

파이썬 라이브러리를 활용한 데이터 분석

9장 그래프와 시각화

2020.07.03 2h

9장 그래프와 시각화

Matplotlib

Matplotlib 개요

- 2D 그래프를 위한 데스크탑 패키지

- 파이썬에서 자료를 차트(chart)나 플롯(plot)으로 시각화(visualaization)하는 패키지
- 정형화된 차트나 플롯 이외에도 저수준 API를 사용한 다양한 시각화 기능을 제공
 - 라인 플롯(line plot)
 - 스캐터 플롯(scatter plot)
 - 컨투어 플롯(contour plot)
 - 서피스 플롯(surface plot)
 - 바 차트(bar chart)
 - 히스토그램(histogram)
 - 박스 플롯(box plot)
- 2002년 존 헌터가 시작
- Matplotlib 갤러리 웹사이트
 - <http://matplotlib.org/gallery.html>

주피터 노트북 매직 명령어

- **%matplotlib inline**
 - 그림을 셀 아래에 결과로 삽입
- **%matplotlib notebook**
 - 대화형 시각화 도구

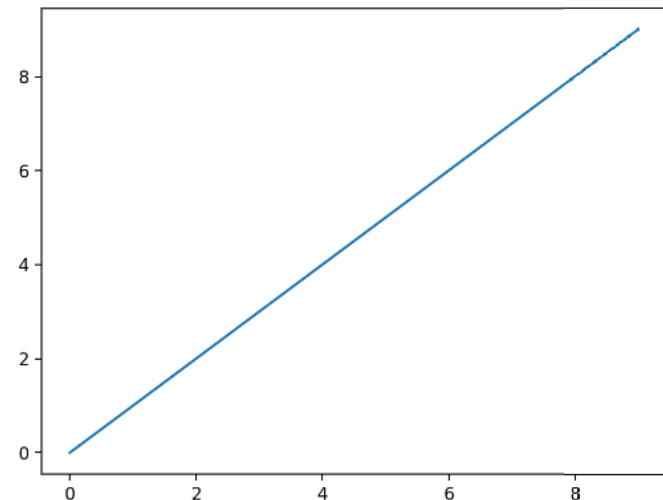
In [7]: `%matplotlib notebook`

A Brief matplotlib API Primer

In [2]: `import matplotlib.pyplot as plt`

In [8]: `import numpy as np
data = np.arange(10)
data
plt.plot(data)`

Figure 1



x=0.711141 y=4.76086

Out [8]: `[<matplotlib.lines.Line2D at 0x25e9bc1c5c8>]`

API

- **plot()**
 - 가장 간단한 플롯은 선을 그리는 라인 플롯(line plot)
 - 라인 플롯은 데이터가 시간, 순서 등에 따라 어떻게 변화하는지 보여주기 위해 사용
 - http://matplotlib.org/api/pyplot_api.html#Matplotlib.pyplot.plot



[home](#) | [examples](#) | [gallery](#) | [pyplot](#) | [docs](#) » The Matplotlib API »

pyplot

matplotlib.pyplot

Provides a MATLAB-like plotting framework.

`pylab` combines `pyplot` with `numpy` into a single namespace. This is convenient for interactive work, but for programming it is recommended that the namespaces be kept separate, e.g.:

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

x = np.arange(0, 5, 0.1);
y = np.sin(x)
plt.plot(x, y)
```

```
matplotlib.pyplot.acorr(x, hold=None, data=None, **kwargs)
```

한글 지원

- 지원 폰트 확인 후
 - 지정

```
In [10]: import matplotlib as mpl
         sorted([f.name for f in mpl.font_manager.fontManager.ttflist])
```

```
'Arial',
'Arial',
'Arial',
'Arial',
'Arial',
'Arial',
'Arial',
'Arial',
'Arial',
'Arial Rounded MT Bold',
'Arvo',
'Arvo',
'Arvo',
'Arvo',
'Arvo',
'Bahnschrift',
'Baskerville Old Face',
'Batang',
'Bauhaus 93',
'Bell MT',
'Bell MT',
'Bell MT',
```

```
In [11]: import matplotlib as mpl

         # 폰트 설정
         #mpl.rc('font', family='NanumGothic')
         mpl.rc('font', family='Malgun Gothic')
         # 유니코드에서 음수 부호설정
         mpl.rc('axes', unicode_minus=False)
```

Figure와 subplot

• Figure 개요

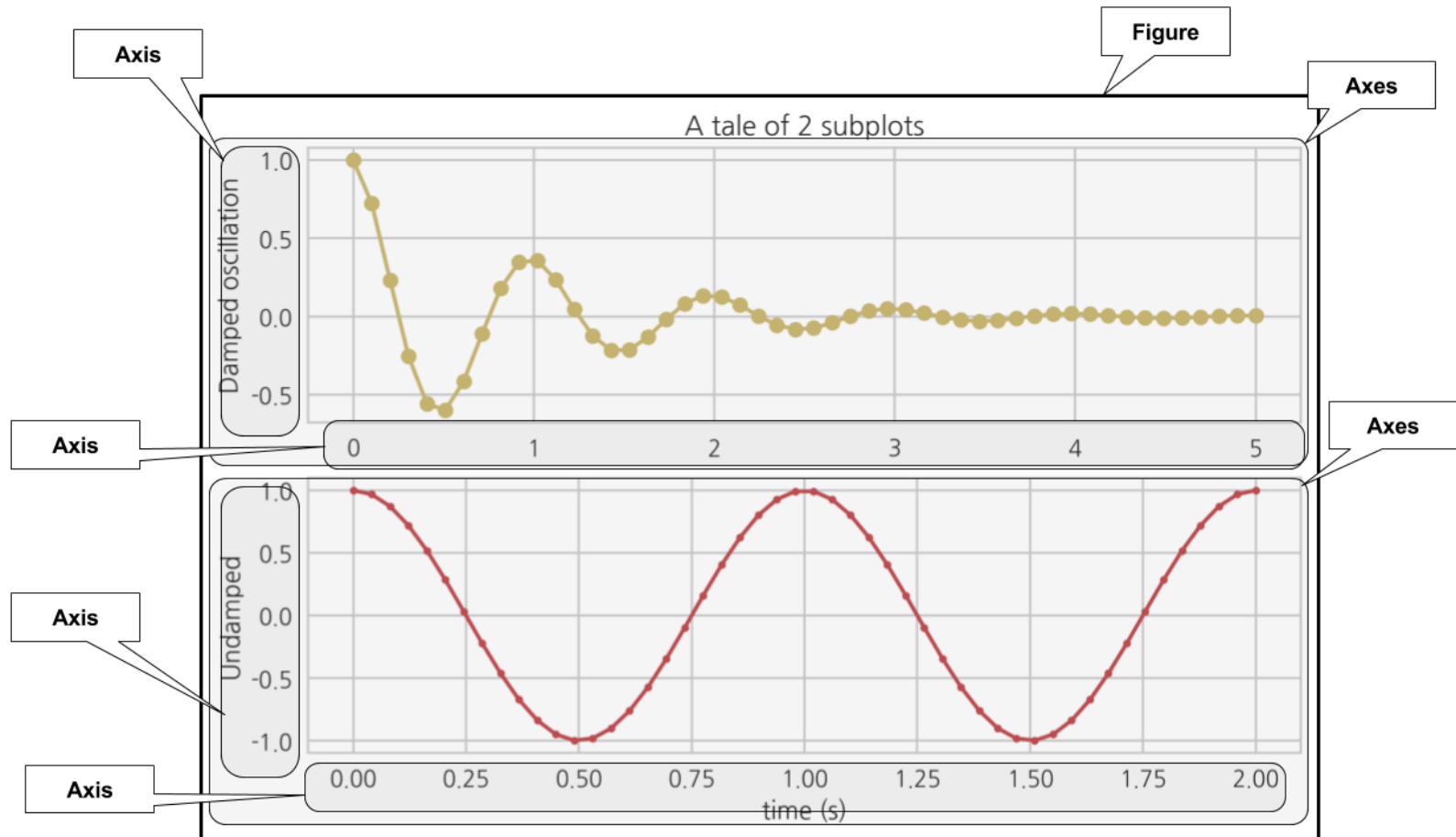
- 그려지는 그래프가 figure 객체 내에 존재, 그림을 그리는 종이라고 이해
- `figsize=(10, 6)`
 - 그림 크기: `figsize` 인수로 설정(크기와 비율을 지정)
- `fig = plt.figure()`
 - Figure 생성
 - `figure` 명령을 사용하여 그 반환 값으로 Figure 객체를 얻음
 - 일반적인 `plot` 명령 등을 실행하면 자동으로 Figure를 생성해 주기 때문에 일반적으로는 `figure` 명령을 잘 사용하지 않음
 - `figure` 명령을 명시적으로 사용하는 경우
 - 여러 개의 윈도우를 동시에 띄워야 하거나(line plot이 아닌 경우)
 - Jupyter 노트북 등에서(line plot의 경우) 그림의 크기를 설정할 경우

• `show()`

- 시각화 명령을 실제로 차트로 렌더링(rendering)하고 마우스 움직임 등의 이벤트를 기다리라는 지시
- 주피터 노트북에서는 셀 단위로 플롯 명령을 자동 렌더링 해주므로 `show` 명령이 필요 없음
- 일반 파이썬 인터프리터로 가동되는 경우를 대비하여 항상 마지막에 실행

그림 구조

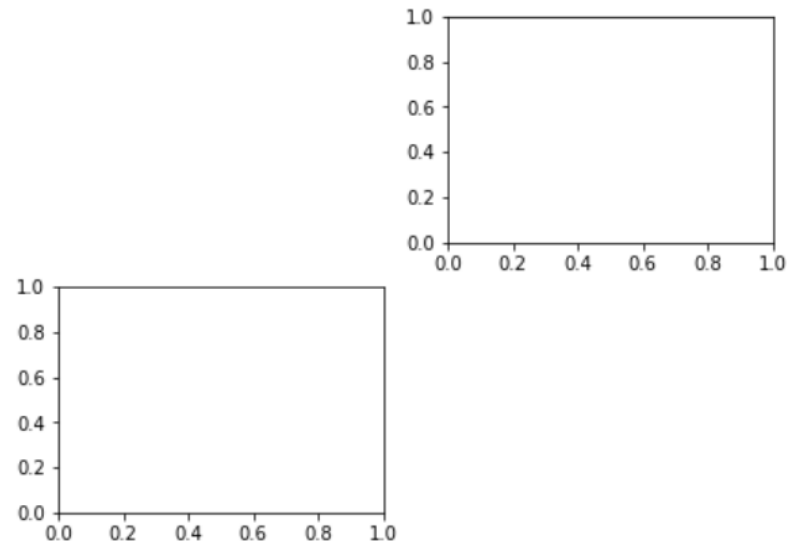
- Figure는 그림이 그려지는 캔버스나 종이
 - Axes는 하나의 플롯, 그리고 Axis는 가로축이나 세로축 등의 축



Subplot 개요

- **그리드(grid) 형태의 Axes 객체들을 생성**
 - Figure 안에 있는 각각의 플롯은 Axes 라고 불리는 객체
 - Figure 내부에 행렬(matrix) 형태의 여러 그림이 있으며 이를 Axes라 함
- **Axes: 내부의 subplot**
 - `ax1 = fig.add_subplot(2, 2, 1)`
 - 크기 2 x 2
 - 4개의 그림 중 1번 그림
 - `ax2 = fig.add_subplot(2, 2, 2)`
 - `ax3 = fig.add_subplot(2, 2, 3)`

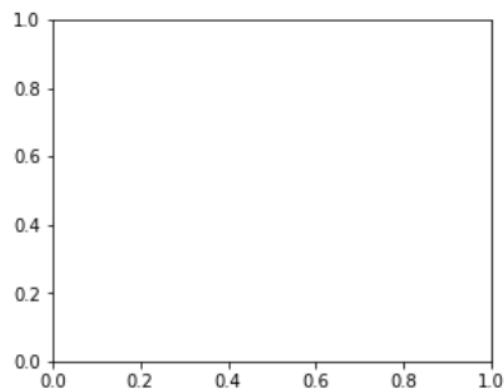
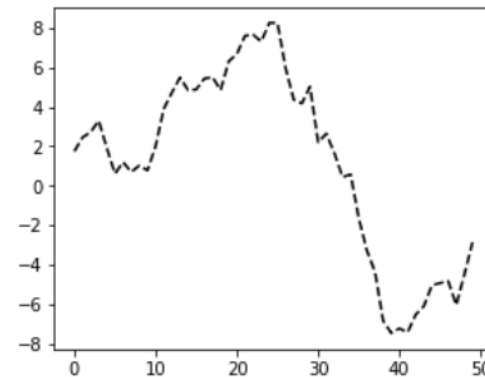
```
In [15]: fig = plt.figure(figsize=(7, 5))
          #ax1 = fig.add_subplot(2, 2, 1)
          ax2 = fig.add_subplot(2, 2, 2)
          ax3 = fig.add_subplot(2, 2, 3)
```



여러 그림 그리기

- 가장 최근의 figure와 그 서브플롯에 그림

```
In [19]: fig = plt.figure(figsize=(10, 8))  
#ax1 = fig.add_subplot(2, 2, 1)  
ax2 = fig.add_subplot(2, 2, 2)  
plt.plot(np.random.randn(50).cumsum(), 'k--')  
ax3 = fig.add_subplot(2, 2, 3)
```



옵션 'k--'와 서브플롯 객체

- **black, 점선**
 - 색상
 - 선 스타일
 - 실선(solid), 대시선(dashed)
 - 점선(dotted), 대시-점선(dash-dot)
 - 마커(marker)
 - 데이터 위치를 나타내는 기호
- **문자열 형태**
 - `fmt = '[marker][line][color]'` 또는 `[color][marker][line]`
 - 파악만 되면 순서는 상관 없음
 - 'b'
 - # blue markers with default shape
 - 'or'
 - # red circles
 - '-g'
 - # green solid line
 - '...'
 - # dashed line with default color
 - '^k:'
 - # black triangle_up markers connected by a dotted line

문자열	약자
blue	b
green	g
red	r
cyan	c
magenta	m
yellow	y
black	k
white	w

선 스타일 문자열	의미
-	solid line style
--	dashed line style
-. .	dash-dot line style
:	dotted line style

마커의 다양한 종류

• 마커

- 특정 지점의 실제 데이터를 돋보이게 그리기



마커 문자열

의미

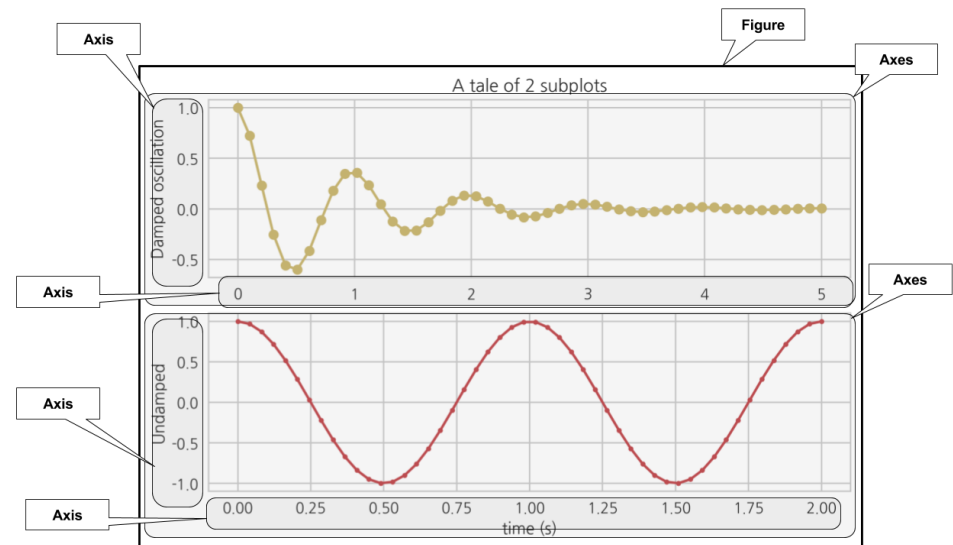
RAMMING

.	point marker
,	pixel marker
o	circle marker
v	triangle_down marker
^	triangle_up marker
<	triangle_left marker
>	triangle_right marker
1	tri_down marker
2	tri_up marker
3	tri_left marker
4	tri_right marker
s	square marker
p	pentagon marker
*	star marker
h	hexagon1 marker
H	hexagon2 marker
+	plus marker
x	x marker
D	diamond marker
d	thin_diamond marker

thon

plt.subplot

- **그리드(grid) 형태의 Axes 객체들을 생성**
 - Figure 안에 있는 각각의 플롯은 Axes 라고 불리는 객체
 - Figure 내부에 행렬(matrix) 형태의 여러 그림이 있으며 이를 Axes라 함
- **`ax1 = plt.subplot(2, 1, 1)`**
 - `ax1 = plt.subplot('211')` 로도 가능
 - `subplot(m, n, number)`
 - 세 개의 인수
 - **m, n**
 - 전체 그리드 행렬의 모양을 지시하는 두 숫자
 - **number**
 - 인수가 네 개 중 어느 것인지를 의미하는 숫자
 - 첫번째 플롯을 가리키는 숫자가 0이 아니라 1임에 주의



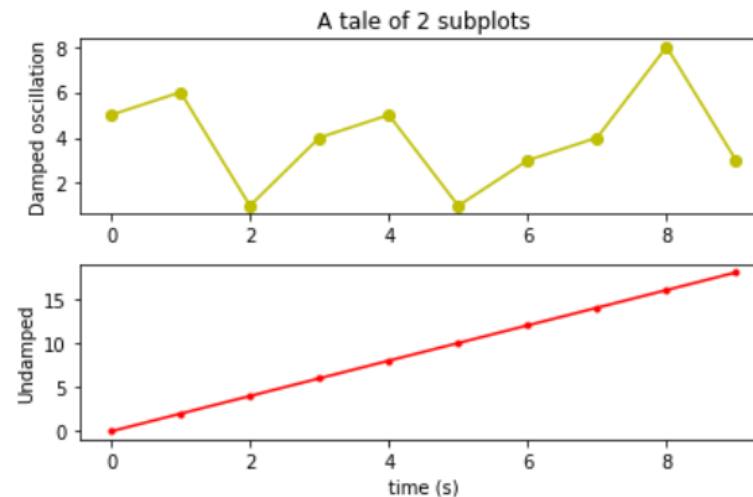
서브플롯 예

```
In [45]: ax1 = plt.subplot(2, 1, 1)
plt.plot(np.random.randint(1, 10, 10), 'yo-')
plt.title('A tale of 2 subplots')
plt.ylabel('Damped oscillation')
print(ax1)

#ax2 = plt.subplot(2, 1, 2)
ax2 = plt.subplot(2, 1, 2)
plt.plot(np.arange(10), np.arange(10)*2, 'r.-')
plt.xlabel('time (s)')
plt.ylabel('Undamped')
print(ax2)

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

AxesSubplot(0.125,0.536818;0.775x0.343182)
 AxesSubplot(0.125,0.125;0.775x0.343182)



plt.subplots(m, n)

• 메소드

- matplotlib.pyplot.subplots(nrows=1, ncols=1, sharex=False, sharey=False, squeeze=True, subplot_kw=None, gridspec_kw=None, **fig_kw)
- 특정한 배치에 맞추어 여러 개의 서브플롯을 포함하는 figure를 생성
 - 인자 **sharex=True, sharey=True**
 - 각 축의 인자를 하나로 공유

Table 9-1. *pyplot.subplots* options

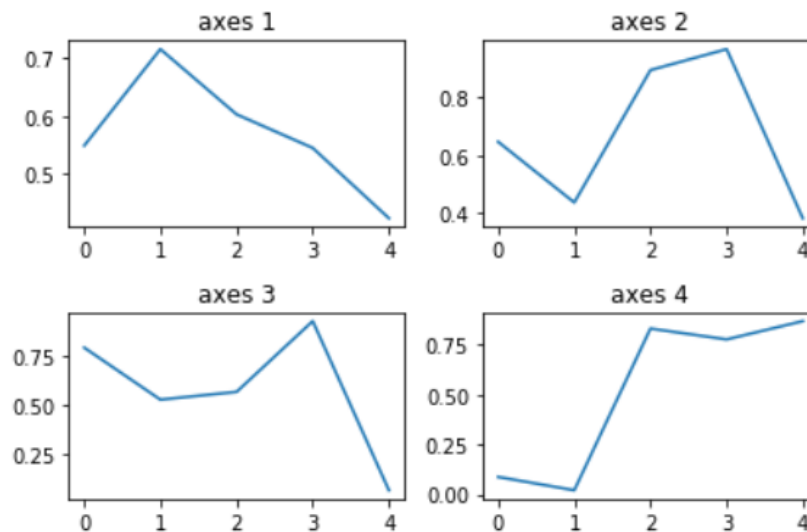
Argument	Description
nrows	Number of rows of subplots
ncols	Number of columns of subplots
sharex	All subplots should use the same x-axis ticks (adjusting the <code>xlim</code> will affect all subplots)
sharey	All subplots should use the same y-axis ticks (adjusting the <code>ylim</code> will affect all subplots)
subplot_kw	Dict of keywords passed to <code>add_subplot</code> call used to create each subplot
**fig_kw	Additional keywords to <code>subplots</code> are used when creating the figure, such as <code>plt.subplots(2, 2, figsize=(8, 6))</code>

서브플롯 예

```
In [47]: fig, axes = plt.subplots(2, 2)

np.random.seed(0)
axes[0, 0].plot(np.random.rand(5))
axes[0, 0].set_title("axes 1")
axes[0, 1].plot(np.random.rand(5))
axes[0, 1].set_title("axes 2")
axes[1, 0].plot(np.random.rand(5))
axes[1, 0].set_title("axes 3")
axes[1, 1].plot(np.random.rand(5))
axes[1, 1].set_title("axes 4")

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



서브 플롯 간의 간격 조절하기

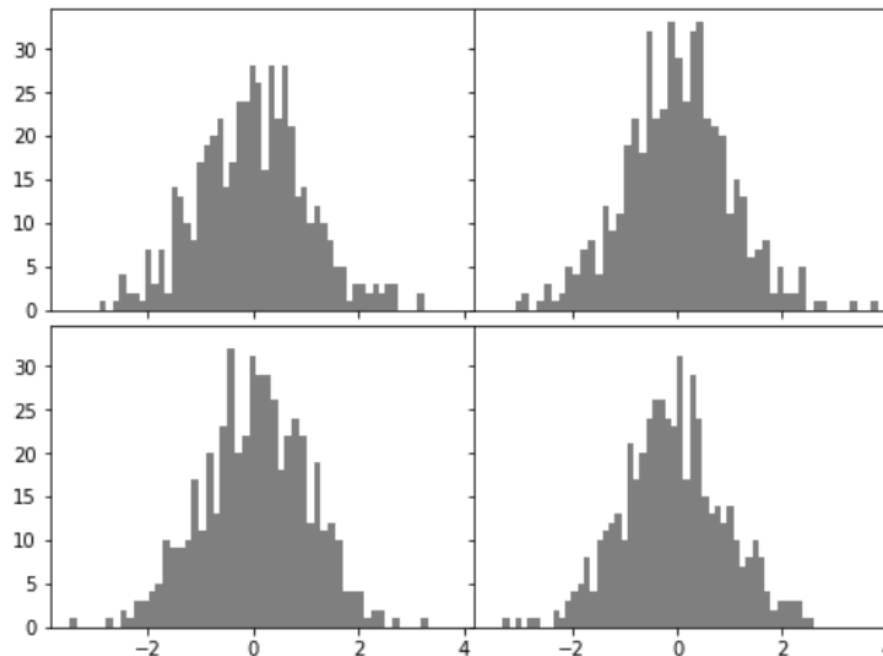
- `plt.subplots_adjust(wspace=0, hspace=.3)`

- 서브플롯 간의 간격을 설정

- 비율로 명시

- .3은 30%의 공간 비우기

```
In [61]: fig, axes = plt.subplots(2, 2, sharex=True, sharey=True, figsize=(8, 6))
         for i in range(2):
             for j in range(2):
                 axes[i, j].hist(np.random.randn(500), bins=50, color='k', alpha=0.5)
         plt.subplots_adjust(wspace=0, hspace=.05)
         #plt.subplots_adjust(wspace=None, hspace=None) #적정한 공간 비우기
```

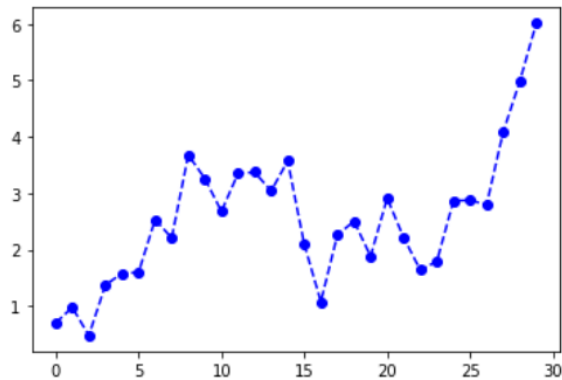


색상 마커 선 스타일

- **ax.plot(x, y, 'g--')**
 - ax.plot(x, y, linestyle='--', color='g')

```
In [66]: plt.plot(np.random.randn(30).cumsum(), color='b', linestyle='dashed', marker='o')
```

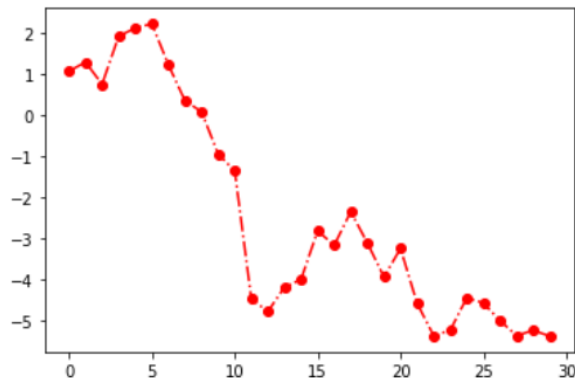
```
Out[66]: [<matplotlib.lines.Line2D at 0x23c0c615608>]
```



문자열	약자
blue	b
green	g
red	r
cyan	c
magenta	m
yellow	y
black	k
white	w

```
In [68]: plt.plot(np.random.randn(30).cumsum(), 'ro-.')
```

```
Out[68]: [<matplotlib.lines.Line2D at 0x23c0c541d48>]
```



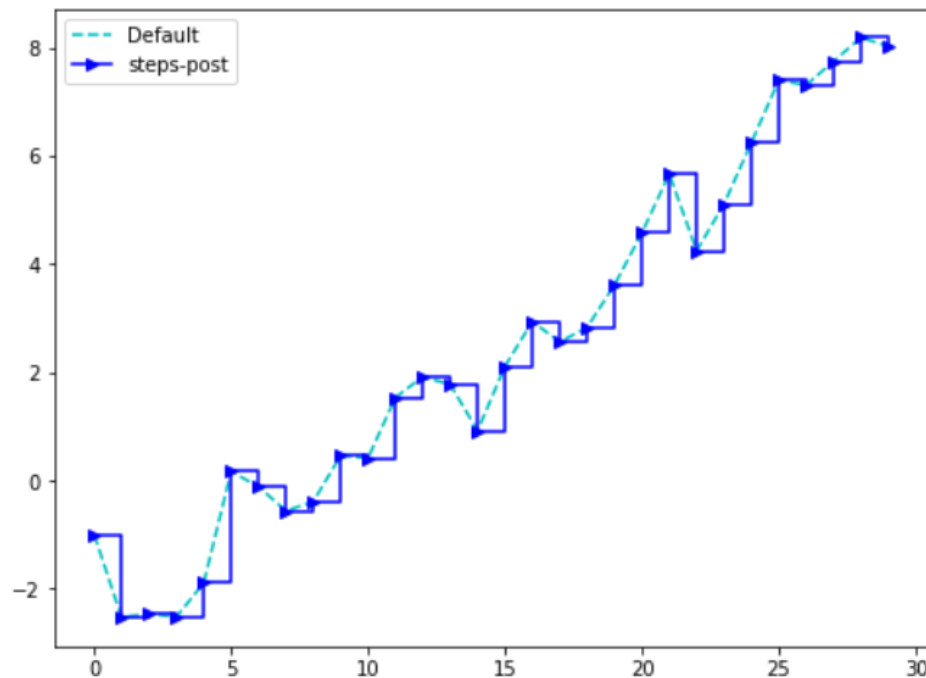
선 스타일 문자열	의미
-	solid line style
--	dashed line style
-.	dash-dot line style
:	dotted line style

옵션 **drawstyle**

- 일정한 간격으로 지정된 지점을 연결
 - drawstyle or ds: {'default', 'steps', 'steps-pre', 'steps-mid', 'steps-post'}
 - **default: 'default'**

```
In [77]: plt.figure(figsize=(8, 6))
data = np.random.randn(30).cumsum()
plt.plot(data, 'c--', label='Default')
plt.plot(data, '>b-', drawstyle='steps-post', label='steps-post')
plt.legend(loc='best')
```

Out[77]: <matplotlib.legend.Legend at 0x23c11063b48>



그래프를 꾸미는 방법 2 가지(1)

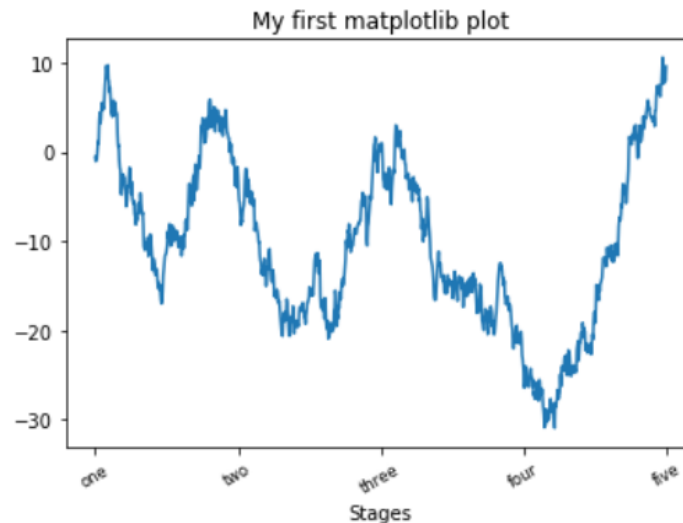
• 방법 1

- pyplot()으로 순차적으로
 - `ax =`
`fig.add_subplot(1, 1, 1)`
- 제공 API 사용
 - `set_title()`
 - `set_xlabel()`

```
In [89]: fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ticks = ax.set_xticks([0, 250, 500, 750, 1000])
labels = ax.set_xticklabels(['one', 'two', 'three', 'four', 'five'],
                             rotation=30, fontsize='small')
ax.plot(np.random.randn(1000).cumsum())

ax.set_title('My first matplotlib plot')
ax.set_xlabel('Stages')
```

Out[89]: Text(0.5, 0, 'Stages')



그래프를 꾸미는 방법 2 가지(2)

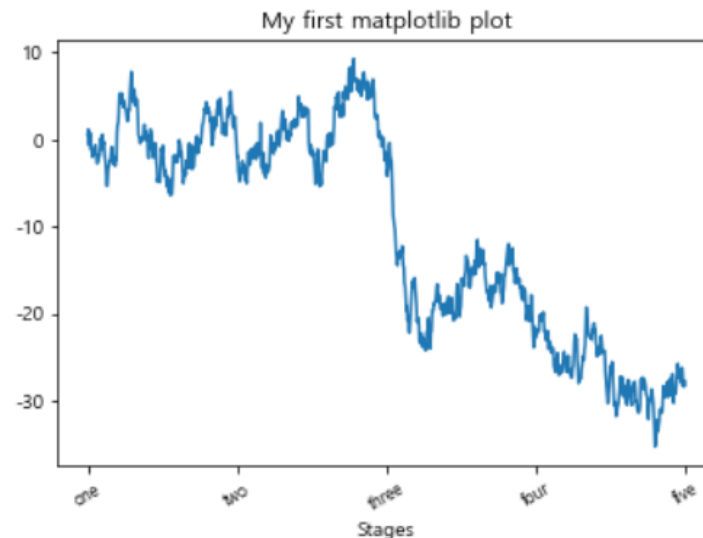
• 방법 2

- 클래스 axes의 메소드 set()
 - 여러 속성을 사전으로 지정

```
In [101]: fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ticks = ax.set_xticks([0, 250, 500, 750, 1000])
labels = ax.set_xticklabels(['one', 'two', 'three', 'four', 'five'],
                             rotation=30, fontsize='small')
ax.plot(np.random.randn(1000).cumsum())

#ax.set_title('엣플롯리브 그림')
#ax.set_xlabel('단계')
props = { 'title': 'My first matplotlib plot', 'xlabel': 'Stages' }
ax.set(**props)
```

```
Out[101]: [Text(0.5, 0, 'Stages'), Text(0.5, 1.0, 'My first matplotlib plot')]
```



범례 추가하기

- **legend()**
 - 위치 지정
 - **loc='best'**

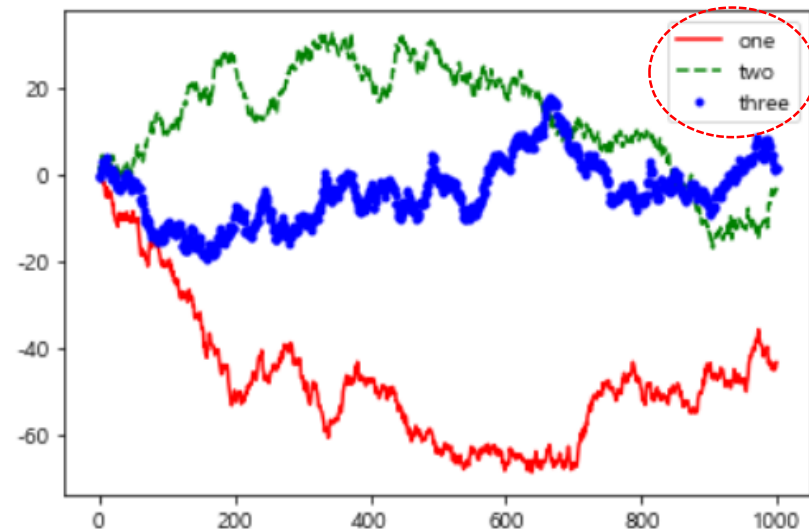
Location String	Location Code
'best'	0
'upper right'	1
'upper left'	2
'lower left'	3
'lower right'	4
'right'	5
'center left'	6
'center right'	7
'lower center'	8
'upper center'	9
'center'	10

Adding legends

```
In [109]: from numpy.random import randn
fig = plt.figure();
ax = fig.add_subplot(1, 1, 1)
ax.plot(randn(1000).cumsum(), 'r', label='one')
ax.plot(randn(1000).cumsum(), 'g--', label='two')
ax.plot(randn(1000).cumsum(), 'b.', label='three')

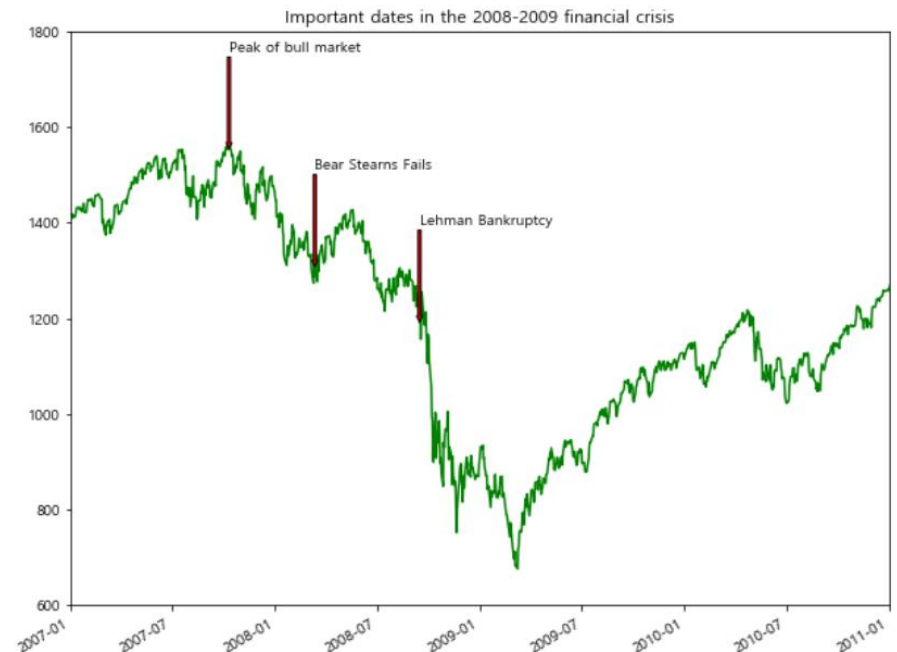
ax.legend(loc='best')
```

Out[109]: <matplotlib.legend.Legend at 0x23c125c9bc8>



주식 달기

- `matplotlib.pyplot.annotate(s, xy, *args, **kwargs)`
 - `ax.annotate(label, xy=(date, spx.asof(date) + 75),`
 - `xytext=(date, spx.asof(date) + 225),`
 - `arrowprops=dict(facecolor='blue', headwidth=4, width=2,`
 - `headlength=4),`
 - `horizontalalignment='left', verticalalignment='top')`
- `xy`: 주식의 위치
- `xytext`: 주식 글자의 위치
- 화살표는 `xytext`에서 `xy`로 그려짐



주식 화살표 그리기

- 옵션 arrowprops

In [131]:

```
import pandas as pd
from datetime import datetime

fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(1, 1, 1)

data = pd.read_csv('examples/spx.csv', index_col=0, parse_dates=True)
spx = data['SPX']

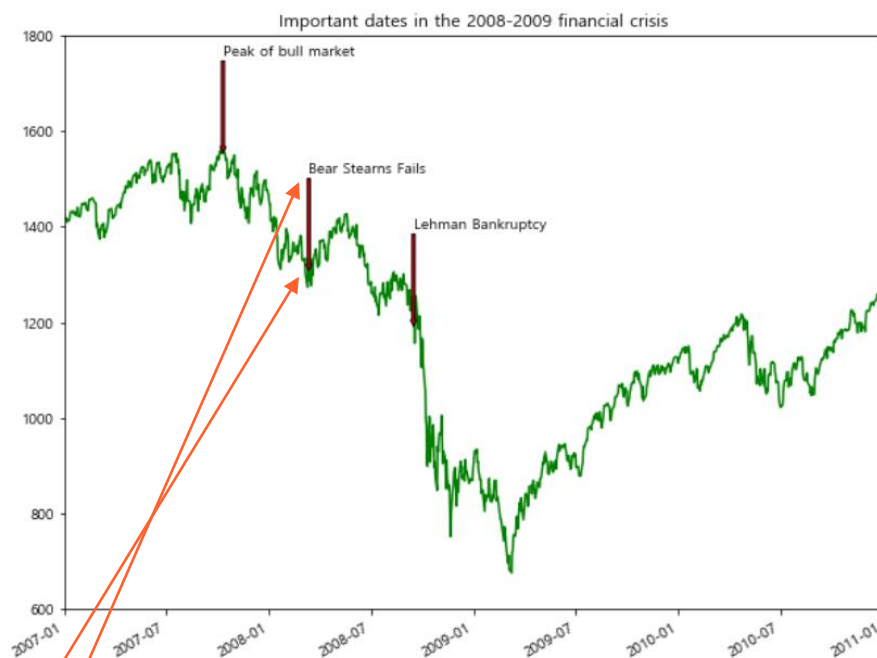
spx.plot(ax=ax, style='g-')

crisis_data = [
    (datetime(2007, 10, 11), 'Peak of bull market'),
    (datetime(2008, 3, 12), 'Bear Stearns Fails'),
    (datetime(2008, 9, 15), 'Lehman Bankruptcy')
]

for date, label in crisis_data:
    ax.annotate(label, xy=(date, spx.asof(date)),
                xytext=(date, spx.asof(date) + 225),
                arrowprops=dict(facecolor='red', headwidth=4, width=2,
                                headlength=4),
                horizontalalignment='left', verticalalignment='top')

# Zoom in on 2007-2010
ax.set_xlim(['1/1/2007', '1/1/2011'])
ax.set_ylim([600, 1800])

ax.set_title('Important dates in the 2008-2009 financial crisis')
```

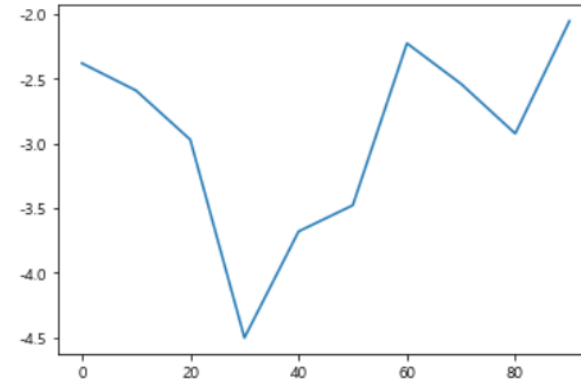


판다스 데이터 그리기

- 시리즈
- 데이터프레임

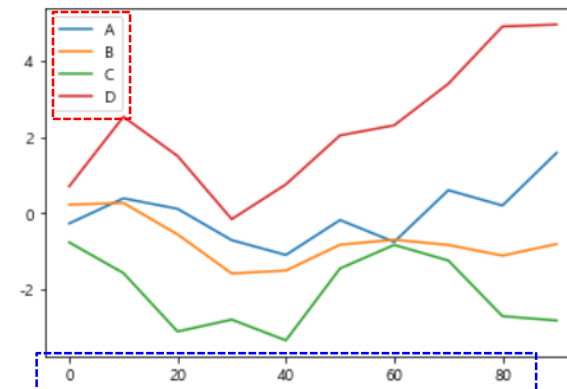
```
In [150]: import pandas as pd
s = pd.Series(np.random.randn(10).cumsum(), index=np.arange(0, 100, 10))
s.plot()
```

Out[150]: <matplotlib.axes._subplots.AxesSubplot at 0x23c18844908>



```
In [151]: df = pd.DataFrame(np.random.randn(10, 4).cumsum(0),
                             columns=['A', 'B', 'C', 'D'],
                             index=np.arange(0, 100, 10))
df.plot()
```

Out[151]: <matplotlib.axes._subplots.AxesSubplot at 0x23c188aa988>

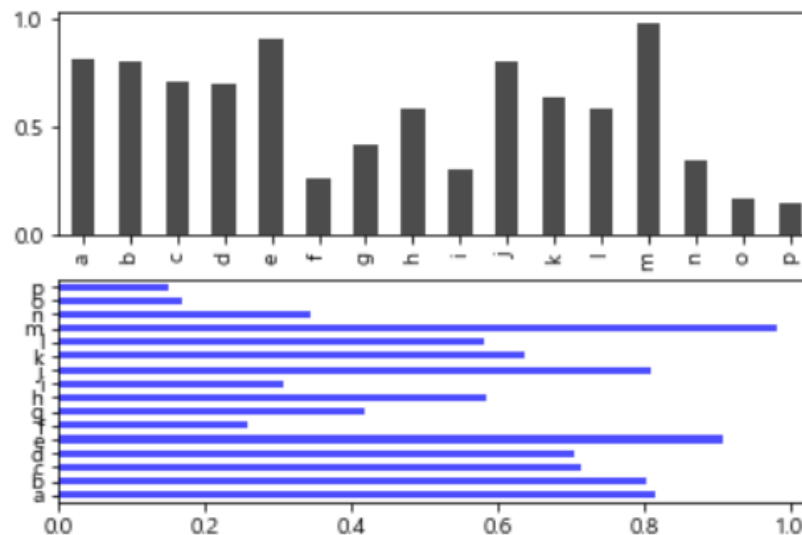


막대 그래프

- 메소드 `bar()`, `barh()`
 - 색상의 투명도 옵션 `alpha=`

```
In [153]: fig, axes = plt.subplots(2, 1)
data = pd.Series(np.random.rand(16), index=list('abcdefghijklmnop'))
data.plot.bar(ax=axes[0], color='k', alpha=0.7)
data.plot.barh(ax=axes[1], color='b', alpha=0.7)
```

Out[153]: <matplotlib.axes._subplots.AxesSubplot at 0x23c18a95b88>



데이터프레임

- 막대 그래프
 - index 이름이
범례 이름

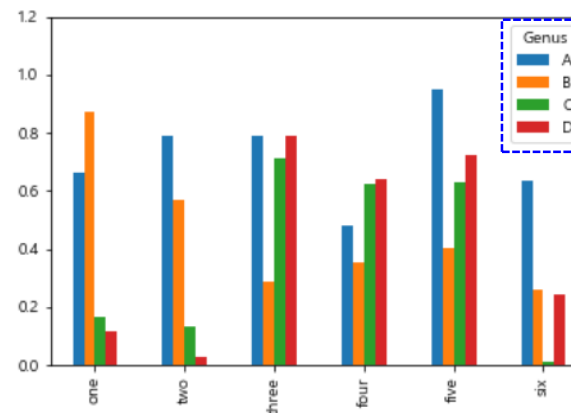
```
In [155]: df = pd.DataFrame(np.random.rand(6, 4),
                             index=['one', 'two', 'three', 'four', 'five', 'six'],
                             columns=pd.Index(['A', 'B', 'C', 'D'], name='Genus'))
df
```

```
Out[155]:
```

Genus	A	B	C	D
one	0.664737	0.872716	0.167458	0.114329
two	0.789941	0.569340	0.135387	0.029364
three	0.790764	0.285031	0.714794	0.790372
four	0.481273	0.356014	0.622913	0.641138
five	0.951389	0.405259	0.632018	0.724206
six	0.632867	0.259472	0.012550	0.240667

```
In [160]: df.plot.bar(ylim=[0, 1.2])
```

```
Out[160]: <matplotlib.axes._subplots.AxesSubplot at 0x23c1711e608>
```



누적 가로 막대

- 옵션 `stacked=True`

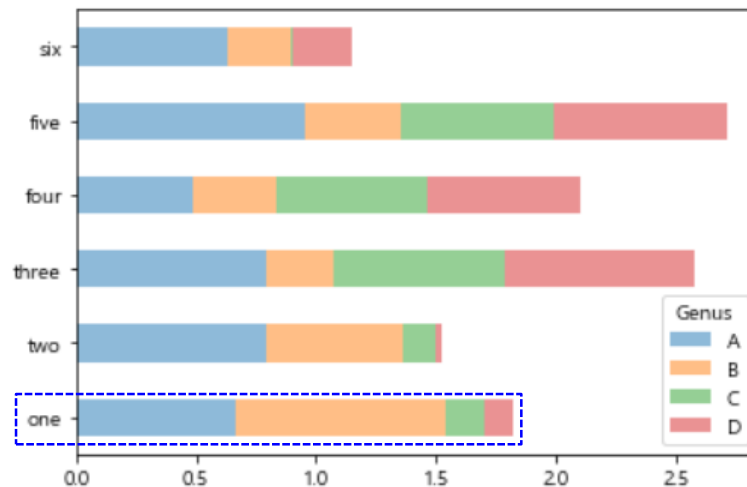
```
In [155]: df = pd.DataFrame(np.random.rand(6, 4),
                             index=['one', 'two', 'three', 'four', 'five', 'six'],
                             columns=pd.Index(['A', 'B', 'C', 'D'], name='Genus'))
df
```

```
Out[155]:
```

Genus	A	B	C	D
one	0.664737	0.872716	0.167458	0.114329
two	0.789941	0.569340	0.135387	0.029364
three	0.790764	0.285031	0.714794	0.790372
four	0.481273	0.356014	0.622913	0.641138
five	0.951389	0.405259	0.632018	0.724206
six	0.632867	0.259472	0.012550	0.240667

```
In [162]: df.plot.barh(stacked=True, alpha=0.5)
```

```
Out[162]: <matplotlib.axes._subplots.AxesSubplot at 0x23c14c65b08>
```



요일별 식당 인원 수 규모

- 각 행의 합이 1이 되도록 정규화
 - 요일별 식당 방문객 인원 수 비율

```
In [4]: tips = pd.read_csv('examples/tips.csv')
tips.head()
```

Out[4]:

	total_bill	tip	smoker	day	time	size
0	16.99	1.01	No	Sun	Dinner	2
1	10.34	1.66	No	Sun	Dinner	3
2	21.01	3.50	No	Sun	Dinner	3
3	23.68	3.31	No	Sun	Dinner	2
4	24.59	3.61	No	Sun	Dinner	4

```
In [6]: party_counts = pd.crosstab(tips['day'], tips['size'])
party_counts.head()
```

Out[6]:

size	1	2	3	4	5	6
day						
Fri	1	16	1	1	0	0
Sat	2	53	18	13	1	0
Sun	0	39	15	18	3	1
Thur	1	48	4	5	1	3

```
In [8]: # Normalize to sum to 1
party_pcts = party_counts.div(party_counts.sum(1), axis=0)
party_pcts.head()
```

Out[8]:

size	1	2	3	4	5	6
day						
Fri	0.052632	0.842105	0.052632	0.052632	0.000000	0.000000
Sat	0.022989	0.609195	0.206897	0.149425	0.011494	0.000000
Sun	0.000000	0.513158	0.197368	0.236842	0.039474	0.013158
Thur	0.016129	0.774194	0.064516	0.080645	0.016129	0.048387

```
In [7]: party_pcts.plot.bar()
```

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x2cf8d3f7c88>

