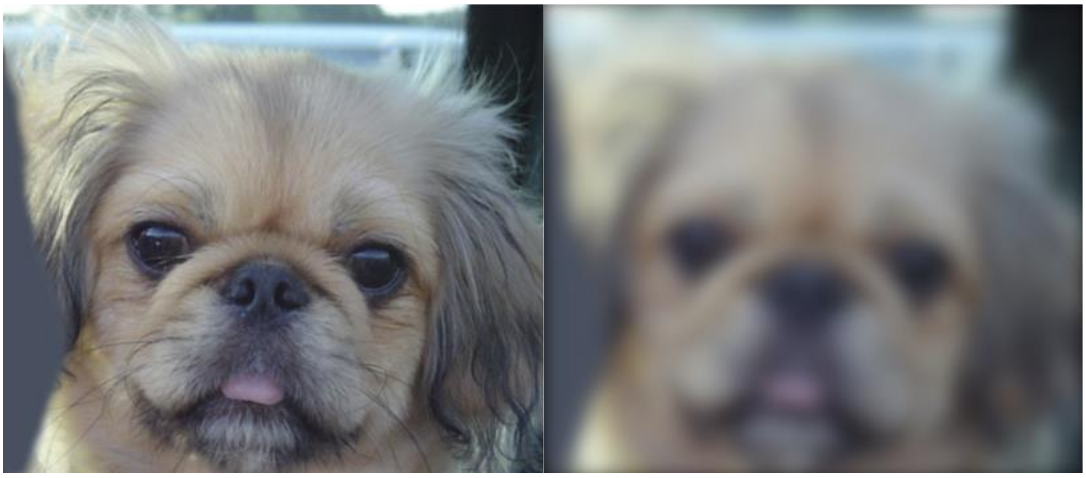


========== Part 1.5==========

With a seperable 2D Gaussian filter, there are 2m multiplications at each pixel (X,Y). Also there are n x n pixels in (X,Y). Hence there are 2m * n^2 multiplications.
However, the convolution can be sped up by taking a natural log on both sides. At the expense of two ln() and one exp() computations, multiplication is reduced to admission.

========== Part 2.1==========

```
dog_image = Image.open('/Users/gautamsoni/Desktop/CPSC 425/assignment1/dog.jpg')
dog_image.show()

# Split the image in 3 channels
r, g, b = dog_image.split()

# Convert each channel (R,G,B) to an array
b_array = np.asarray(b)
g_array = np.asarray(g)
r_array = np.asarray(r)

# Apply filter to each channel
b_gauss = gaussconvolve2d(b_array, 7)[:,:,np.newaxis]
g_gauss = gaussconvolve2d(g_array, 7)[:,:,np.newaxis]
r_gauss = gaussconvolve2d(r_array, 7)[:,:,np.newaxis]

# Compose the 3 channels to get a colored picture
new_blurr_dog = np.concatenate((r_gauss, g_gauss, b_gauss), axis=2)

blurr_dog_image = Image.fromarray(new_blurr_dog.astype('uint8'))
blurr_dog_image.show()
```



========== Part 2.2==========

```
cat_image = Image.open("/Users/gautamsoni/Desktop/CPSC
425/assignment1/hw1/0a_cat.bmp")
cat_image.show()

cat_image_array = np.asarray(cat_image)

# Split the image in 3 channels
r_cat, g_cat, b_cat = cat_image.split()

# Convert each channel (R,G,B) to an array
b_array_cat = np.asarray(b_cat)
g_array_cat = np.asarray(g_cat)
r_array_cat = np.asarray(r_cat)

# Apply filter to each channel
b_gauss_cat = gaussconvolve2d(b_array_cat, 7)[:,:,np.newaxis]
g_gauss_cat = gaussconvolve2d(g_array_cat, 7)[:,:,np.newaxis]
r_gauss_cat = gaussconvolve2d(r_array_cat, 7)[:,:,np.newaxis]

# Compose the 3 channels to get a colored picture
new_blurr_cat = np.concatenate((r_gauss_cat, g_gauss_cat, b_gauss_cat), axis=2)

new_blurr_cat_image = Image.fromarray(new_blurr_cat.astype('uint8'))
new_blurr_cat_image.show()

high_frequency_cat = np.subtract(cat_image_array,new_blurr_cat)

high_frequency_cat_image = Image.fromarray(high_frequency_cat.astype('uint8') + 128)
high_frequency_cat_image.show()
```

========== Part 2.3==========

```
# Add the low frequency and high frequency images to create a hybrid image
hybrid_image_array = np.add(high_frequency_cat, new_blurr_dog)

hybrid_image = Image.fromarray(hybrid_image_array.astype('uint8'))
hybrid_image.show()
```

==========

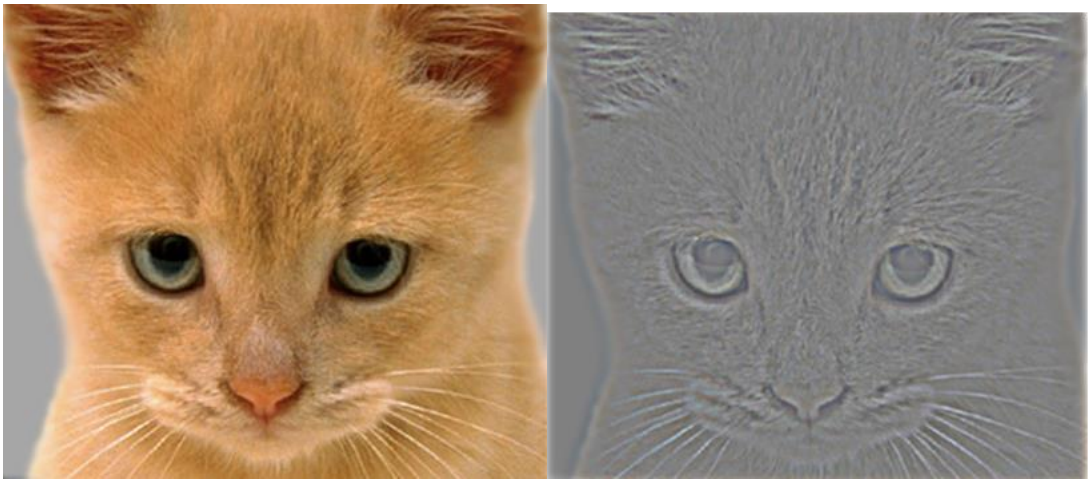
Part 2

Cat and Dog images

Sigma = 4

2.1

2.2



2.3

==========

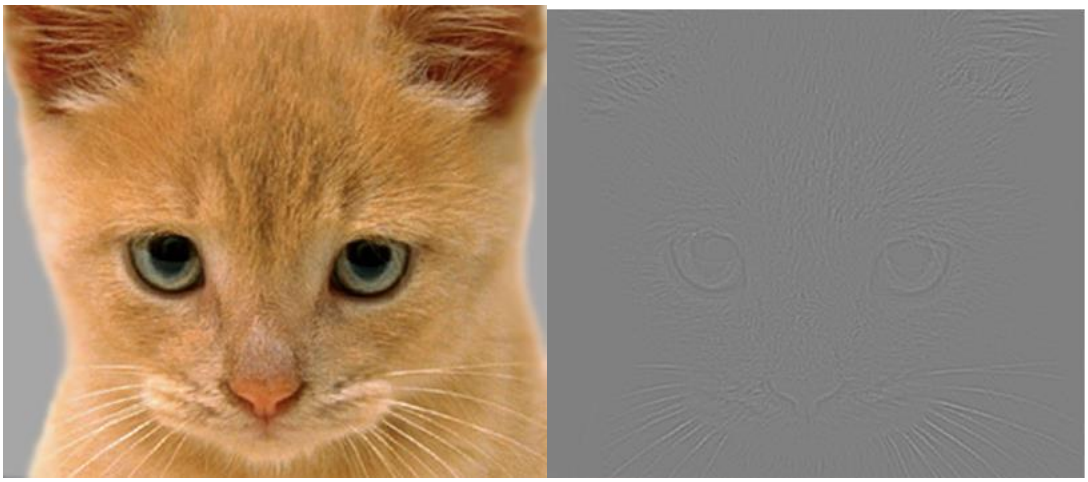
Part 2

Cat and Dog images

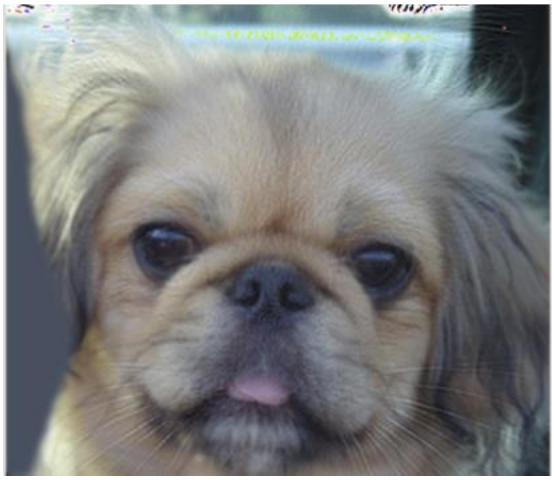
Sigma = 1

2.1

2.2



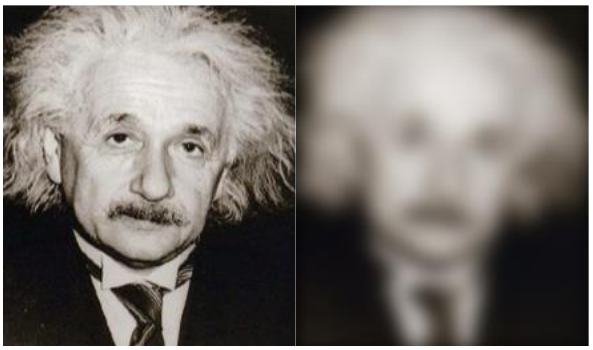
2.3



==========

Part 2
Einstein and Marilyn
Sigma = 7

2.1



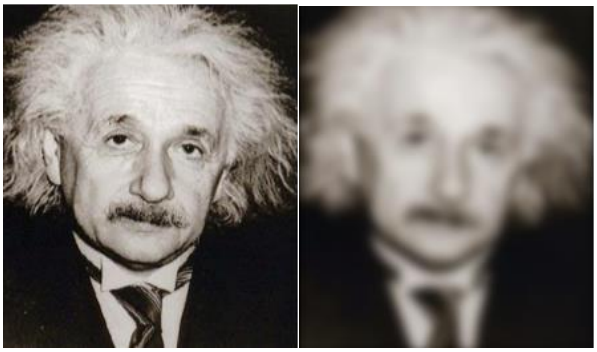
2.2

2.3

==========

Part 2
Einstein and Marilyn
Sigma = 4

2.1



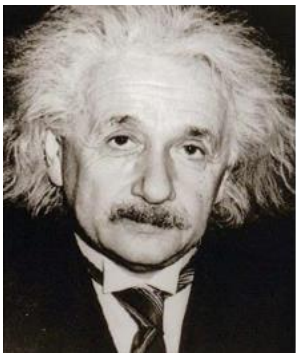
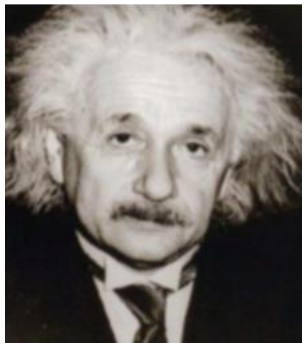
2.2

2.3

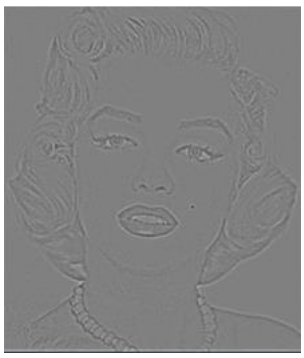
==========

Part 2
Einstein and Marilyn
Sigma = 1

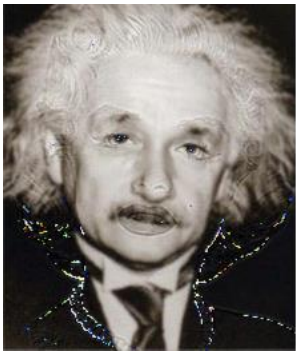
2.1

2.2

2.3



==========

Part 2

Bird and Plane

Sigma = 7

2.1


2.2


2.3

==========

Part 2

Bird and Plane

Sigma = 10

2.1


2.2

2.3

==========

Part 2

Bird and Plane

Sigma = 2

2.1

2.2

2.3