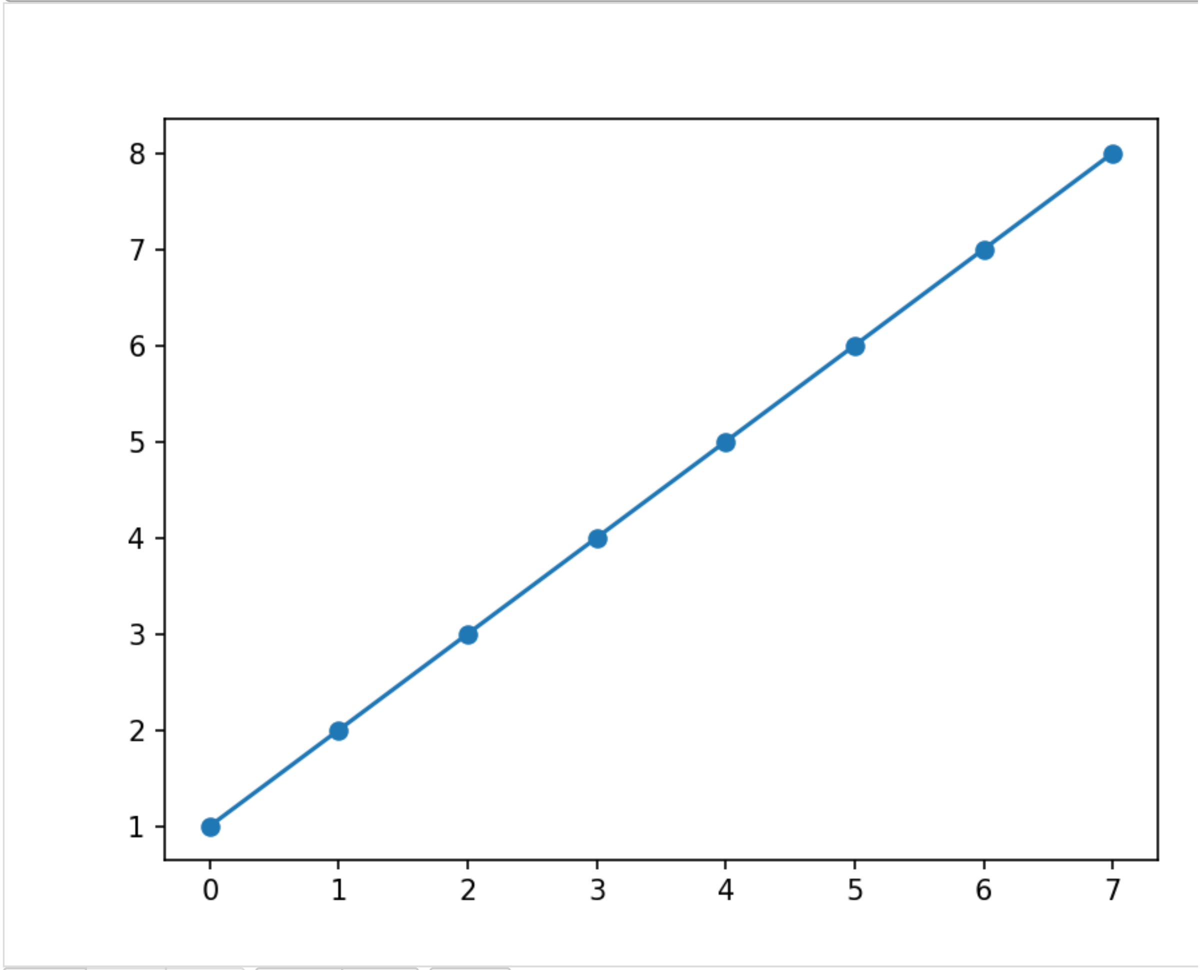
Subplots

In [1]:

```
import pandas as pd
import numpy as np
%matplotlib notebook
import matplotlib.pyplot as plt

x = np.array([1, 2, 3, 4, 5, 6, 7, 8])
plt.figure()
plt.plot(x, '-o')
```

Figure 1



Out[1]:

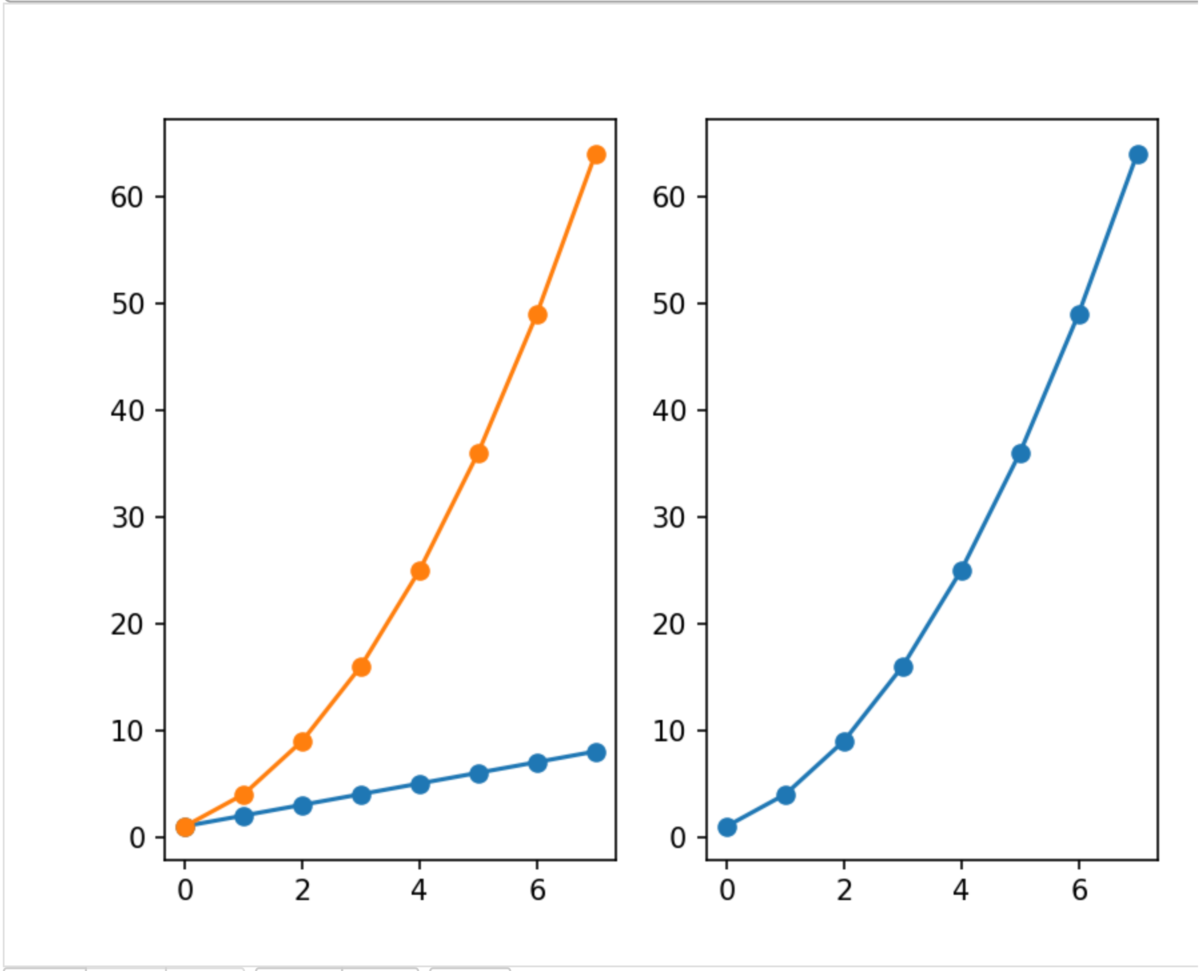
```
[<matplotlib.lines.Line2D at 0x20921f35f40>]
```

In [2]:

```
plt.figure()
plt.subplot(1, 2, 1)   #행, 열, 인덱스 => 첫번째 열만 사용하겠다는 것

linear_data = x
plt.plot(linear_data, '-o')
```

Figure 2



Downlc

Out[2]:

```
[<matplotlib.lines.Line2D at 0x20923706430>]
```

In [3]:

```
exponential_data = linear_data**2

plt.subplot(1, 2, 2)
plt.plot(exponential_data, '-o')
```

Out[3]:

```
[<matplotlib.lines.Line2D at 0x20923cb4400>]
```

In [4]:

```
plt.subplot(1, 2, 1)
plt.plot(exponential_data, '-o')
```

```
<ipython-input-4-3c3bc4898854>:1: MatplotlibDeprecationWarning: Adding an axes using
the same arguments as a previous axes currently reuses the earlier instance.  In a f
uture version, a new instance will always be created and returned.  Meanwhile, this
warning can be suppressed, and the future behavior ensured, by passing a unique labe
l to each axes instance.
  plt.subplot(1, 2, 1)
```

Out[4]:

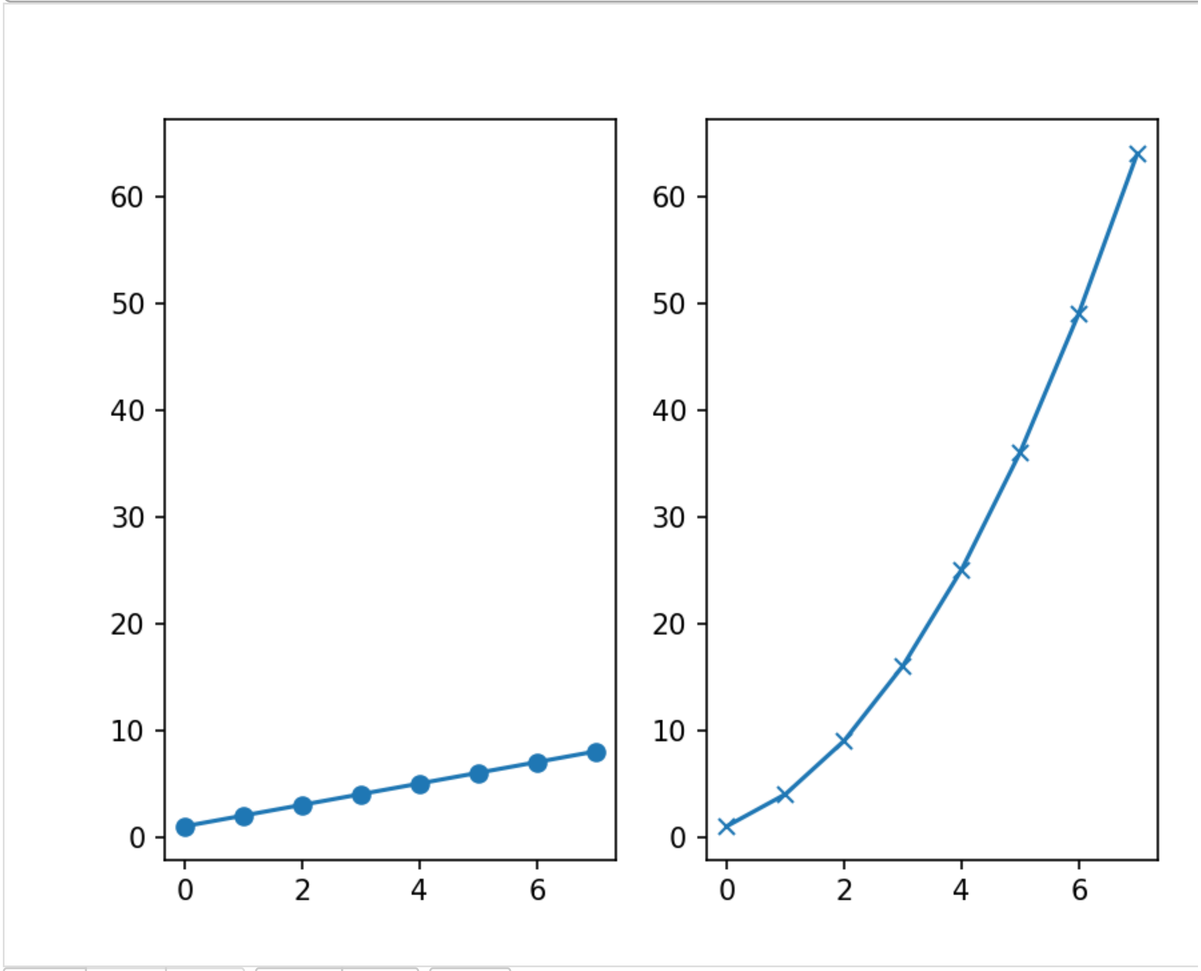
```
[<matplotlib.lines.Line2D at 0x2092446ce80>]
```

In [7]:

```
plt.figure()
ax1 = plt.subplot(1, 2, 1)
plt.plot(linear_data, '-o')

ax2 = plt.subplot(1, 2, 2, sharey = ax1)
plt.plot(exponential_data, '-x')
```

Figure 3



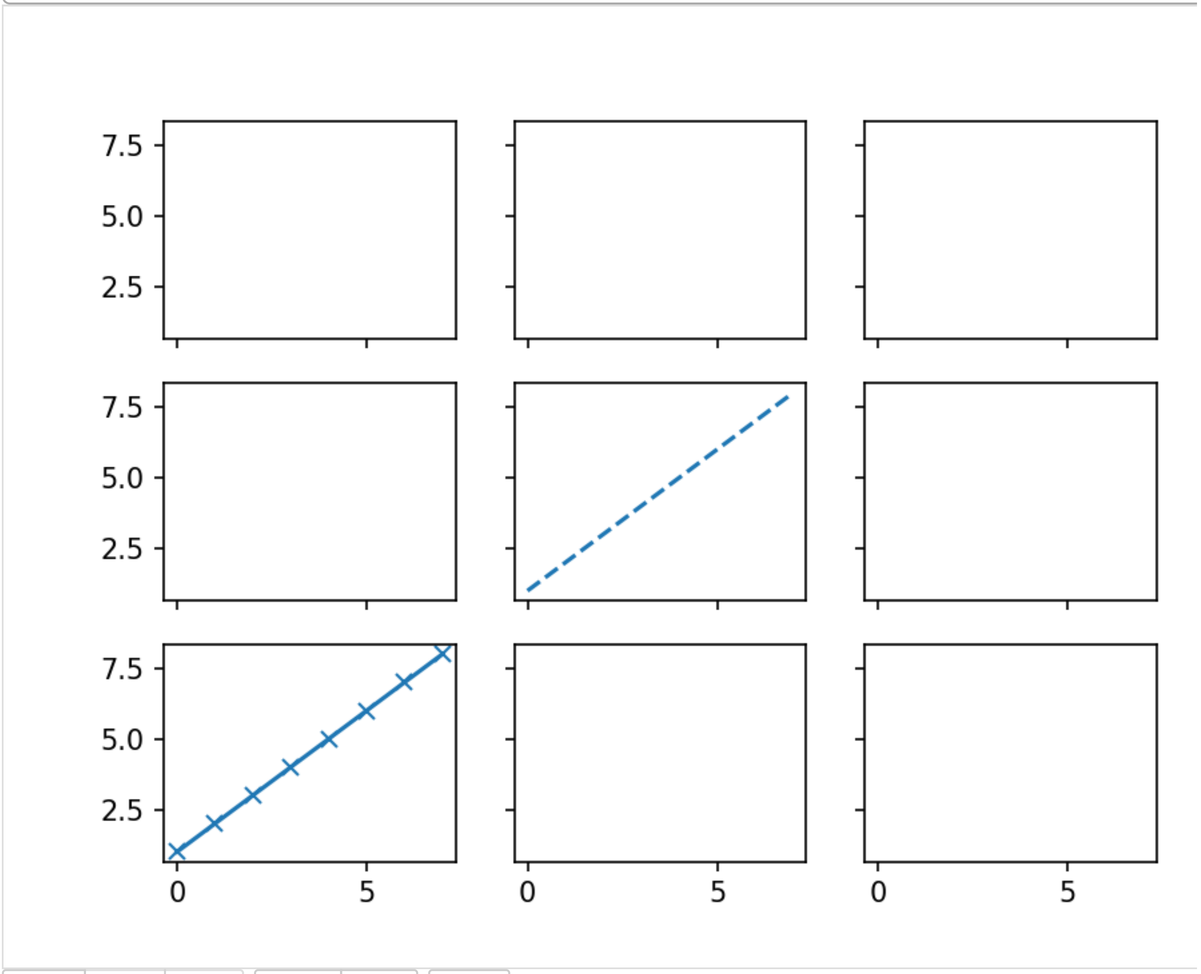
Out[7]:

```
[<matplotlib.lines.Line2D at 0x20923dca370>]
```

In [8]:

```
fig, ((ax1, ax2, ax3), (ax4, ax5, ax6), (ax7, ax8, ax9)) = plt.subplots(3, 3, sharex = True, sharey
```

Figure 4



In [9]:

```
ax5.plot(linear_data, '--')
```

Out[9]:

```
[<matplotlib.lines.Line2D at 0x20924a7a100>]
```

In [10]:

```
ax7.plot(linear_data, '-x')
```

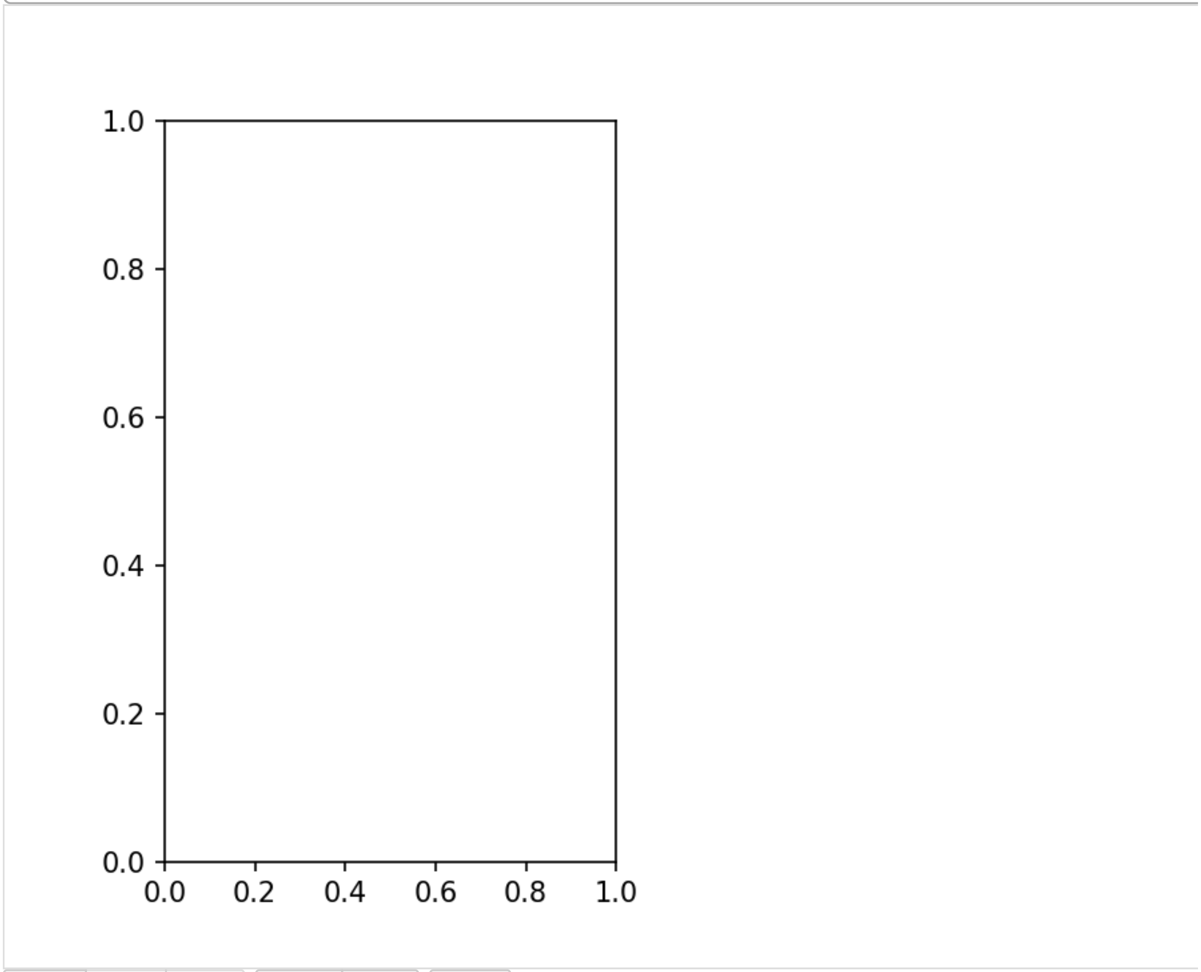
Out[10]:

```
[<matplotlib.lines.Line2D at 0x20924ad9cd0>]
```

In [11]:

```
plt.figure()
plt.subplot(1, 2, 1) == plt.subplot(121)
```

Figure 5



```
<ipython-input-11-01d80671c68d>:2: MatplotlibDeprecationWarning: Adding an axes usin
g the same arguments as a previous axes currently reuses the earlier instance.  In a
future version, a new instance will always be created and returned.  Meanwhile, this
warning can be suppressed, and the future behavior ensured, by passing a unique labe
l to each axes instance.
  plt.subplot(1, 2, 1) == plt.subplot(121)
```

Out[11]:

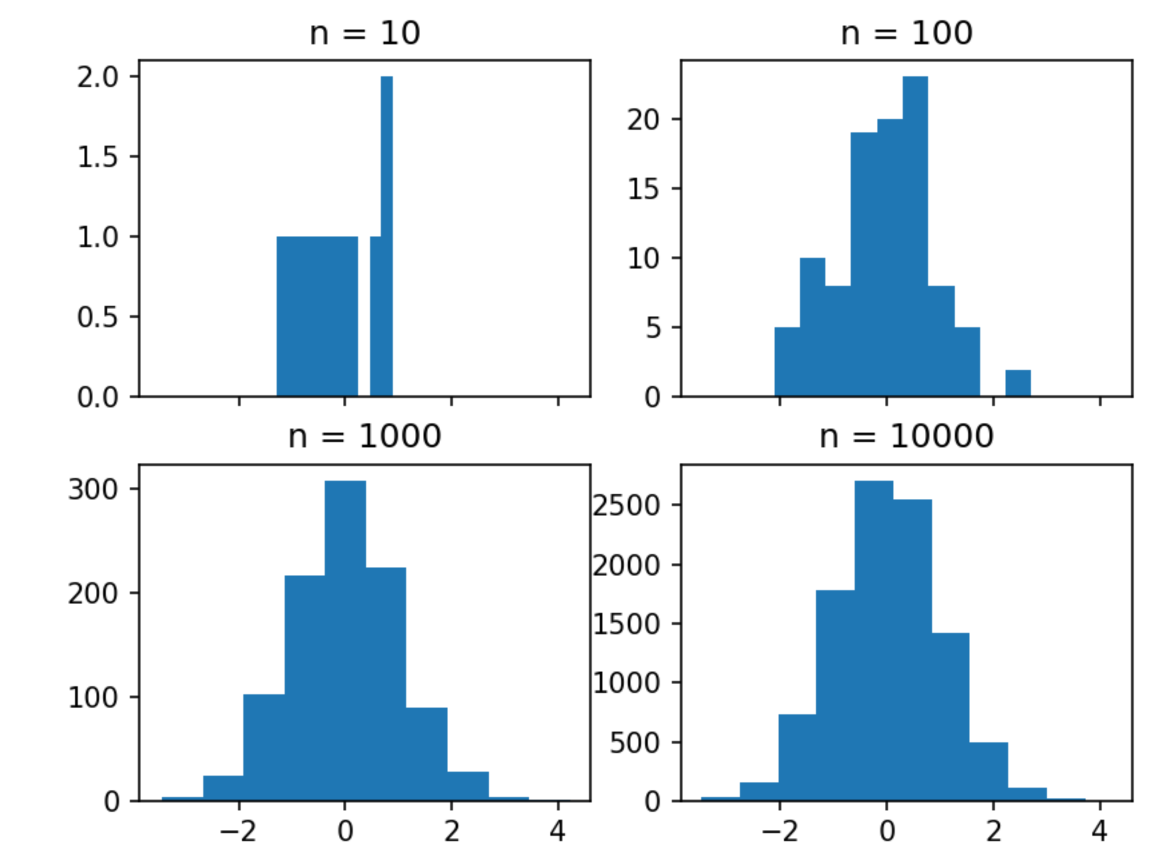
```
True
```

Histograms

In [16]:

```
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, sharex = True)
axs = [ax1, ax2, ax3, ax4]
```

Figure 6



In [17]:

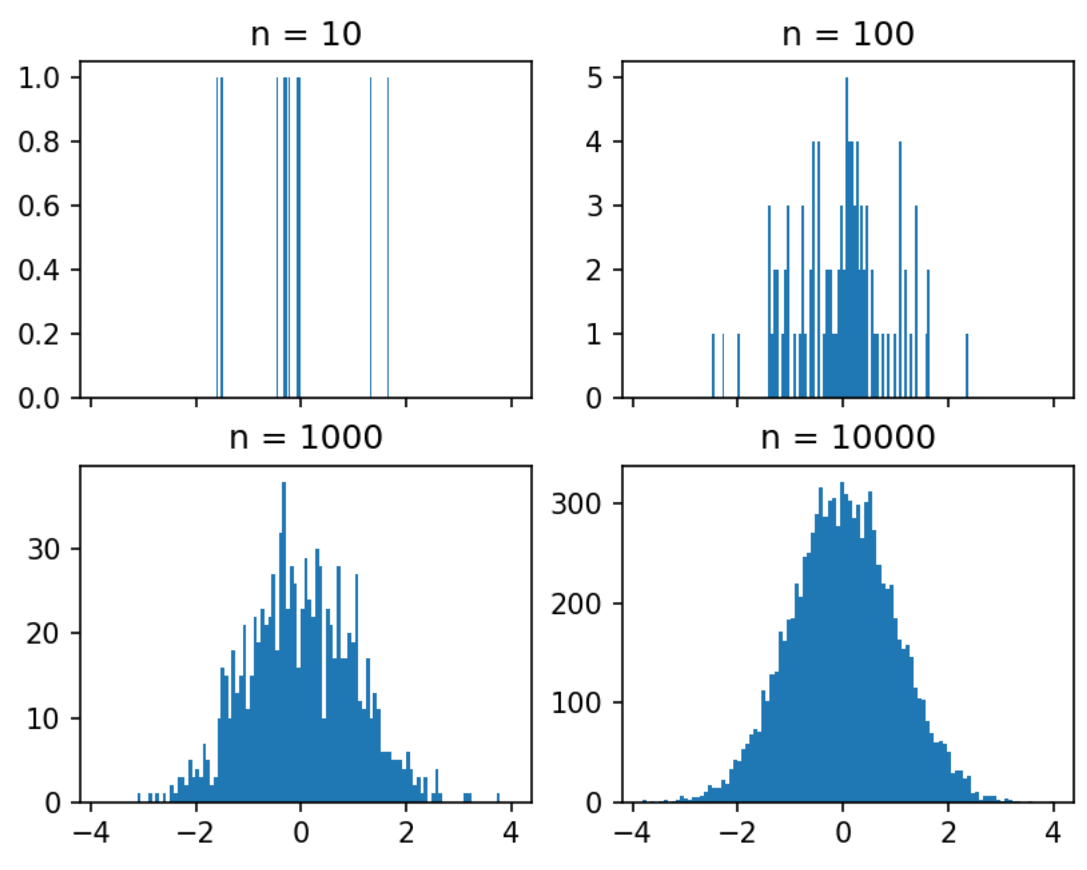
```
for n in range(0, len(axs)):
    sample_size = 10**(n + 1)
    sample = np.random.normal(loc = 0.0, scale = 1.0, size = sample_size)
    axs[n].hist(sample)
    axs[n].set_title('n = {}'.format(sample_size))
```

In [18]:

```
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, sharex = True)
axs = [ax1, ax2, ax3, ax4]

for n in range(0, len(axs)):
    sample_size = 10**(n + 1)
    sample = np.random.normal(loc = 0.0, scale = 1.0, size = sample_size)
    axs[n].hist(sample, bins = 100)    #bins == 구간 => 100마다 구간 자른 것
    axs[n].set_title('n = {}'.format(sample_size))
```

Figure 7



x=4.38

In [19]:

```
sample
```

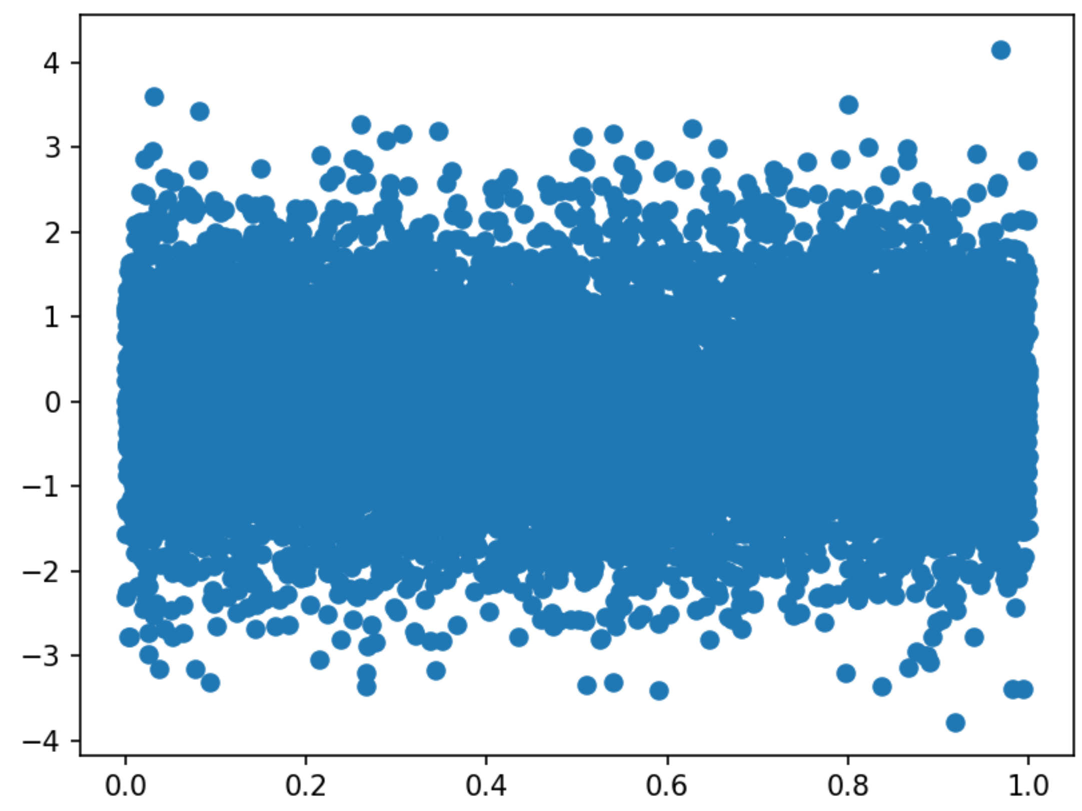
Out[19]:

```
array([-0.63679911,  0.69843968, -0.85573477, ..., -0.50086997,
        0.88027581, -0.19821531])
```

In [20]:

```
plt.figure()
Y = np.random.normal(loc = 0.0, scale = 1.0, size = 10000) #loc = 평균, scale = 분산
X = np.random.random(size = 10000)
plt.scatter(X, Y)
```

Figure 8



Out[20]:

```
<matplotlib.collections.PathCollection at 0x20928fbf2e0>
```

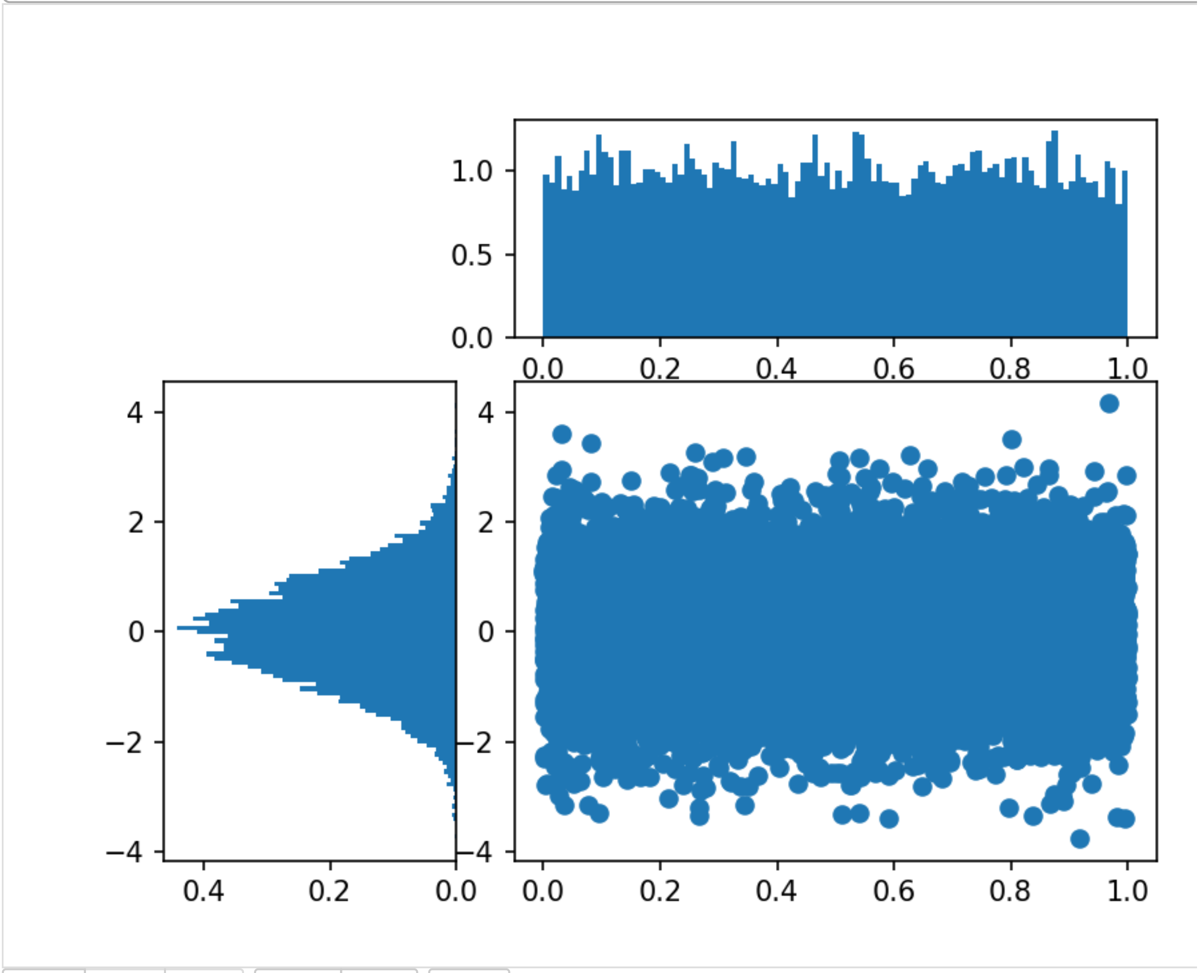
In [21]:

```
import matplotlib.gridspec as gridspec

plt.figure()
gspec = gridspec.GridSpec(3, 3)

top_histogram = plt.subplot(gspec[0,1:])    #첫번째 row에서 두번째컬럼부터 쭉 사용하겠다는 뜻
side_histogram = plt.subplot(gspec[1:,0])   #두번째 로우에서 세번째로우, 첫번째 컬럼에 그리겠다
lower_right = plt.subplot(gspec[1:,1:])     #두번째 로우 세번째로우, 두번째컬럼 세번째 컬럼에 그리겠
```

Figure 9



In [22]:

```
lower_right.scatter(X, Y)
top_histogram.hist(X, bins = 100)
s = side_histogram.hist(Y, bins = 100, orientation = 'horizontal')   #vertical로 있던것을 horizonta
```

In [25]:

```
top_histogram.clear()    #그래프 지우는 것
top_histogram.hist(X, bins = 100, density = True)
side_histogram.clear()
side_histogram.hist(Y, bins = 100, orientation = 'horizontal', density = True)
side_histogram.invert_xaxis()    #좌우반전
```