**파이썬의 대표적 데이터 시각화 모듈 - Matplotlib**[**¶**](#gjdgxs)

많은 내용들이 Matplotlib 모듈에 포함 되어 있지만, 우리는 주로 pyplot이라는 시각화 도구를 제일 많이 사용합니다. 먼저 import 부터 하겠습니다.

In [1]:

**import** **numpy** **as** **np**  
**import** **matplotlib.pyplot** **as** **plt** *# pyplot 함수 불러와서 plt로 앨리어스*  
%**matplotlib** inline

In [2]:

plt.figure  
plt.plot([1,2,3,4,5,6,7,8,9,8,7,6,5,4,3,2,1,0])  
plt.show()

아주 간단하게 그래프를 그려 보았습니다. plot 함수에는 그래프에 표현될 데이터들이 들어가고, 각 x축과 y축은 임의로 지정되어서 숫자가 표시됩니다.

조금 더 복잡하게 삼각함수 그래프를 그려보겠습니다. 물론 numpy를 활용합니다

In [9]:

t = np.arange(0,12,0.01) *# 0부터 12 까지 0.01 간격으로 데이터 생성하기*  
y = np.sin(t) *# 발생시킨 데이터를 sin값으로 사용하기*

In [10]:

plt.figure(figsize=(10,6)) *# figsize: 그림 크기*  
plt.plot(t,y) *#사용할 데이터 세팅하기*  
plt.show()

각종 옵션으로 grid나 x축, y축 라벨을 붙여 보겠습니다.

In [40]:

plt.figure(figsize=(10, 6))  
plt.plot(t, y) *# 사용할 데이터 세팅*  
plt.grid() *# 그리드 그리기*  
plt.xlabel('time') *#x축 라벨 붙이기*  
plt.ylabel('Amplitude') *#y축 라벨 붙이기*  
plt.title('Example of sinewave') *# 그래프 제목 붙이기*  
plt.show()

두개 이상의 그래프를 하나의 화면에 그려낼 수도 있습니다. plot만 더 추가 해 주면 됩니다.

In [41]:

plt.figure(figsize=(10, 6))  
plt.plot(t, np.sin(t))  
plt.plot(t, np.cos(t)) *# cos 그래프 추가*  
plt.grid()  
plt.xlabel('time')  
plt.ylabel('Amplitude')  
plt.title('Example of sinewave')  
plt.show()

범례 옵션도 plot 함수에 추가 해주면 범례 표현도 가능 합니다.

In [46]:

plt.figure(figsize=(10, 6))  
plt.plot(t, np.sin(t), label='sin') *# sin 범례 추가*  
plt.plot(t, np.cos(t), label='cos') *# cos 범례 추가*  
plt.grid()  
plt.legend() *# 범례 보여주기*  
plt.xlabel('time')  
plt.ylabel('Amplitude')  
plt.title('Example of sinewave')  
plt.show()

또한 lw 옵션을 이용해 선의 굵기를 지정 할 수 있고, color 옵션으로 색상도 지정 가능합니다

In [45]:

plt.figure(figsize=(10, 6))  
plt.plot(t, np.sin(t), lw=3 ,label='sin') *# 선굵기 설정*  
plt.plot(t, np.cos(t), color='g', label='cos') *# 색상 설정('g' : 초록색)*  
plt.grid()  
plt.legend() *# 범례 보여주기*  
plt.xlabel('time')  
plt.ylabel('Amplitude')  
plt.title('Example of sinewave')  
plt.show()

다른 모양 그래프로 더 알아 보겠습니다. linstyle 옵션으로 선의 모양도 지정 할 수 있습니다.

In [51]:

t = [0,1,2,3,4,5,6]  
y = [1,4,5,8,9,5,3]  
plt.figure(figsize=(10, 6))  
plt.plot(t,y,color='green', linestyle='dashed') *# dotted : 점, dahsed : 끊어진 선*  
plt.show()

데이터가 실제 존재 하는 곳에 점을 찍어 줄 수도 있습니다.

**marker** 옵션을 이용해 점을 찍고, **markerfacecolor** 옵션을 이용해 색상을, **markersize** 옵션으로 크기를 지정 할 수 있습니다.

In [56]:

t = [0,1,2,3,4,5,6]  
y = [1,4,5,8,9,5,3]  
plt.figure(figsize=(10, 6))  
plt.plot(t,y,color='green', linestyle='dashed', marker='o', markerfacecolor='blue', markersize=12) *#마커 꾸며주기 추가*  
plt.show()

**scatter**를 사용하면 점 그래프가 됩니다

In [61]:

t = np.array([0,1,2,3,4,5,6,7,8,9])  
y = np.array([9,8,7,9,8,3,2,4,3,4])  
plt.figure(figsize=(10, 6))  
plt.scatter(t,y, marker='>') *#scatter로 점 그래프 그리기*  
plt.show()

**scatter**에서는 마커의 크기를 **s 옵션**으로 조절합니다.

**c 옵션(색상)**으로 color map까지 만들어 낼 수 있습니다.

In [64]:

colormap = y *# y 값에 따라서 colormap 지정해주기*  
  
plt.figure(figsize=(10, 6))  
plt.scatter(t,y,s=50, c = colormap, marker='>')  
plt.colorbar() *#색상 바 보여주기*  
plt.show()

numpy의 랜덤함수를 이용해서 그래프를 그려 보겠습니다.

loc 옵션으로 평균을, scale 옵션으로 표준 편차를 조절 할 수 있습니다.

In [65]:

s1 = np.random.normal(loc=0, scale=1, size=1000)  
s2 = np.random.normal(loc=5, scale=0.5, size=1000)  
s3 = np.random.normal(loc=10, scale=2, size=1000)  
  
plt.figure(figsize=(10, 6))  
plt.plot(s1, label='s1')  
plt.plot(s2, label='s2')  
plt.plot(s3, label='s3')  
plt.legend()  
plt.show()

위의 그래프를 boxplot으로 표현할 수도 있습니다. boxplot은 나중에 더 자세히 살펴볼 수 있습니다.

In [66]:

plt.figure(figsize=(10, 6))  
plt.boxplot((s1,s2,s3))  
plt.grid()  
plt.show()

In [ ]: