```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm

import cv2 as cv
```

```
sigma = 10
hw = 3*sigma

X, Y = np.meshgrid(np.arange(-hw, hw + 1, 1), np.arange(-hw, hw + 1, 1))
log = 1/(2*np.pi*sigma**2)*((X**2+Y**2)/sigma**2-2)*np.exp(-(X**2+Y**2)/(2*sigma**2)) # la

plt.imshow(log)

plt.show()
```

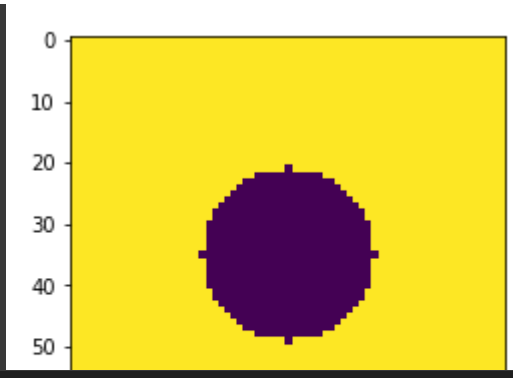
## Q2

```
w, h = 71, 71
hw = w//2
hh = h//2

f = np.ones((h, w), dtype=np.float32)*255
X, Y = np.meshgrid(np.arange(-hh, hh + 1, 1), np.arange(-hw, hw + 1, 1))

r = w//5
f *= X**2 + Y**2 > r**2

plt.imshow(f)
plt.show()
```

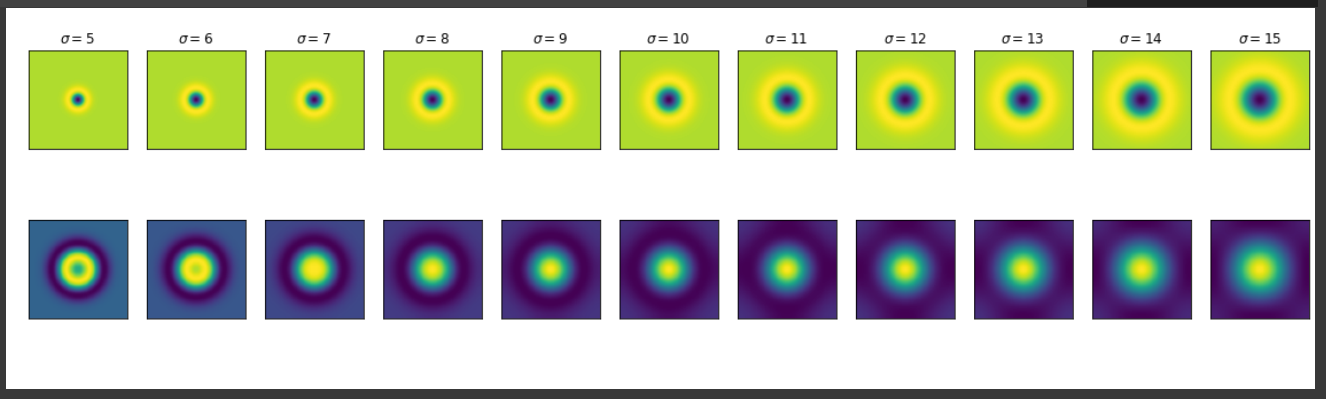



```
s = 11
fig, ax = plt.subplots(2, s, figsize=(20,5))

scale_space = np.empty((h, w, s), dtype=np.float32)

for i, sigma in enumerate(np.arange(5, 16, 1)):
  log_hw = 3*16
  X, Y = np.meshgrid(np.arange(-log_hw, log_hw + 1, 1), np.arange(-log_hw, log_hw + 1, 1))
  log = 1/(2*np.pi*sigma**2)*((X**2+Y**2)/sigma**2-2)*np.exp(-(X**2+Y**2)/(2*sigma**2)) #
  f_log = cv.filter2D(f, -1, log)
  scale_space[:, :, i] = f_log
  ax[0,i].imshow(log)
  ax[1,i].imshow(f_log)
  ax[0, i].set_xticks([])
  ax[0, i].set_yticks([])
  ax[1, i].set_xticks([])
  ax[1, i].set_yticks([])
  ax[0, i].set_title(f'$\sigma={sigma}$')

indices = np.unravel_index(np.argmax(scale_space, ))
```




# Fitting

## Q4

```
m = 2 # Line equation : y = m*x + c . m i s the s lope . c i s the int e r c ept .
c = 1
x = np.arange(1 ,10 , 1)
np.random.seed(45)
noise = 2.*np.random.randn( len( x ) )
```

```
o = np.zeros( x.shape )
# o[-1] = 20
y = m*x + c + noise + o

n = len(x)
X = np.concatenate([np.reshape(x, (n,1)), np.ones((n,1))], axis=1)
B = np.linalg.pinv(X.T@X)@X.T@y
mstar = B[0]
cstar = B[1]

plt.plot(x, y, '+', label="Noisy line")
plt.plot([x[0], x[-1]], [m*x[0]+c, m*x[-1]+c], label="True line")
plt.plot([x[0], x[-1]], [mstar*x[0]+cstar, mstar*x[-1]+cstar], label="Predicted line")
plt.legend()

plt.show()
```

```
from re import U
m = 2 # Line equation : y = m*x + c . m i s the s lope . c i s the int e r c ept .
c = 1
x = np.arange(1 ,10 , 1)
np.random.seed(45)
noise = 2.*np.random.randn( len( x ) )
o = np.zeros( x.shape )
# o[-1] = 20
y = m*x + c + noise + o

n = len(x)

u11 = np.sum((x-np.mean(x))**2)
u12 = np.sum((x-np.mean(x))*(y-np.mean(y)))
u21 = u12
u22 = np.sum((y-np.mean(y))**2)

U = np.array([[u11, u12], [u21, u22]])
W, V = np.linalg.eig(U)
ev_corresponding_to_smallest_ev = V[:, np.argmin(W)]

a = ev_corresponding_to_smallest_ev[0]
```

```
b = ev_corresponding_to_smallest_ev[1]
d = a*np.mean(x) + b*np.mean(y)

mstar = -a/b
cstar = d/b

plt.plot(x, y, '+', label="Noisy line")
plt.plot([x[0], x[-1]], [m*x[0]+c, m*x[-1]+c], label="True line")
plt.plot([x[0], x[-1]], [mstar*x[0]+cstar, mstar*x[-1]+cstar], label="Predicted line")
plt.legend()

plt.show()
```

```

```

✓ 0s completed at 3:08 PM