matplotlib

- matplotlib.pyplot
  - 각종 그래프를 그리게 도와주는 모듈
  - MATLAB의 그림과 비슷하게 그릴 수 있도록 해 주는 모듈
  - 2D 그래픽용으로 가장 많이 쓰이는 모듈
- http://matplotlib.org/ (http://matplotlib.org/)

In [3]:

```
import matplotlib.pyplot as plt
```

- 위의 형태로 import하여 사용

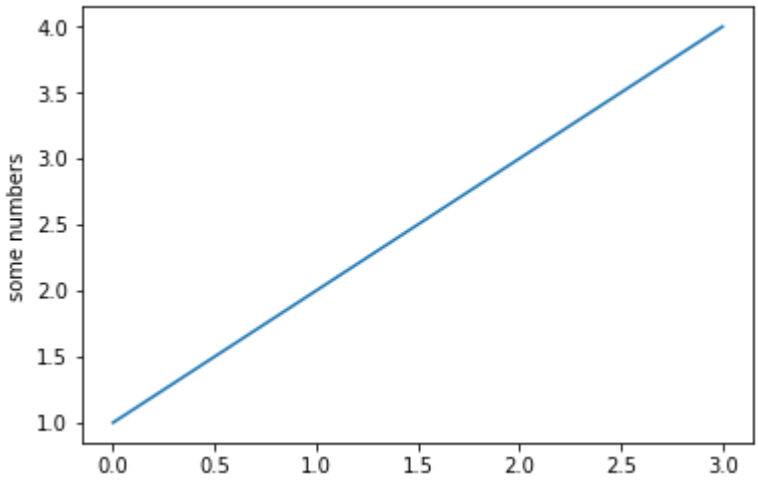
간단한 예제

In [4]:

```
%matplotlib inline    # Notebook에서 그림을 보이기 위해 필요
```

In [5]:

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4])
plt.ylabel('some numbers')
plt.show()
```

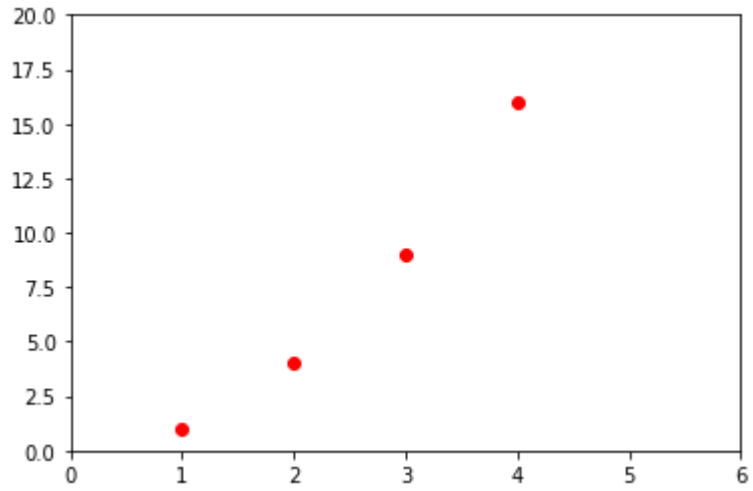


- 위의 예제처럼 plot() 함수의 인자에 하나의 리스트만 입력하면 이를 y값으로 인식
  - 즉 y값은 현재 [1,2,3,4]
  - x값은 default로 [0,1,2,3]이 됨
  - 기본적으로 직선 그래프를 생성
  - y축에 대한 설명은 plt.ylabel() 함수를 이용

간단한 예제(2)

In [6]:

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4], [1,4,9,16], 'ro')
plt.axis([0, 6, 0, 20])
plt.show()
```

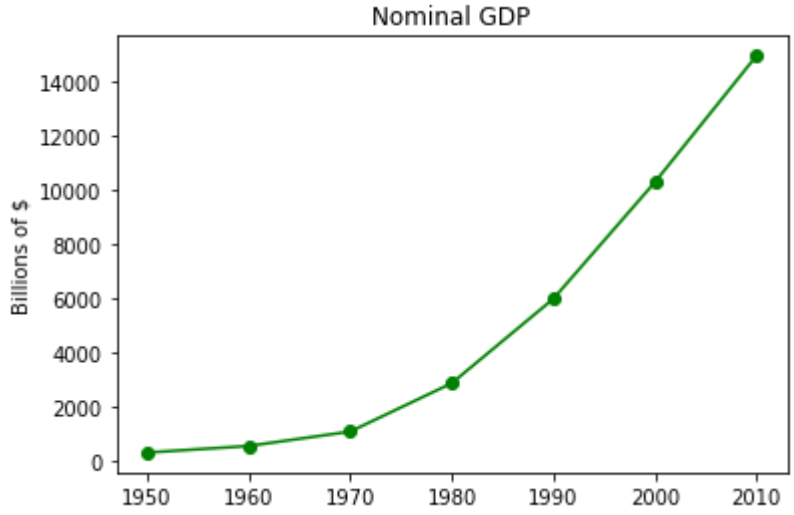


- plot()에 x값과 y값을 둘 다 입력한 형태를 볼 수 있다.
  - x 값 : [1, 2, 3, 4]
  - y 값 : [1, 4, 9 ,16]
  - 'ro'는 빨간색(red) 동그라미(o)를 의미
  - https://matplotlib.org/api/colors_api.html (https://matplotlib.org/api/colors_api.html)
  - https://matplotlib.org/api/markers_api.html (https://matplotlib.org/api/markers_api.html)
- axis() 함수의 인자 : [xmin, xmax, ymin, ymax]

선 그래프

In [7]:

```
from matplotlib import pyplot as plt
years = [1950, 1960, 1970, 1980, 1990, 2000, 2010]
gdp = [300, 543, 1075, 2862, 5979, 10289, 14958]

plt.plot(years, gdp, color='green', marker='o', linestyle='solid')

plt.title("Nominal GDP")

plt.ylabel("Billions of $")
plt.show()
```

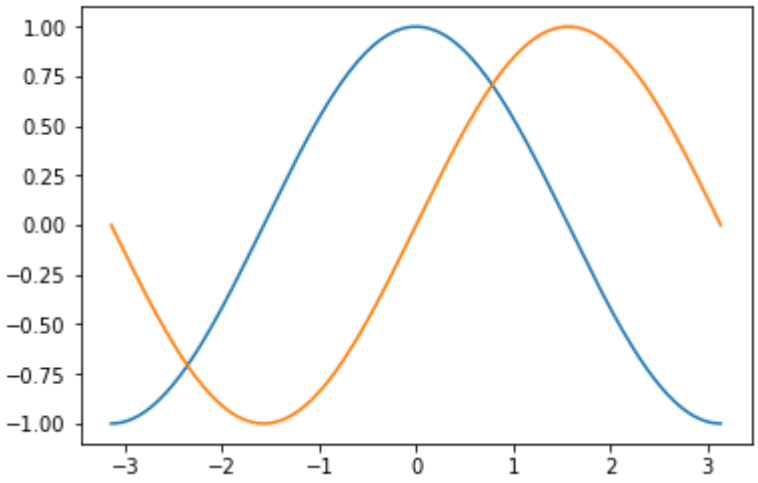


- 그래프의 color, marker, linestyle를 인자로 직접 명시
  - https://matplotlib.org/api/lines_api.html (https://matplotlib.org/api/lines_api.html)

기본 셋팅에서 그림 그리기

In [8]:

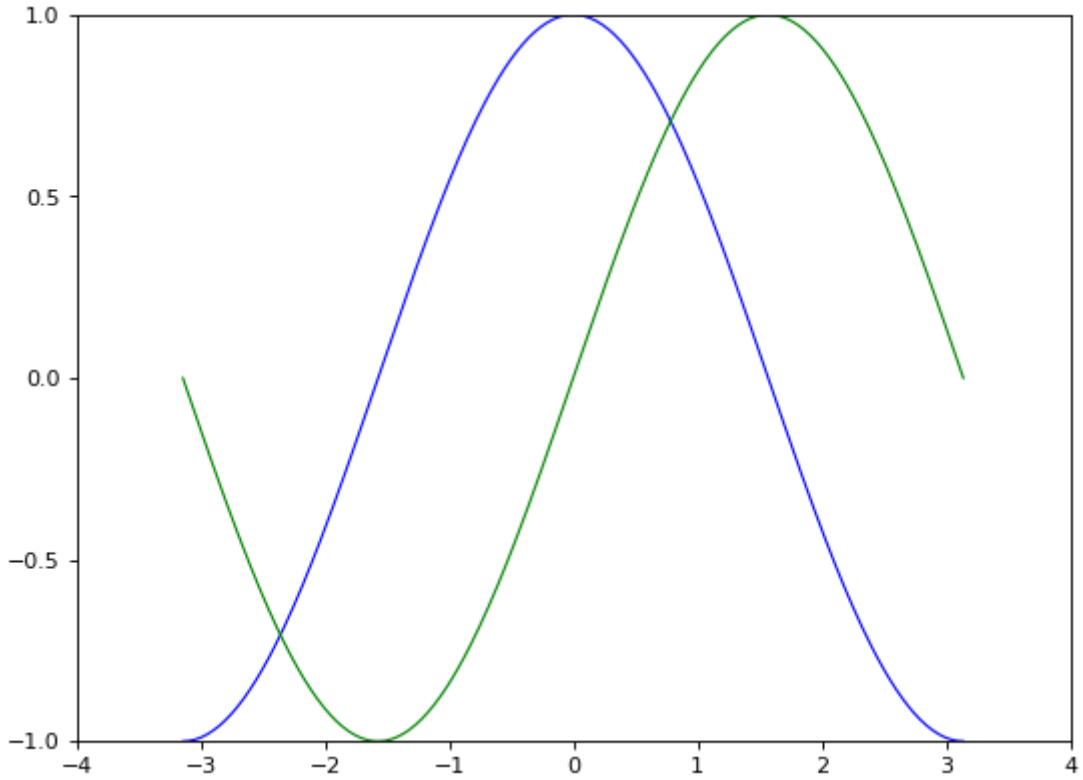
```
import numpy as np
import matplotlib.pyplot as plt

X = np.linspace(-np.pi, np.pi, 256, endpoint=True)
C, S = np.cos(X), np.sin(X)

plt.plot(X, C)
plt.plot(X, S)

plt.show()
```



여러 셋팅 자세히 살펴 보기

In [9]:

```
plt.figure(figsize=(8, 6), dpi=80)        # 8x6 inches 크기의 그림 생성, 80 dots per inch
plt.subplot(1, 1, 1)                      # 1x1 subplot 생성

X = np.linspace(-np.pi, np.pi, 256, endpoint=True)
C, S = np.cos(X), np.sin(X)

plt.plot(X, C, color="blue", linewidth=1.0, linestyle="-")       # 라인두께 1의 파란색 cos 그리기
plt.plot(X, S, color="green", linewidth=1.0, linestyle="-")      # 라인두께 1의 초록색 sin 그리기

plt.xlim(-4.0, 4.0)                                              # Set x limits
plt.ylim(-1.0, 1.0)                                              # Set y limits

plt.xticks(np.linspace(-4, 4, 9, endpoint=True))                 # Set x ticks
plt.yticks(np.linspace(-1, 1, 5, endpoint=True))                 # Set y ticks
plt.show()                                                       # Show result on screen
```

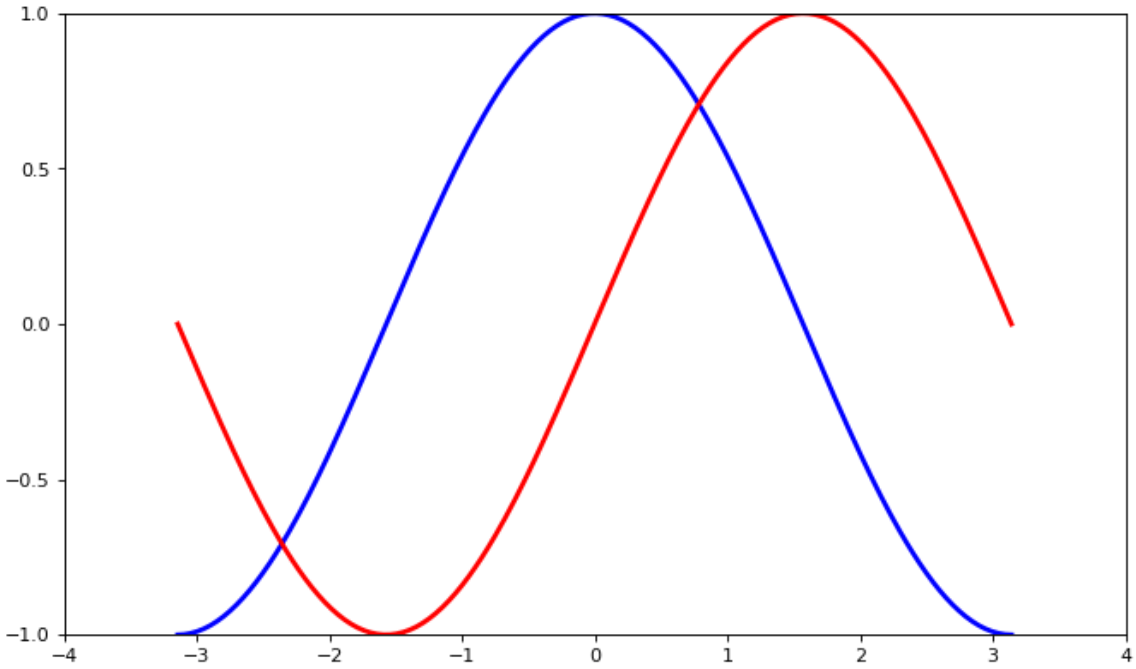


셋팅 바꾸어 보기

- 선의 색깔과 두께를 바꾸려면 다음을 수정

In [10]:

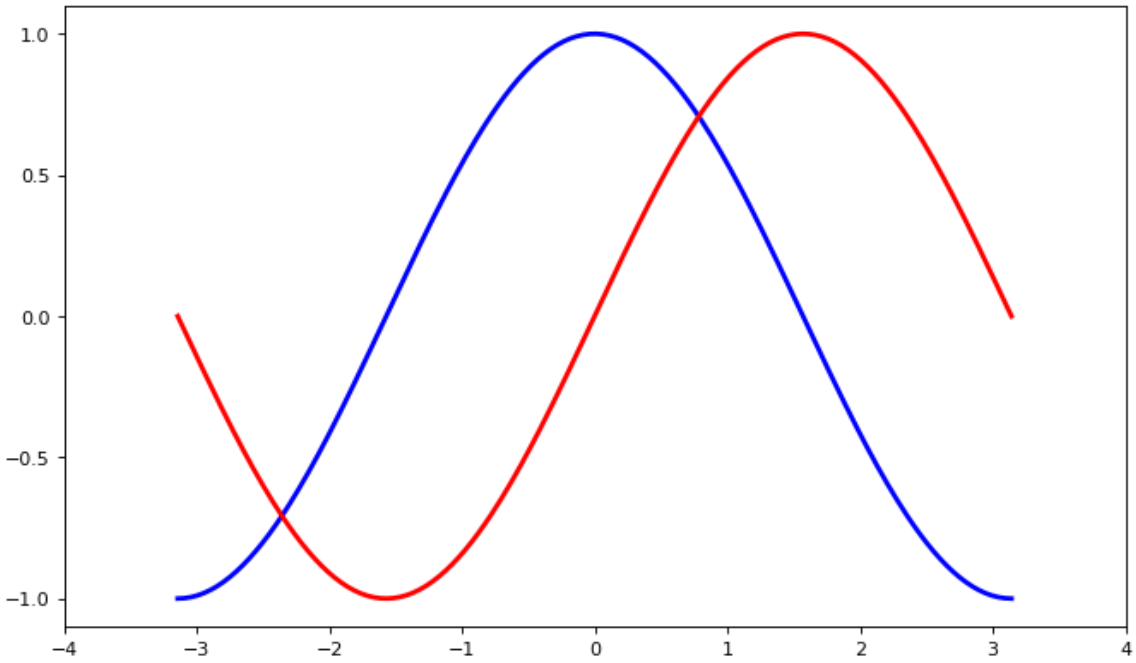
```
plt.figure(figsize=(10, 6), dpi=80)

#### 수정한 부분 ##########################################
plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-")
plt.plot(X, S, color="red",  linewidth=2.5, linestyle="-")
##########################################################

plt.xlim(-4.0, 4.0)
plt.ylim(-1.0, 1.0)

plt.xticks(np.linspace(-4, 4, 9, endpoint=True))
plt.yticks(np.linspace(-1, 1, 5, endpoint=True))
plt.show()
```



- x, y의 한계값 바꾸려면 다음을 수정

In [11]:

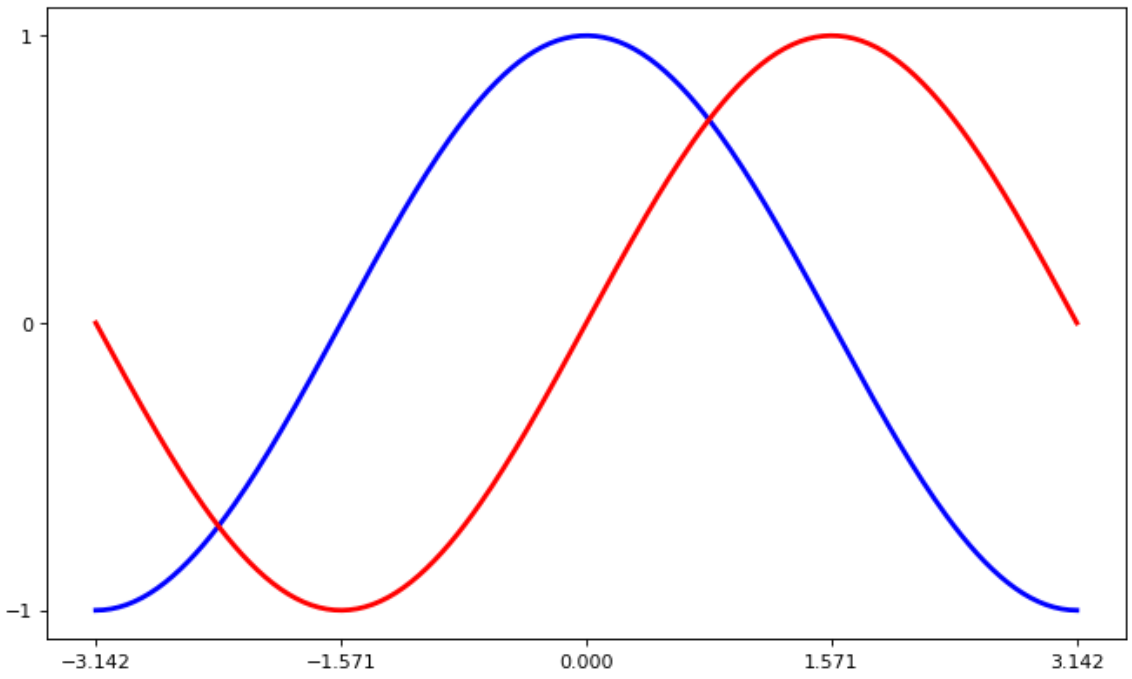
```
plt.figure(figsize=(10, 6), dpi=80)

plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-")
plt.plot(X, S, color="red",  linewidth=2.5, linestyle="-")

#### 수정한 부분 ############################################
plt.xlim(X.min() * 1.1, X.max() * 1.1)
plt.ylim(C.min() * 1.1, C.max() * 1.1)
############################################################

plt.xticks(np.linspace(-4, 4, 9, endpoint=True))
plt.yticks(np.linspace(-1, 1, 5, endpoint=True))
plt.show()
```



- tick의 위치를 바꾸려면

In [12]:

```
plt.figure(figsize=(10, 6), dpi=80)

plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-")
plt.plot(X, S, color="red",  linewidth=2.5, linestyle="-")

plt.xlim(X.min() * 1.1, X.max() * 1.1)
plt.ylim(C.min() * 1.1, C.max() * 1.1)

#### 수정한 부분 ########################################
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
plt.yticks([-1, 0, +1])
#########################################################

plt.show()
```

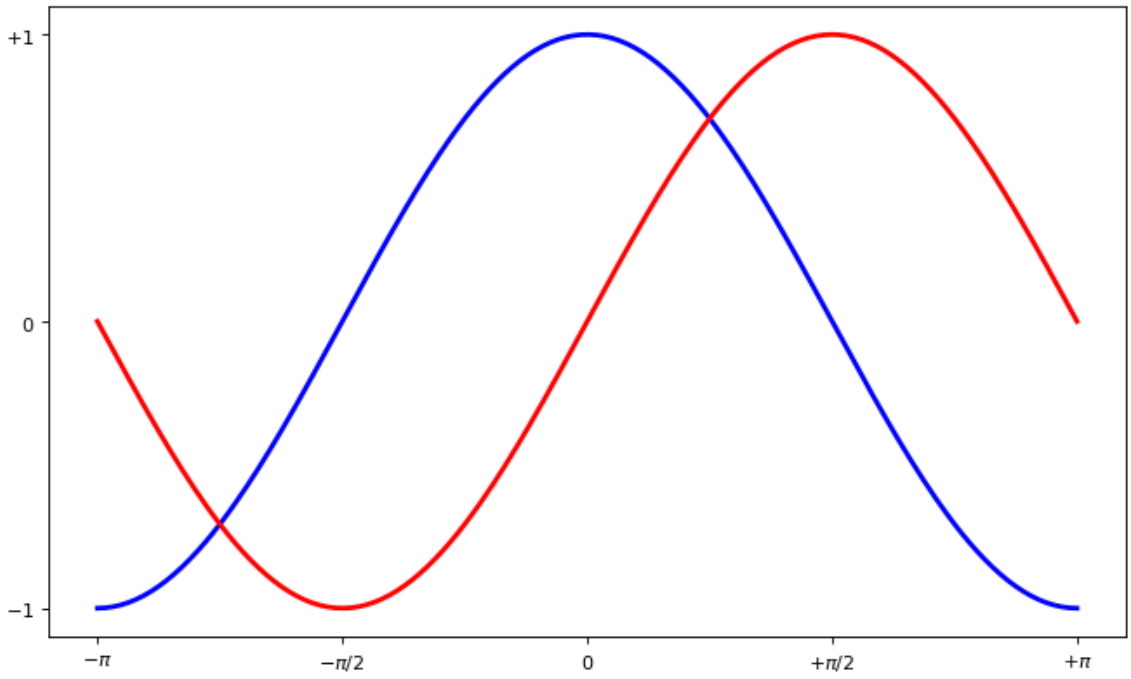


셋팅 바꾸어 보기(2)

- tick에 label 넣기

In [13]:

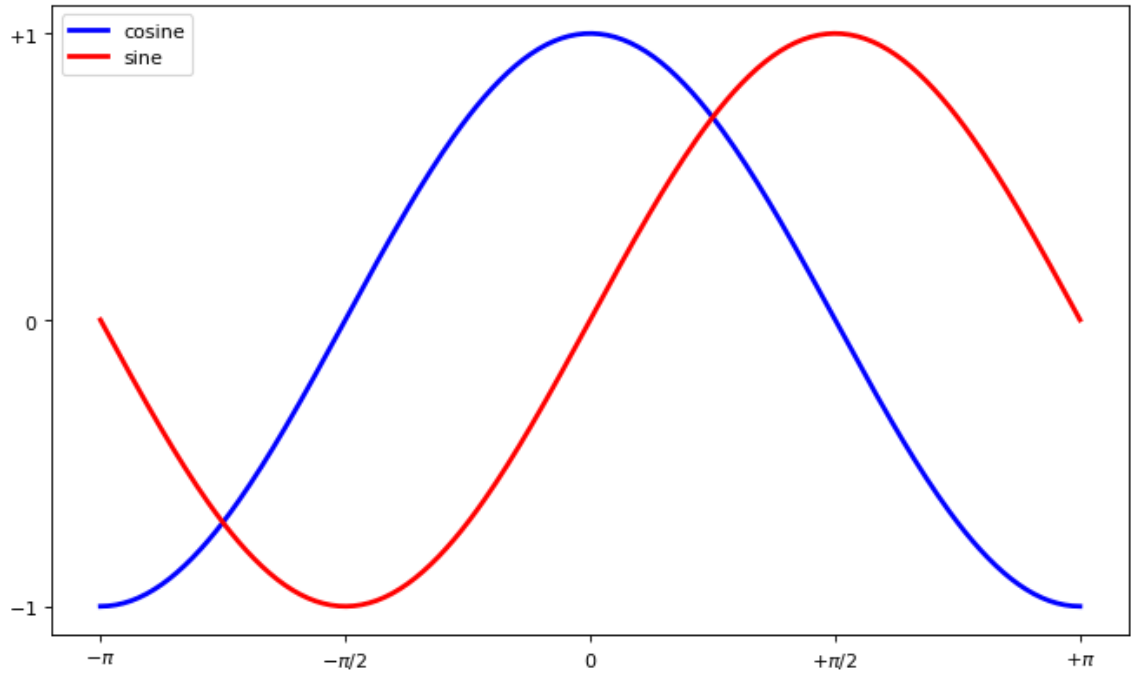
```
plt.figure(figsize=(10, 6), dpi=80)

plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-")
plt.plot(X, S, color="red",  linewidth=2.5, linestyle="-")

plt.xlim(X.min() * 1.1, X.max() * 1.1)
plt.ylim(C.min() * 1.1, C.max() * 1.1)

plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
plt.yticks([-1, 0, +1])

#### 수정한 부분 ##########################################################################################
###########
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi], [r'$-₩pi$', r'$-₩pi/2$', r'$0$', r'$+₩pi/2$',
r'$+₩pi$'])
plt.yticks([-1, 0, +1], [r'$-1$', r'$0$', r'$+1$'])
#########################################################################################################
###########

plt.show()
```



- legend 넣기

In [14]:

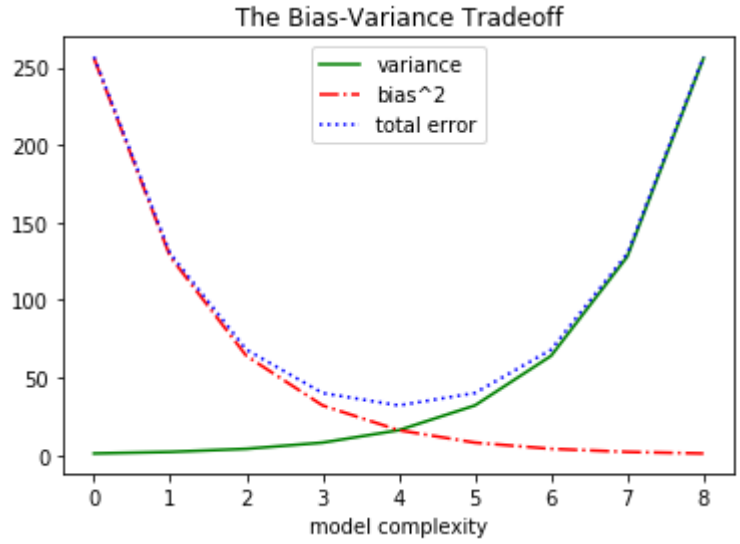
```
plt.figure(figsize=(10, 6), dpi=80)

#### 수정한 부분 ##################################################################
plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-", label="cosine")
plt.plot(X, S, color="red",  linewidth=2.5, linestyle="-", label="sine")
###################################################################################

plt.xlim(X.min() * 1.1, X.max() * 1.1)
plt.ylim(C.min() * 1.1, C.max() * 1.1)

plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
plt.yticks([-1, 0, +1])

plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi], [r'$-₩pi$', r'$-₩pi/2$', r'$0$', r'$+₩pi/2$',
r'$+₩pi$'])
plt.yticks([-1, 0, +1], [r'$-1$', r'$0$', r'$+1$'])

#### 추가한 부분 ##################################################################
plt.legend(loc='upper left')
###################################################################################

plt.show()
```



선 그래프 – plot()

In [15]:

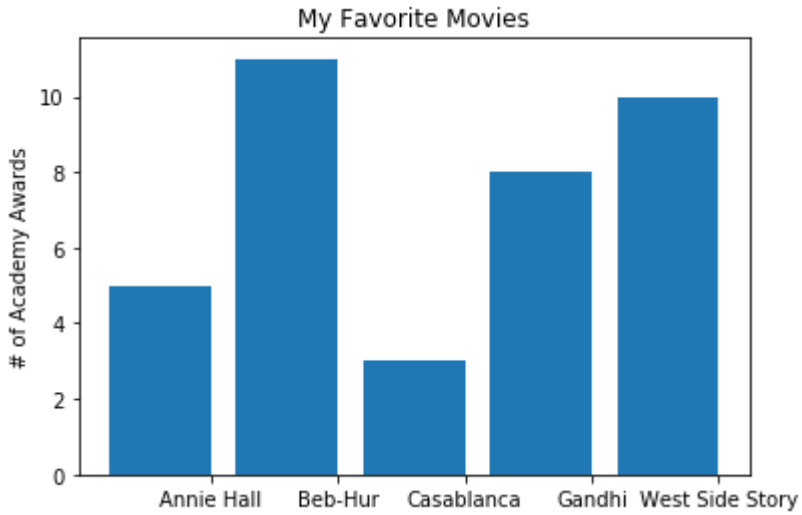
```
variance = [1, 2, 4, 8, 16, 32, 64, 128, 256]
bias_squared = [256, 128, 64, 32, 16, 8, 4, 2, 1]
total_error = [x+y for x,y in zip(variance, bias_squared)]
xs = [i for i,_ in enumerate(variance)]

plt.plot(xs, variance, 'g-', label='variance')
plt.plot(xs, bias_squared, 'r-.', label='bias^2')
plt.plot(xs, total_error, 'b:', label='total error')

plt.legend(loc=9)  #loc=9 : top center
plt.xlabel("model complexity")
plt.title("The Bias-Variance Tradeoff")
plt.show()
```



바 그래프 – bar()

In [16]:

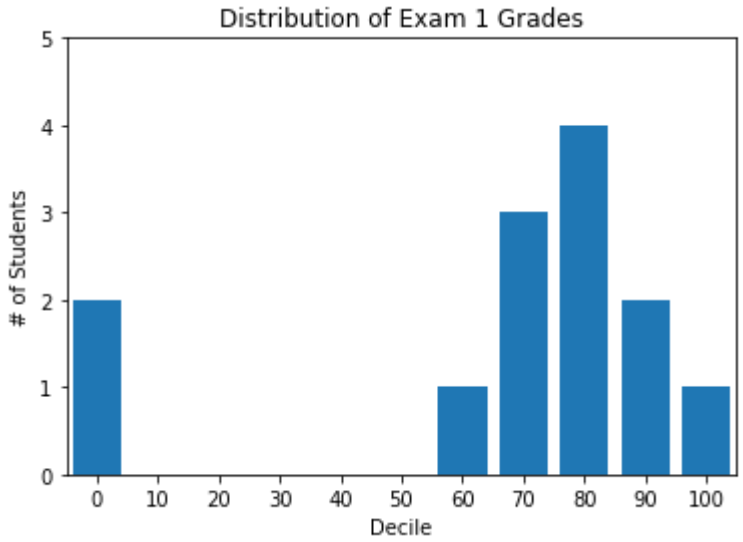
```
movies = ["Annie Hall", "Beb-Hur", "Casablanca", "Gandhi", "West Side Story"]
num_oscars = [5, 11, 3, 8, 10]
xs = [i+0.1 for i, _ in enumerate(movies)]
plt.bar(xs, num_oscars)
plt.ylabel("# of Academy Awards")
plt.title("My Favorite Movies")
plt.xticks([i+0.5 for i, _ in enumerate(movies)], movies)
plt.show()
```



바 그래프 (2)

In [17]:

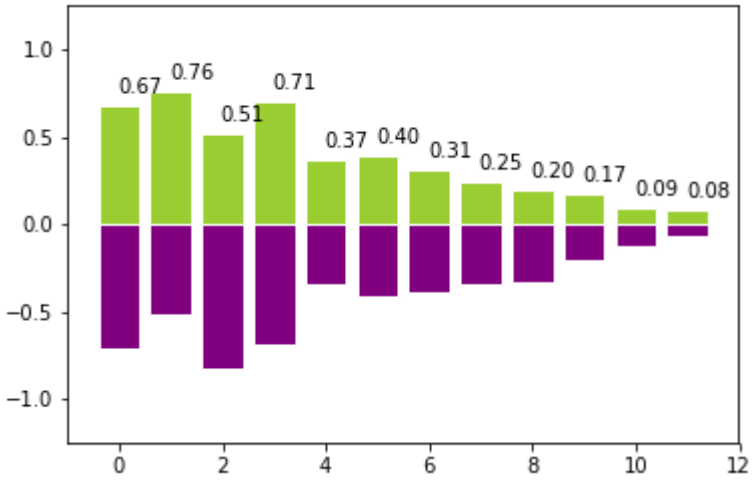
```
from collections import Counter
from matplotlib import pyplot as plt

grades = [83, 95, 91, 87, 70, 0, 85, 82, 100, 67, 73, 77, 0]
def decile(grade) : return (grade // 10) * 10
histogram = Counter(decile(grade) for grade in grades)

plt.bar(histogram.keys(), histogram.values(), 8)  # 바 너비 8
plt.axis([-5, 105, 0, 5])     # x-축 from -5 to 105,   y-축 from 0 to 5
plt.xticks([10*i for i in range(11)])    # x-축의 값들이 0, 10, …, 100에서 표현

plt.xlabel("Decile")
plt.ylabel("# of Students")
plt.title("Distribution of Exam 1 Grades")
plt.show()
```



바 그래프 (3)

In [18]:

```
n = 12
X = np.arange(n)
Y1 = (1 - X / float(n)) * np.random.uniform(0.5, 1.0, n)
Y2 = (1 - X / float(n)) * np.random.uniform(0.5, 1.0, n)

plt.bar(X, +Y1,  facecolor='yellowgreen',  edgecolor='white')
plt.bar(X, -Y2,  facecolor='purple',  edgecolor='white')

for x, y in zip(X, Y1):
    plt.text(x + 0.4, y + 0.05, '%.2f' % y, ha='center', va='bottom')

plt.ylim(-1.25, +1.25)
plt.show()
```



산점도 – scatter()

In [19]:

```
friends = [70, 65, 72, 63, 71, 64, 60, 64, 67]
minutes = [175, 170, 205, 120, 220, 130, 105, 145, 190]
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']
plt.scatter(friends, minutes)
for label, friend_count, minute_count in zip(labels, friends, minutes):
    plt.annotate(label,
                 xy=(friend_count, minute_count),
                 xytext=(5,-5),
                 textcoords='offset points')

plt.title("Daily Minutes vs. Number of Friends")
plt.xlabel("# of Friends")
plt.ylabel("daily minutes spent on the site")
plt.show()
```