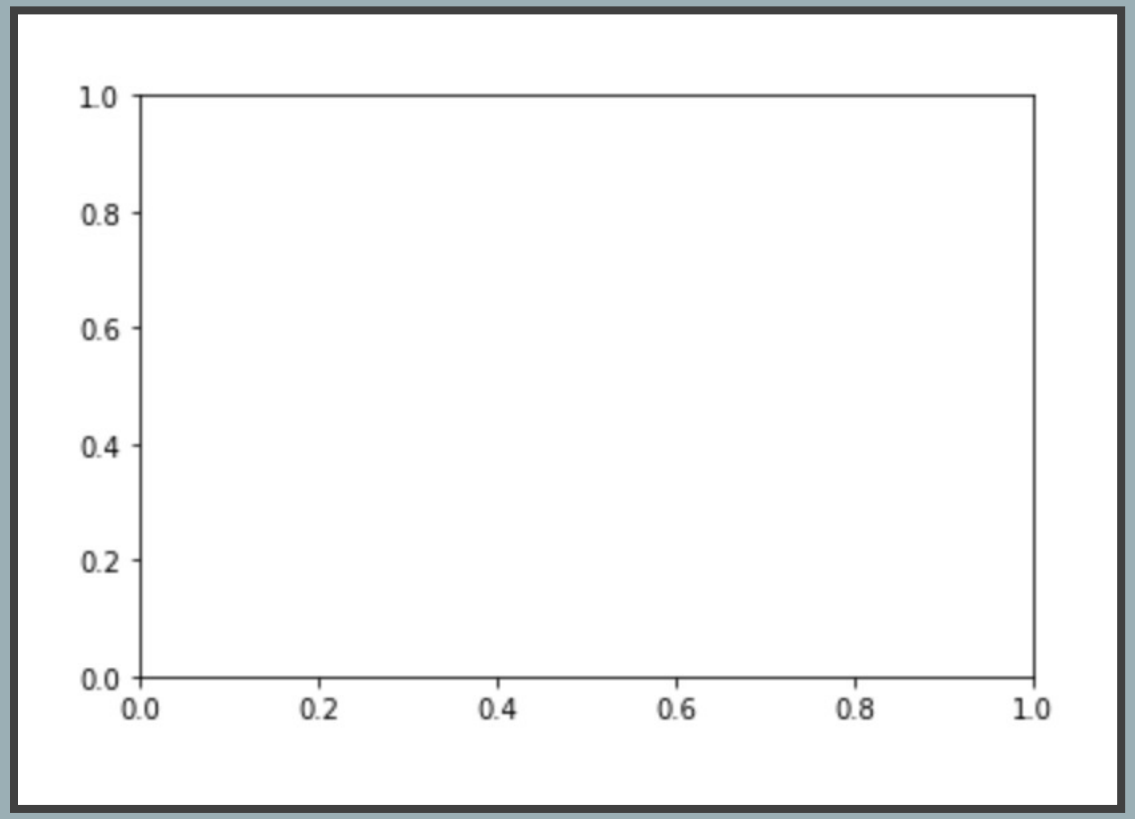
```
f = plt.figure()
```

=> Allocate space for plotting

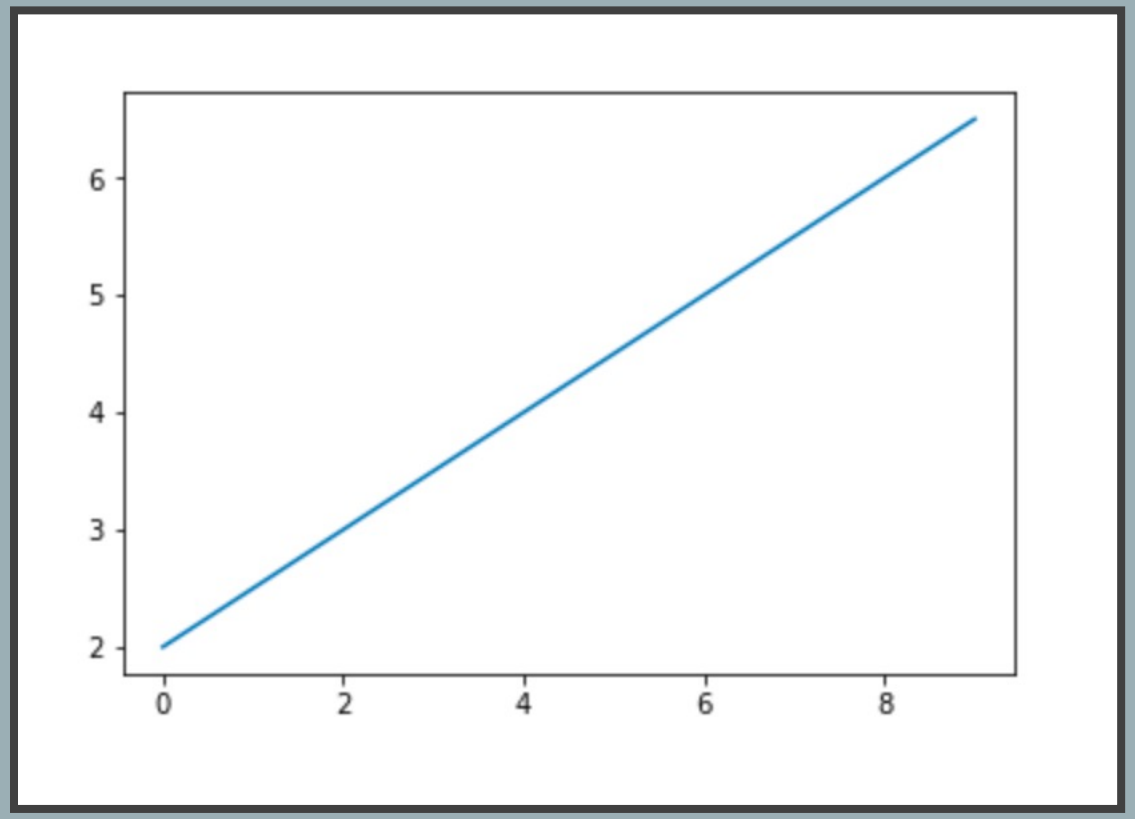



```
f = plt.figure()
```

=> Allocate space for plotting

```
ax = f.add_subplot(1, 1, 1)
```

=> Create an instance of an `axes` object




```
f = plt.figure()
```

=> Allocate space for plotting

```
ax = f.add_subplot(1, 1, 1)
```

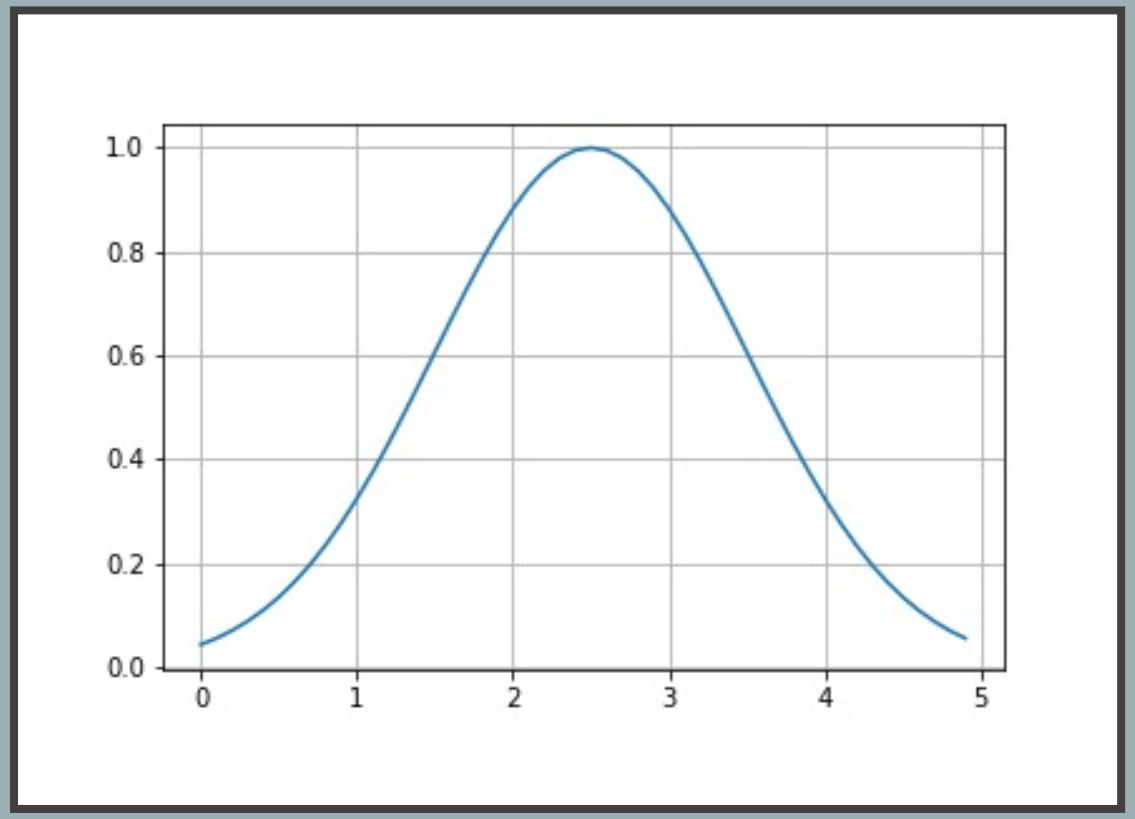
=> Create an instance of an `axes` object

```
ax.plot(x, y)
```

=> Plot x and y in the `axes` object

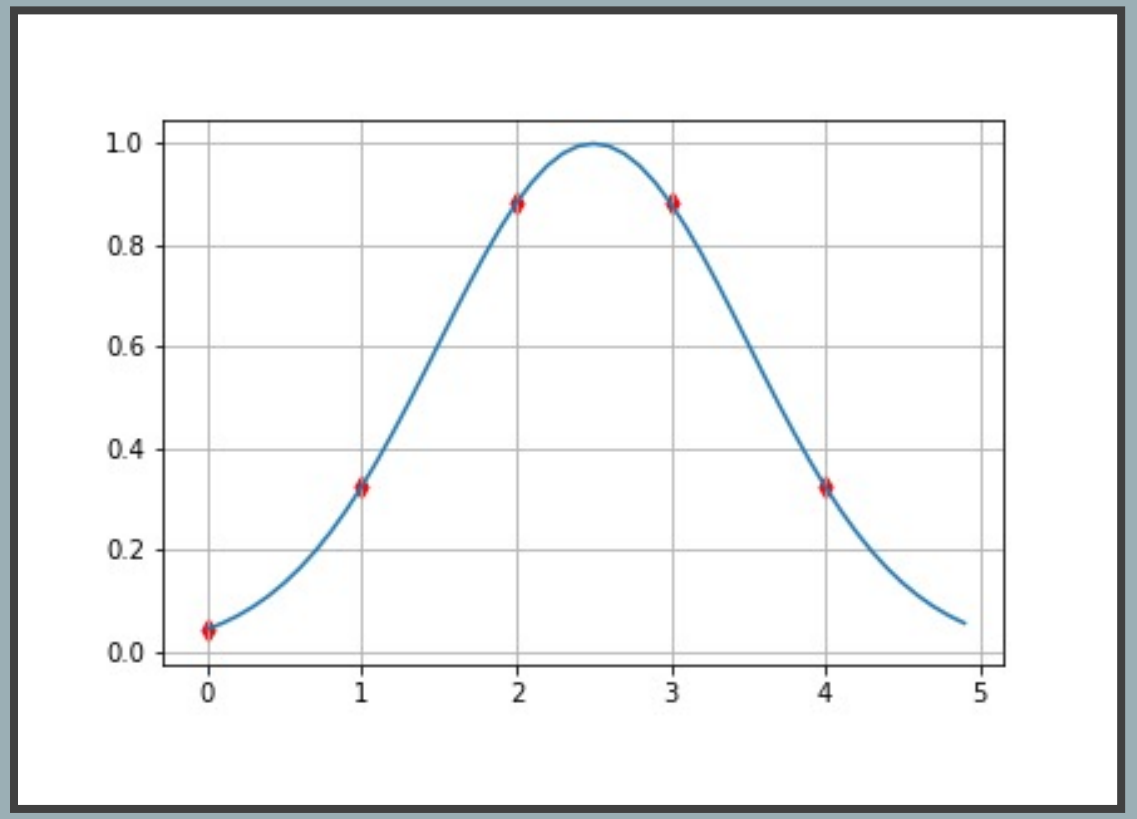
np.meshgrid(): What is it?

```
def gauss1D(x, I, xc, sigx):
    arg = (x - xc) / sigx
    return np.exp( -0.5 * arg**2)
```



np.meshgrid(): What is it?

```
def gauss1D(x, I, xc, sigx):
    arg = (x - xc) / sigx
    return np.exp( -0.5 * arg**2)
```



If you create a 1D array:

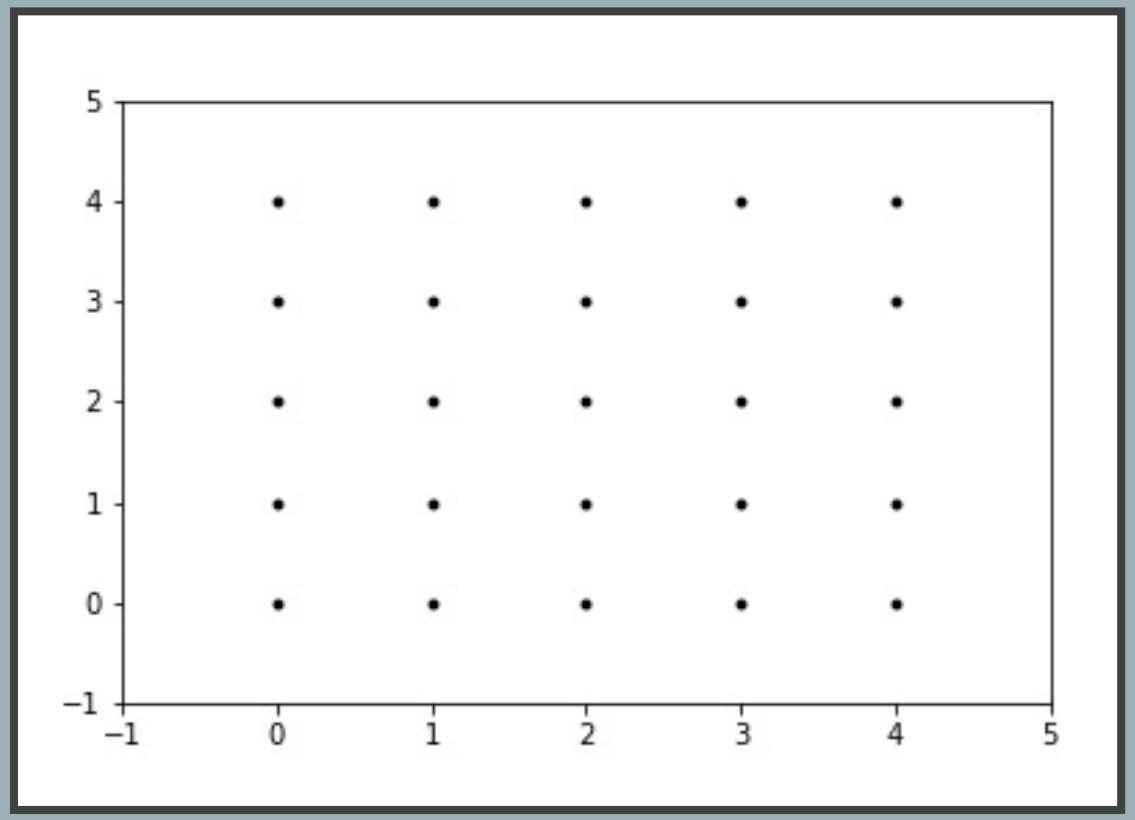
```
x = np.arange(0,  5, 1)
```

You sample your gaussian for x = 0,1…, 5

```
g1D = gauss1D(x, 1, 2.5, 1.)
ax.scatter(x, g1D, marker='d')
```

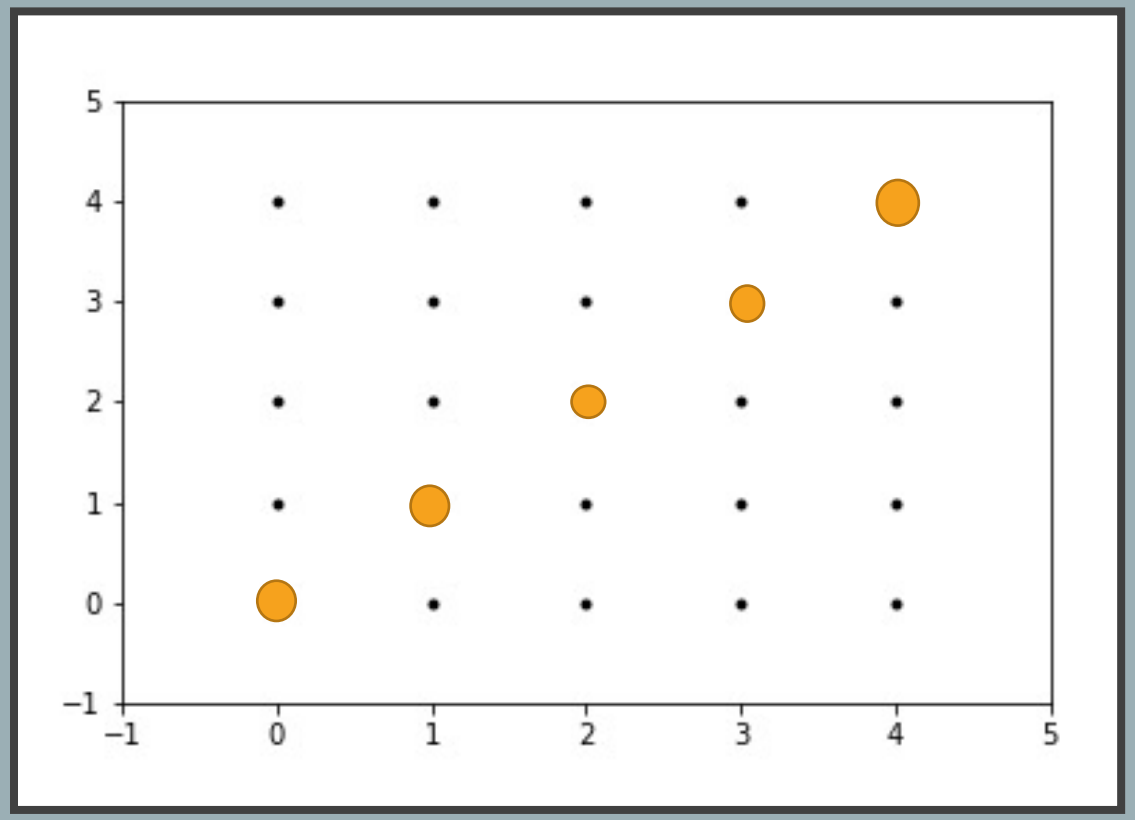
How to do with a 2D gaussian?

```
def gauss2D(x, y, I, xc, sigx, yc, sigy):
    argx = (x - xc) / sigx
    argy = (y - yc) / sigy
    g2D = np.exp(-0.5*(argx**2 + argy**2))
    return g2D
```



How to do with a 2D gaussian?

```
def gauss2D(x, y, I, xc, sigx, yc, sigy):
    argx = (x - xc) / sigx
    argy = (y - yc) / sigy
    g2D = np.exp(-0.5*(argx**2 + argy**2))
    return g2D
```



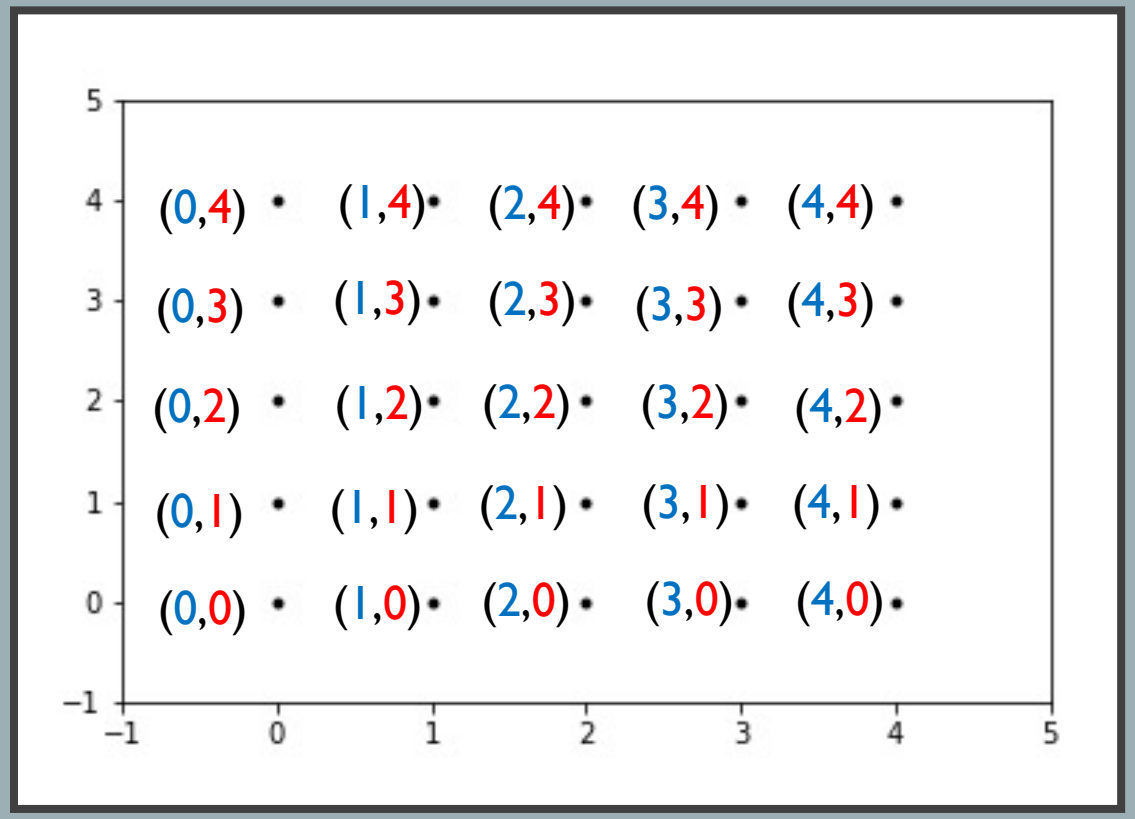
If you create two 1D array:

```
x = np.arange(0,  5, 1)
y = np.arange(0,  5, 1)
```

You sample your gaussian only along the diagonal [(0,0), (1,1), …, (4, 4)]

```
g2D = gauss2D(x, y, *args)
```

Meshgrid is the solution




```
def gauss2D(x, y, I, xc, sigx, yc, sigy):
    argx = (x - xc) / sigx
    argy = (y - yc) / sigy
    g2D = np.exp(-0.5*(argx**2 + argy**2))
    return g2D
```

You need a 2D array with coords. of (x,y):

```
x =   0 1 2 3 4        y =   0 0 0 0 0
      0 1 2 3 4              1 1 1 1 1
      0 1 2 3 4              2 2 2 2 2
      0 1 2 3 4              3 3 3 3 3
      0 1 2 3 4              4 4 4 4 4
```

```
X, Y = np.meshgrid(x, y)
g2D = gauss2D(X, Y, *args)
```