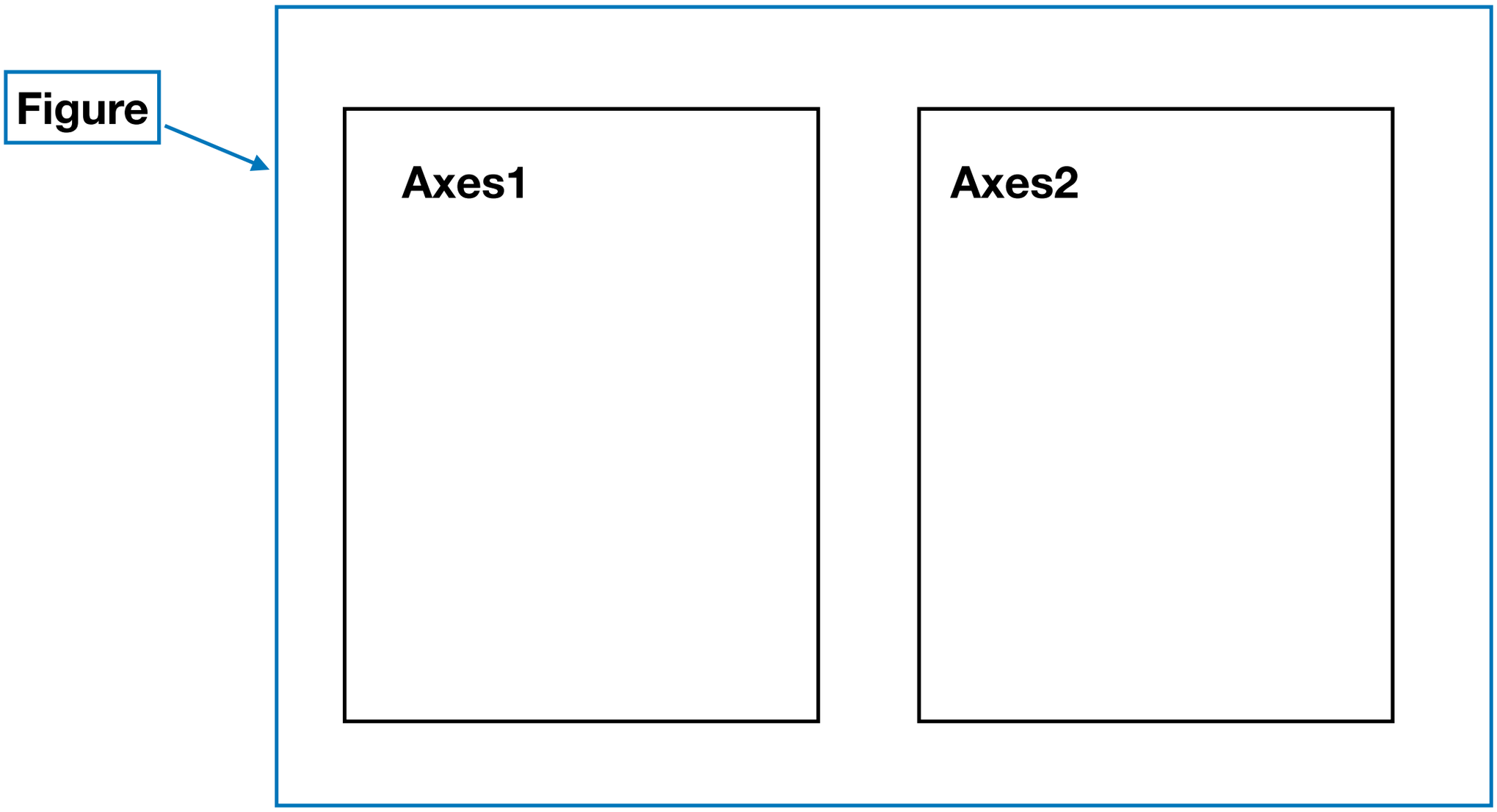
Matplotlib - intermediate

Juyong Lee

Two ways to plot

- Matplotlib에는 plot을 할 수 있는 2가지 방식이 있다.
- Object-based
    - Figure와 Axis를 각각 객체로 지정해서 조정할 수 있다.
    - Customize가 용이하다.
- Statement-based
    - pyplot의 method를 이용해서 현재 figure와 axis를 조정한다.
        - ex: *plt.method()*
    - 직관적이나 복잡한 plot을 그릴 때는 customization이 쉽지 않은 경우들이 발생한다.

Figure, Axes



- Matplotlib에는 가장 기본적으로 **figure**와 **axes**의 개념이 존재한다.

Figure class

- **Figure object**는 다음과 같이 생성할 수 있다.
  - ex) fig = plt.figure(1, figsize=(8,8))
    - 첫번째 숫자는 figure number
      - 한 스크립에서 여러개의 독립적인 그림파일을 생성해야할 때 사용한다.
    - *figsize* option은 인치로 x, y 사이즈를 의미한다.

Axes

- Axes를 생성할 때는 *figure* object의 *add_subplot* method를 사용한다.
- ex: *ax = fig.add_subplot(2,2)*
- 이때 리턴되는 값은 생성된 axes 클래스의 리스트이다.
- 위의 예시에서 4개의 axes가 return된다.
    - ax[0], ax[1], ax[2], ax[3]와 같은 식으로 접근 가능

Axes에 그림그리기

- axes 클래스는 대부분의 plotting method를 사용할 수 있다.
- 일반적으로 physical & chemical data 분석에서 많이 사용되는 plot은 다음과 같다.
- ax.plot() - 기본적인 line plot
- ax.scatter() - scatter plot
- ax.hist() - histogram 만들기
- ax.errorbar() - errorbar가 있는 그래프 만들기
- ax.semilogy(), ax.semilogx(), ax.loglog() - X축 혹은 Y축이 log scale인 그래프 그리기
- 이 외에 대부분의 plotting method를 axes의 method형태로 사용할 수 있다.

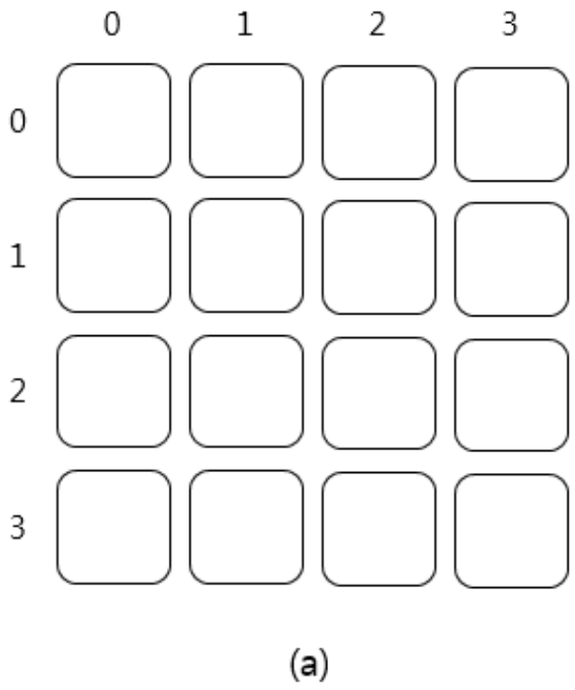
Axes 조정하기

- X축과 Y축의 최대/최소 값을 지정하기
  - ax.set_xlim(xmin, xmax) & ax.set_ylim(ymin, ymax)
- X축과 Y축의 label 지정하기
  - ax.set_xlabel(‘text’, fontsize=‘large’) & ax.set_ylabel(‘text’, fontsize=‘x-large’)
- X축과 Y축의 tick 조정하기
  - ex: ax.set_xticks(*[list of xticks]*) & ax.set_yticks(*[list of yticks]*)
- Legend 표시하기
  - ax.legend()

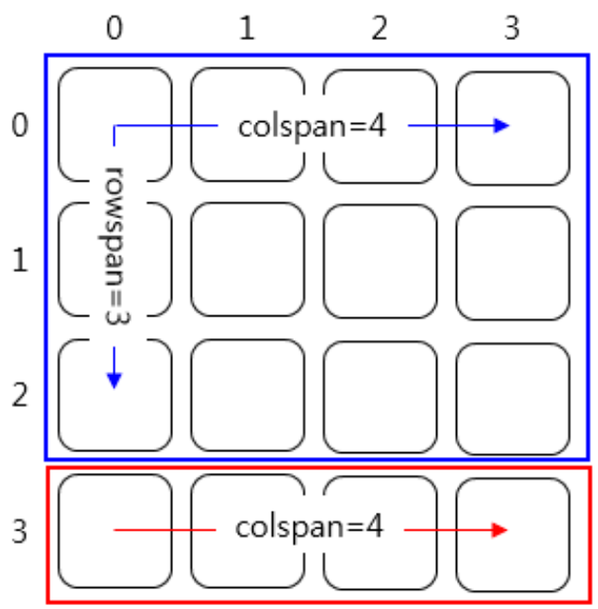
subplot2grid

- Subplot2grid method는 아래와 같이 격자 행태로 subplot을 생성한다.
- 첫번째 인자는 격자의 사이즈
- 두번째 인자는 시작되는 격자의 위치
- rowspan과 colspan은 행과 열의 개수를 의미

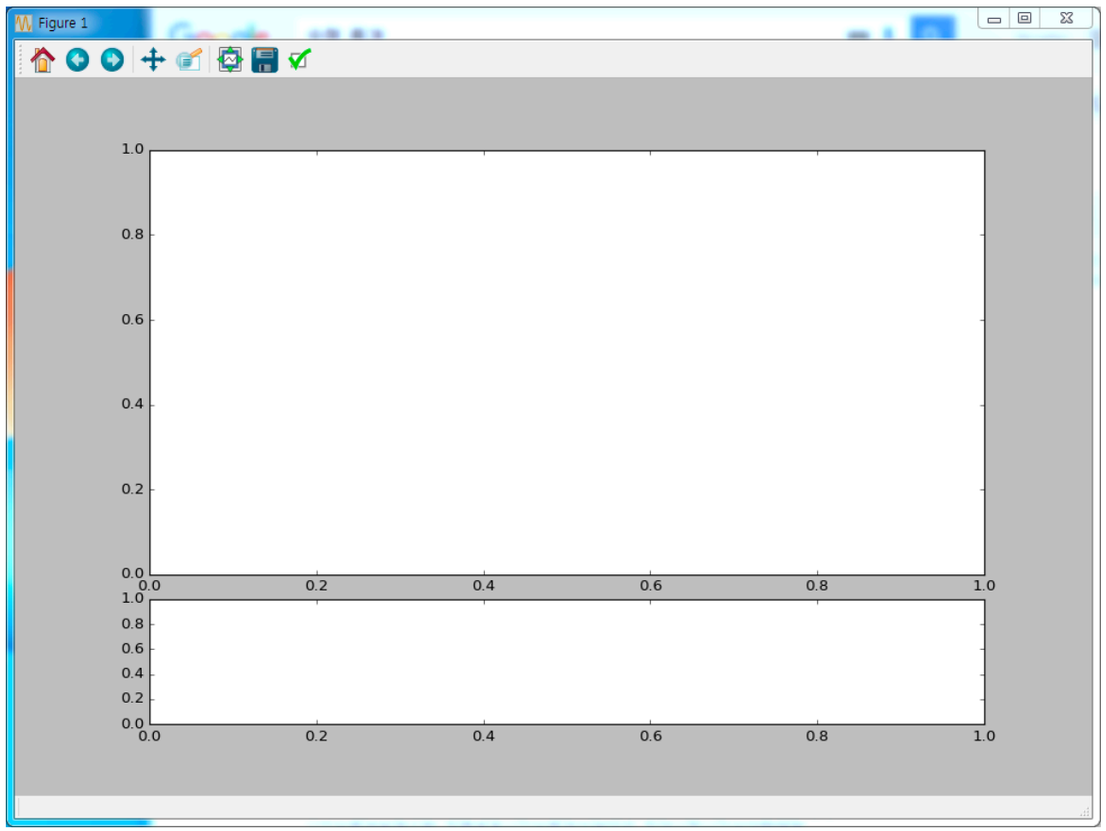
```
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(12, 8))
top_axes = plt.subplot2grid((4,4), (0,0), rowspan=3, colspan=4)
bottom_axes = plt.subplot2grid((4,4), (3,0), rowspan=1, colspan=4)

plt.show()
```



(a)

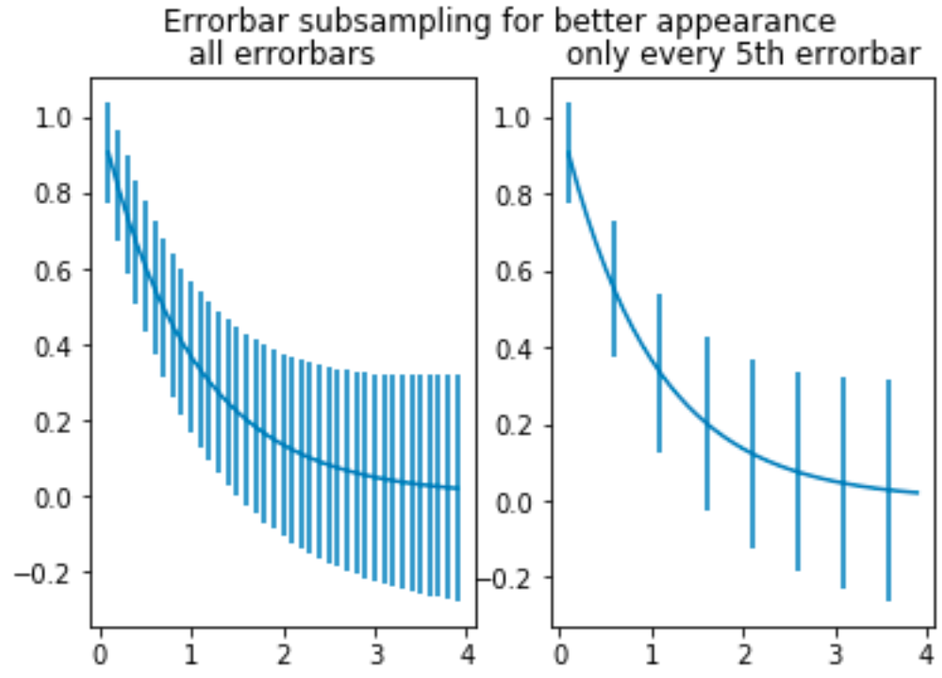


(b)



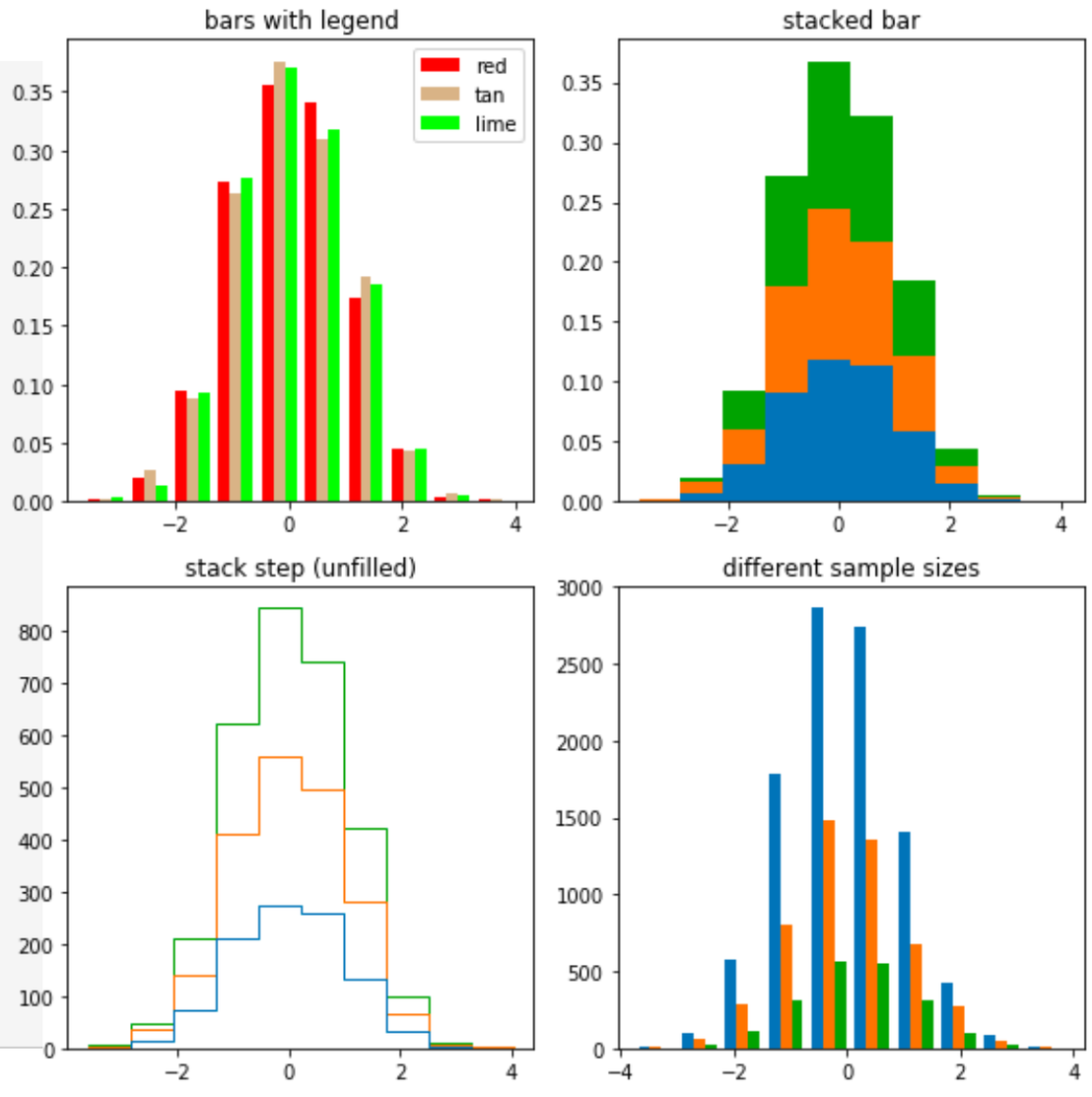
Errorbar example

```
### Errorbar graphs ###
# example data
x = np.arange(0.1, 4, 0.1)
y = np.exp(-x)

# example variable error bar values
yerr = 0.1 + 0.1 * np.sqrt(x)

# Now switch to a more OO interface to exercise more features.
fig, axs = plt.subplots(nrows=1, ncols=2)
ax = axs[0]
ax.errorbar(x, y, yerr=yerr)
ax.set_title('all errorbars')

ax = axs[1]
ax.errorbar(x, y, yerr=yerr, errorevery=5)
ax.set_title('only every 5th errorbar')

fig.suptitle('Errorbar subsampling for better appearance')

plt.show()
```



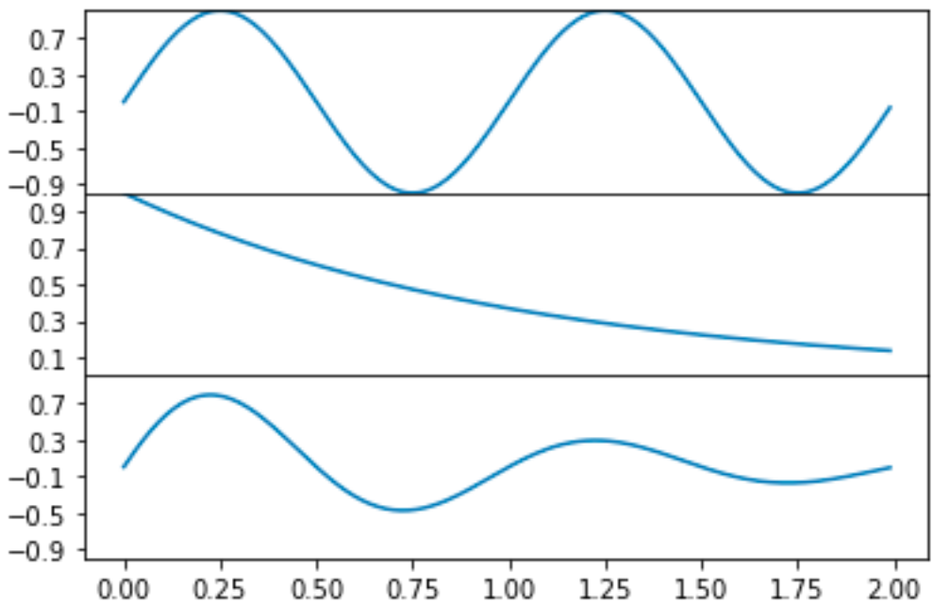
Histogram example

```
### Various histograms ###
np.random.seed(19680801)

n_bins = 10
x = np.random.randn(1000, 3)

fig = plt.figure(figsize=(8,8))

axes = fig.subplots(nrows=2, ncols=2)
ax0, ax1, ax2, ax3 = axes.flatten()

colors = ['red', 'tan', 'lime']
ax0.hist(x, n_bins, density=True, histtype='bar', color=colors, label=colors)
ax0.legend(prop={'size': 10})
ax0.set_title('bars with legend')

ax1.hist(x, n_bins, density=True, histtype='bar', stacked=True)
ax1.set_title('stacked bar')

ax2.hist(x, n_bins, histtype='step', stacked=True, fill=False)
ax2.set_title('stack step (unfilled)')

# Make a multiple-histogram of data-sets with different length.
x_multi = [np.random.randn(n) for n in [10000, 5000, 2000]]
ax3.hist(x_multi, n_bins, histtype='bar')
ax3.set_title('different sample sizes')

plt.tight_layout()
plt.show()
```



sharex

In [9]:
```
### Adjacent subplots ###

t = np.arange(0.0, 2.0, 0.01)

s1 = np.sin(2 * np.pi * t)
s2 = np.exp(-t)
s3 = s1 * s2

fig, axs = plt.subplots(3, 1, sharex=True)
# Remove horizontal space between axes
fig.subplots_adjust(hspace=0)

# Plot each graph, and manually set the y tick values
axs[0].plot(t, s1)
axs[0].set_yticks(np.arange(-0.9, 1.0, 0.4))
axs[0].set_ylim(-1, 1)

axs[1].plot(t, s2)
axs[1].set_yticks(np.arange(0.1, 1.0, 0.2))
axs[1].set_ylim(0, 1)

axs[2].plot(t, s3)
axs[2].set_yticks(np.arange(-0.9, 1.0, 0.4))
axs[2].set_ylim(-1, 1)

plt.show()
```



왼쪽과 오른쪽에 다른 y축 사용하기

```python
# Create some mock data
t = np.arange(0.01, 10.0, 0.01)
data1 = np.exp(t)
data2 = np.sin(2 * np.pi * t)

fig, ax1 = plt.subplots()

color = 'tab:red'
ax1.set_xlabel('time (s)')
ax1.set_ylabel('exp', color=color)
ax1.plot(t, data1, color=color)
ax1.tick_params(axis='y', labelcolor=color)

ax2 = ax1.twinx()  # instantiate a second axes that shares the same x-axis

color = 'tab:blue'
ax2.set_ylabel('sin', color=color)  # we already handled the x-label with ax1
ax2.plot(t, data2, color=color)
ax2.tick_params(axis='y', labelcolor=color)

fig.tight_layout()  # otherwise the right y-label is slightly clipped
plt.show()
```