Matplotlib을 이용한 가시화

Matplotlib 특징

- 파이썬에서 가장 널리 사용되는 2차원 그래픽 라이브러리
- 논문 출판 수준의 우수한 품질
- MATLAB을 대체할 수 있는 강력한 도구 (pylab)
- 다양한 포맷 지원 (eps, jpg, png, svg, 등)
- 다양한 GUI 라이브러리 지원 (Tkinter, Qt, wxPython, GTK+)
- 철학: 쉬운 일은 쉽게, 어려운 일도 가능하게

Matplotlib 관련 정보 확인

버전 확인

In [1]:
```
import matplotlib
print(matplotlib.__version__)
```

```
2.0.0
```

설치 디렉토리 확인

In [2]:
```
print(matplotlib.__file__)
```

```
C:₩Users₩khkim₩Anaconda3₩lib₩site-packages₩matplotlib₩__init__.py
```

환경설정 및 캐시 디렉토리 확인

In [3]:
```
print(matplotlib.get_configdir())
print(matplotlib.get_cachedir())
```

```
C:₩Users₩khkim₩.matplotlib
C:₩Users₩khkim₩.matplotlib
```

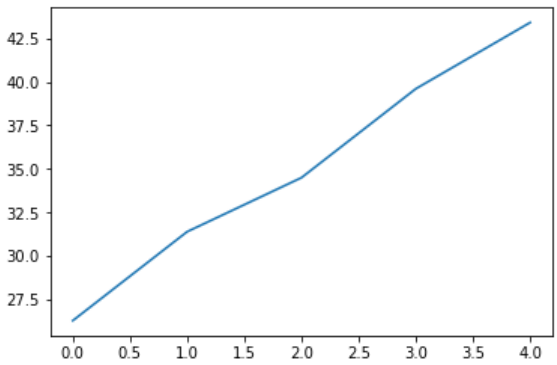
지원 포맷 확인

In [4]:
```
import matplotlib.pyplot as plt
#plt.gcf().canvas.get_supported_filetypes()
plt.gcf().canvas.get_supported_filetypes_grouped()
```

Out[4]:
```
{'Encapsulated Postscript': ['eps'],
 'Joint Photographic Experts Group': ['jpeg', 'jpg'],
 'PGF code for LaTeX': ['pgf'],
 'Portable Document Format': ['pdf'],
 'Portable Network Graphics': ['png'],
 'Postscript': ['ps'],
 'Raw RGBA bitmap': ['raw', 'rgba'],
 'Scalable Vector Graphics': ['svg', 'svgz'],
 'Tagged Image File Format': ['tif', 'tiff']}
```

첫 번째 예제

In [5]:
```
1 import matplotlib.pyplot as plt
2 plt.plot([26.3, 31.4, 34.5, 39.6, 43.4])
3 plt.savefig('first_sample.png')
4 plt.show()
```

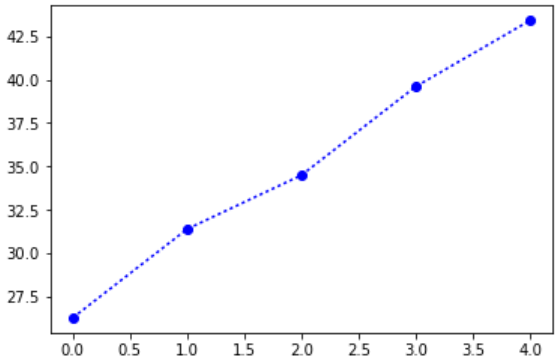


plot() 함수에 리스트 혹은 배열이 주어지면 위와 같은 그림을 그려준다. 인자로 주어진 리스트는 그래프의 y축 값이 되고, x축은 암시적으로 [0, 1, 2, 3, 4]가 된다. 우리는 5개의 값만을 주었지만 그림은 5개의 값을 직선으로 이어준다. savefig()는 파일로 저장해주며 주어진 확장자에 따라 이미지 포맷이 결정된다. savefig() 함수 옵션에는 dpi, facecolor, edgecolor, transparent 등등이 있다. show()는 화면에 그림을 보여주며 간단한 작업을 할 수 있는 사용자 메뉴도 제공한다.

plot() 포맷 형식

이제 주어진 5개의 값을 마커(marker)로 표시하고 실선은 점선으로 바꿔보자.

In [6]:
```
1 import matplotlib.pyplot as plt
2 y = [26.3, 31.4, 34.5, 39.6, 43.4]
3 plt.plot(y, 'bo:')
4 plt.show()
```



옵션 'bo-'에서 'b'는 blue 색상, 'o'는 원형마커, '-'는 실선을 의미한다. Matplotlib은 다음과 같은 포맷 문자들과 색상 문자들을 지원한다.

Format Parameters of pyplot.plot()

character	description
'-'	solid line style
'--'	dashed line style
'-.'	dash-dot line style
':'	dotted line style
'.'	point marker
','	pixel marker
'o'	circle marker

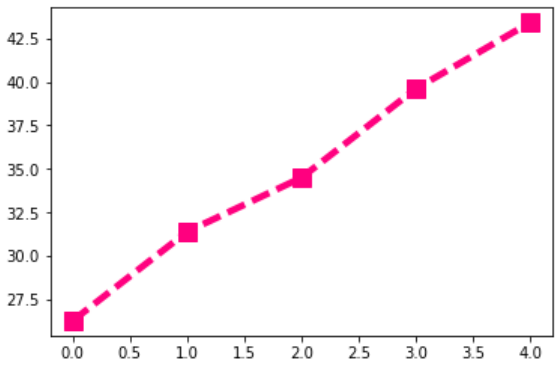
character	description
'v'	triangle_down marker
'^'	triangle_up marker
'<'	triangle_left marker
'>'	triangle_right marker
'1'	tri_down marker
'2'	tri_up marker
'3'	tri_left marker
'4'	tri_right marker
's'	square marker
'p'	pentagon marker
'*'	star marker
'h'	hexagon1 marker
'H'	hexagon2 marker
'+'	plus marker
'x'	x marker
'D'	diamond marker
'd'	thin_diamond marker
'|'	vline marker
'_'	hline marker

Color abbreviations

character	color
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

8개의 색상 문자들 외에 다양한 색상을 사용하고 싶다면 color 옵션으로 RGB FME 또는 HEX 형식으로 줄 수 있다(RGB 컬러 계산은 이곳 (http://www.rapidtables.com/web/color/RGB_Color.htm)을 참고). linewidth, markersize 옵션으로 선의 두께 및 마커 사이즈도 변경할 수 있다.

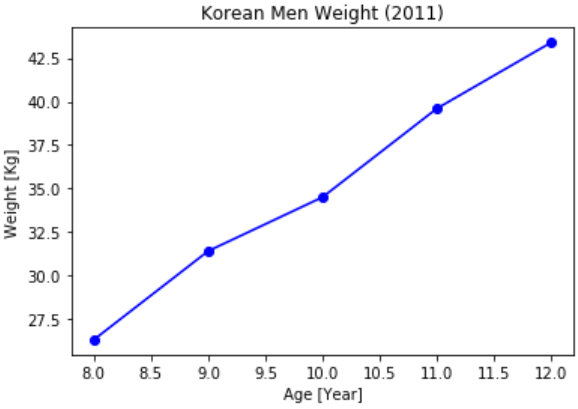
In [7]:
```
1 import matplotlib.pyplot as plt
2 y = [26.3, 31.4, 34.5, 39.6, 43.4]
3 plt.plot(y, 's--', color='#FF007F', linewidth=4.5, markersize=12)
4 plt.show()
```



제목과 축 라벨 달기

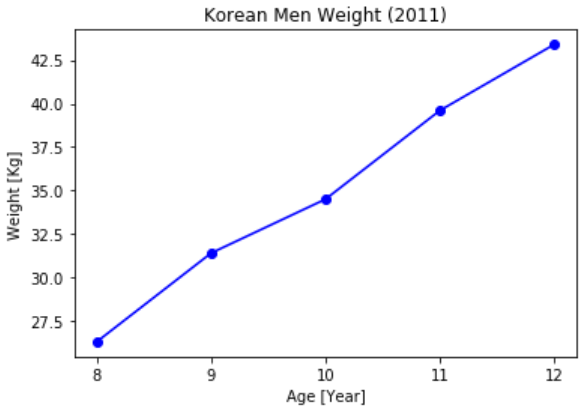
x축 값을 명시적으로 주고 각각의 축에 라벨을 달아보자.

In [8]:
```
1 import matplotlib.pyplot as plt
2 x = [8, 9, 10, 11, 12]
3 y = [26.3, 31.4, 34.5, 39.6, 43.4]
4 plt.plot(x, y, 'bo-')
5 plt.title('Korean Men Weight (2011)')
6 plt.xlabel('Age [Year]')
7 plt.ylabel('Weight [Kg]')
8 plt.show()
```



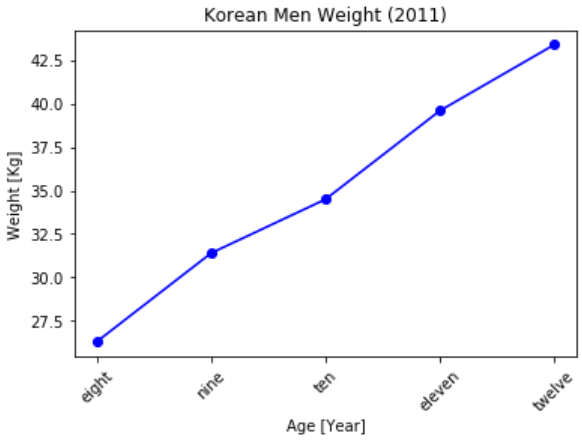
plot() 인자로 x, y 배열을 모두 주었다. title(), xlabel(), ylabel() 함수들은 이름 그대로 각각 제목, x축 라벨, y축 라벨을 지정해 준다. 그런데, x축 데이터가 연령이라서 x축 눈금값이 정수만 나오도록 하는 게 자연스러울 것 같다. xticks() 함수를 이용해서 눈금값을 변경해 보자.

In [9]:
```
import matplotlib.pyplot as plt
ages = [8, 9, 10, 11, 12]
weights = [26.3, 31.4, 34.5, 39.6, 43.4]
plt.plot(ages, weights, 'bo-')
plt.title('Korean Men Weight (2011)')
plt.xlabel('Age [Year]')
plt.ylabel('Weight [Kg]')
plt.xticks(ages)
plt.show()
```



x축 눈금값을 다음과 같이 문자열로 줄 수도 있고, 보기 좋게 회전시킬 수도 있다.

In [10]:
```
import matplotlib.pyplot as plt
ages = [8, 9, 10, 11, 12]
weights = [26.3, 31.4, 34.5, 39.6, 43.4]
plt.plot(ages, weights, 'bo-')
plt.title('Korean Men Weight (2011)')
plt.xlabel('Age [Year]')
plt.ylabel('Weight [Kg]')
plt.xticks(ages, ['eight', 'nine', 'ten', 'eleven', 'twelve'], rotation=45)
plt.show()
```



좀 더 많은 자료를 그려보자

다음은 2011년도 대한민국 남여 연령대별 평균키와 평균몸무게 자료이다. 편의상 CSV(Comma Separated Values) 형식 파일을 사용한다.

(옵션) 파일 생성

csv 모듈을 사용하여 파일을 생성한다.

In [11]:

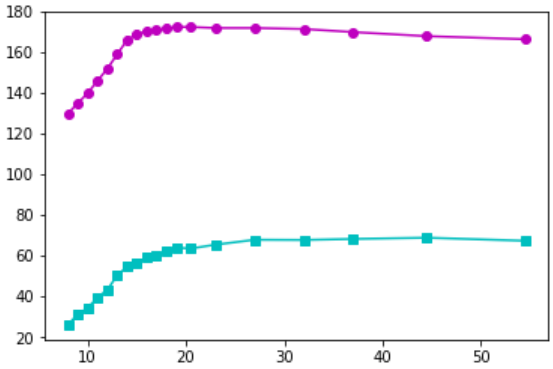
```
import csv

# age, height, weight
men = [
    [8, 130, 26.3],
    [9, 135, 31.4],
    [10, 140, 34.5],
    [11, 146, 39.6],
    [12, 152, 43.4],
    [13, 159, 50.4],
    [14, 166, 54.8],
    [15, 169, 56.6],
    [16, 170.5, 59.6],
    [17, 171, 60.4],
    [18, 172, 62.4],
    [19, 172.5, 64.1],
    ['20-21', 172.5, 63.7],
    ['22-24', 172, 65.6],
    ['25-29', 172, 68],
    ['30-34', 171.5, 67.9],
    ['35-39', 170, 68.4],
    ['40-49', 168, 69],
    ['50-59', 166.5, 67.5]]

women = [
    [8, 129, 26],
    [9, 136, 30.1],
    [10, 143, 34.6],
    [11, 149, 39],
    [12, 155, 43.4],
    [13, 157, 47.6],
    [14, 159, 50.4],
    [15, 161, 52],
    [16, 161.5, 52.3],
    [17, 162, 53],
    [18, 162, 52.8],
    [19, 162, 52.2],
    ['20-21', 161.5, 52.2],
    ['22-24', 161.5, 51.7],
    ['25-29', 160, 52.2],
    ['30-34', 159, 54.8],
    ['35-39', 158, 54.9],
    ['40-49', 157, 57.1],
    ['50-59', 154, 57.2]]

def save_to_csv(gender):
    filename = 'data/korean_{}_height_weight.csv'.format(gender)
    data = globals()[gender]
    with open(filename, 'w', newline='') as csvfile:
        cw = csv.writer(csvfile, delimiter=',')
        cw.writerow(['age', 'height', 'weight'])
        cw.writerows(data)

save_to_csv('men')
save_to_csv('women')
```

CSV 파일 읽고 그리기

대한민국 남성의 연령에 따른 키와 몸무게를 CSV 파일을 읽어서 그래프로 그려보자. CSV 파일 로딩은 편의상 Pandas의 read_csv() 함수를 이용하였다. 여기서 주의할 점은 age 값이 정수와 문자열이 섞여 있다는 것이다. 정수는 특정 나이인 반면에, 문자열은 나이의 범위를 표시하고 있다. 이대로 그래프를 그리면 에러가 나므로, 문자열은 '-'를 '+'로 치환한 후 계산하여 나이 범위의 중간값으로 변환시켰다.

In [12]:

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 df = pd.read_csv('data/korean_men_height_weight.csv', sep=',')
5 age_avg = [eval(s.replace('-','+'))/2 if '-' in s else int(s) for s in df.age]
6 plt.plot(age_avg, df.height, 'mo-')
7 plt.plot(age_avg, df.weight, 'cs-')
8 plt.show()
```



5번 라인은 파이썬의 List comprehension을 이용하였다. 풀어쓰면 다음과 같다.

age_avg = [] for s in df.age: if '-' in s: avg = eval(s.replace('-', '+'))/2 age_avg.append(avg) else: age_avg.append(int(s))

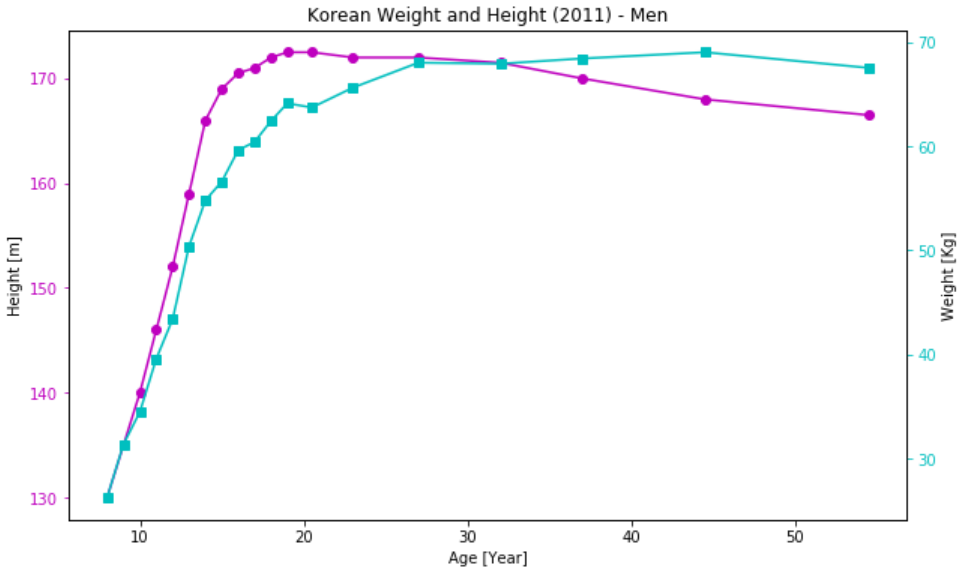
List comprehension은 모양도 간결하지만 성능도 더 좋으므로 적극적으로 활용하자.

두 개의 y축 겹쳐 그리기

6번, 7번 라인에서 height와 weight를 각각 그렸다. 그런데 두 개의 값의 단위도 다르고 범위도 다르다. 이런 경우에는 아래와 같이 두 개의 y축을 겹쳐서 그리는 것이 편리하다. 앞의 예제들에서는 단일 축만을 사용했는데, 아래와 같이 다중 축을 사용하는 경우에는 axis 객체를 직접 제어하는 것이 더 편리하다.

In [13]:

```
1  import pandas as pd
2  import matplotlib.pyplot as plt
3
4  df = pd.read_csv('data/korean_men_height_weight.csv', sep=',')
5  age_avg = [eval(s.replace('-','+'))/2 if '-' in s else int(s) for s in df.age]
6
7  fig = plt.figure(figsize=(10,6))
8  ax1 = fig.add_subplot(1,1,1)
9  ax1.plot(age_avg, df.height, 'mo-')
10 ax1.set_title('Korean Weight and Height (2011) - Men')
11 ax1.set_xlabel('Age [Year]')
12 ax1.set_ylabel('Height [m]')
13 ax1.tick_params('y', colors='m')
14
15 ax2 = ax1.twinx()
16 ax2.plot(age_avg, df.weight, 'cs-')
17 ax2.set_ylabel('Weight [Kg]')
18 ax2.tick_params('y', colors='c')
19
20 plt.show()
```



7번 라인의 figure에서 그림판 사이즈를 인치(inch) 단위로 변경할 수 있다.

8번 라인은 figure 안에 axis 객체(그림이 그려지는 x-y 축)를 하나 생성한다. (1,1,1)의 의미는 일반적으로 (NX,NY,i)이라고 했을 때, figure 안에 NXxNY 개의 subplot을 만들고 그 중에 i번째 axis이다라는 의미이다.

9번 라인은 첫 번째 axis 객체에 그림을 그린다.

10번-12번 라인은 제목과 x,y 라벨을 정해준다. 앞의 예제와 비교하면 set_ 접두어가 붙었다는 차이가 있다.

13번 라인에서 y축의 눈금 색깔을 그림 색상과 동일하게 맞춰주었다.

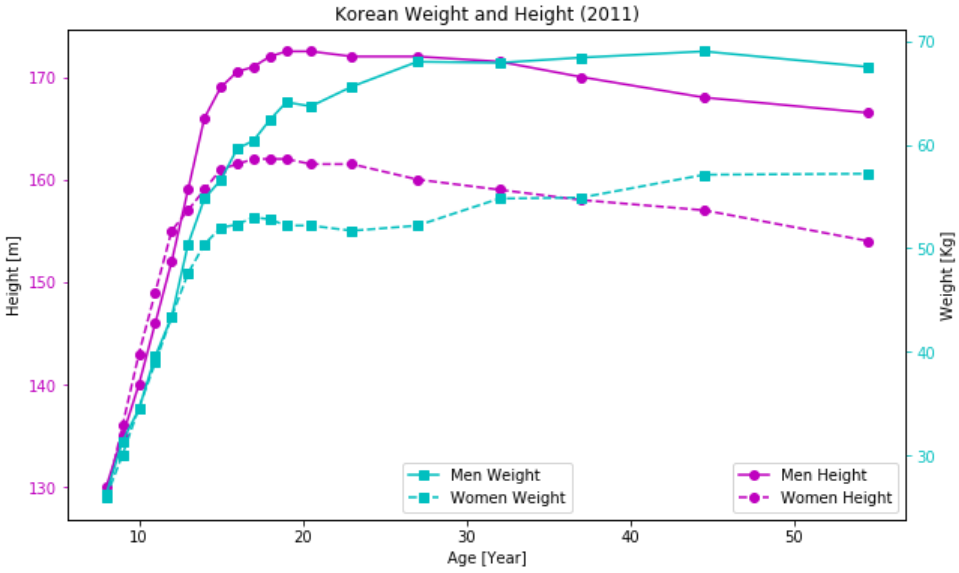
15번 라인에서 twinx() 함수는 첫 번째 axis 객체와 속성이 동일하면서 y축만 변화시킨 axis를 생성해준다.

남,여 데이터 함께 그리기, 범례(Legend) 추가

남성, 여성 데이터를 모두 그려보자. x축, y축이 모두 동일하므로 겹쳐 그려도 무방할 것 같다. 다만 남,여 데이터를 구분할 수 있게 범례(Legend)를 추가하면 더 좋을 것 같다.

In [14]:

```
1  import pandas as pd
2  import matplotlib.pyplot as plt
3
4  men = pd.read_csv('data/korean_men_height_weight.csv', sep=',')
5  women = pd.read_csv('data/korean_women_height_weight.csv', sep=',')
6  age_avg = [eval(s.replace('-','+'))/2 if '-' in s else int(s) for s in men.age]
7
8  fig = plt.figure(figsize=(10,6))
9  ax1 = fig.add_subplot(1,1,1)
10 ax1.plot(age_avg, men.height, 'mo-', label='Men Height')
11 ax1.plot(age_avg, women.height, 'mo--', label='Women Height')
12 ax1.set_title('Korean Weight and Height (2011)')
13 ax1.set_xlabel('Age [Year]')
14 ax1.set_ylabel('Height [m]')
15 ax1.tick_params('y', colors='m')
16 ax1.legend(loc='lower right')
17
18 ax2 = ax1.twinx()
19 ax2.plot(age_avg, men.weight, 'cs-', label='Men Weight')
20 ax2.plot(age_avg, women.weight, 'cs--', label='Women Weight')
21 ax2.set_ylabel('Weight [Kg]')
22 ax2.tick_params('y', colors='c')
23 ax2.legend(loc='lower center')
24
25 plt.show()
```



11번과 20번 라인에 여성 평균키와 평균몸무게 자료를 추가하였다.

10번,11번,19번,20번 라인에서 plot() 함수에 label 인자를 추가하였다. label 이름이 legend() 함수에서 사용된다.

16번, 23번 라인에서 Legend() 함수가 추가되었다. loc 인자는 범례의 위치를 지정하며, 아래 그림과 같이 문자열 또는 번호로 지정할 수 있다.

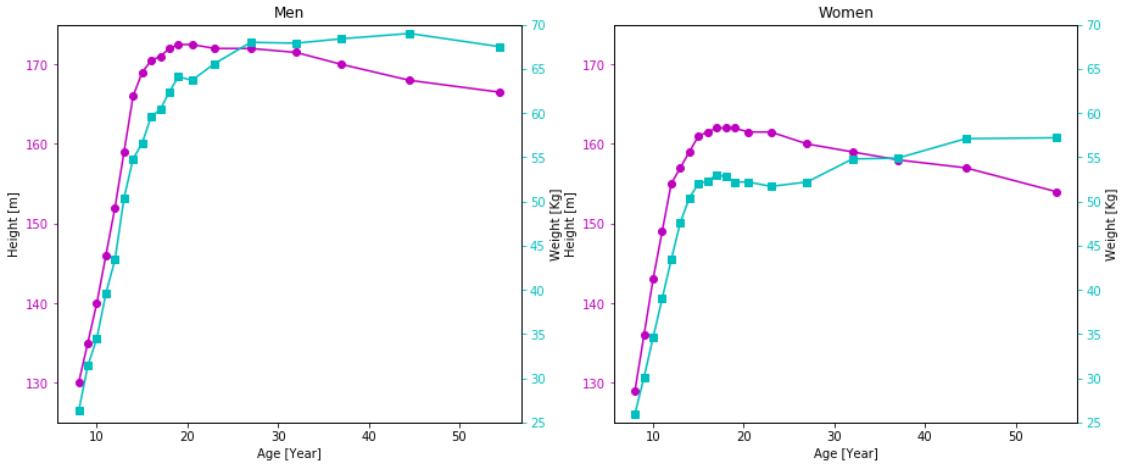
Matplotlib Legend Location

두 개의 subplot 함께 그리기

남성, 여성 데이터를 subplot을 두 개로 만들어서 그릴 수도 있다.

In [15]:

```
1  import pandas as pd
2  import matplotlib.pyplot as plt
3
4  def plot_height_weight(axis, data, age_avg):
5      axis.plot(age_avg, data.height, 'mo-')
6      axis.set_xlabel('Age [Year]')
7      axis.set_ylabel('Height [m]')
8      axis.set_ylim(125, 175)
9      axis.tick_params('y', colors='m')
10
11     axis2 = axis.twinx()
12     axis2.plot(age_avg, data.weight, 'cs-')
13     axis2.set_ylabel('Weight [Kg]')
14     axis2.set_ylim(25, 70)
15     axis2.tick_params('y', colors='c')
16
17 def main():
18     men = pd.read_csv('data/korean_men_height_weight.csv', sep=',')
19     women = pd.read_csv('data/korean_women_height_weight.csv', sep=',')
20     age_avg = [eval(s.replace('-','+'))/2 if '-' in s else int(s) for s in men.age]
21
22     fig = plt.figure(figsize=(15,6))
23     ax1 = fig.add_subplot(1,2,1)
24     ax2 = fig.add_subplot(1,2,2)
25     plot_height_weight(ax1, men, age_avg)
26     plot_height_weight(ax2, women, age_avg)
27     ax1.set_title('Men')
28     ax2.set_title('Women')
29     plt.show()
30
31 if __name__ == '__main__':
32     main()
```



가독성을 위해 두 개의 함수 plot_height_weight()와 main()으로 정리하였다.

23번, 24번 라인에서 subplot을 두 개 생성하였고, 인자로 주어진 (1,2,1)과 (1,2,2)는 subplot을 1개 행, 2개 열로 나누고 그 중에 첫 번째, 두 번째 subplot을 의미한다.

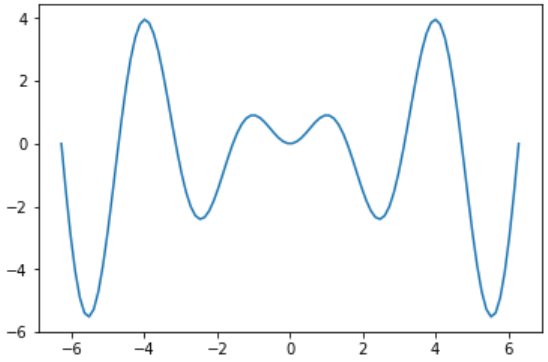
8번, 14번 라인에서 y축 범위를 동일하게 고정시키기 위해 set_ylim() 함수를 사용하였다.

수학 함수 그려보기

간단한 버전

In [16]:

```
from numpy import pi, sin
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-2*pi, 2*pi, 100, endpoint=True)
y = sin(2*x)*x

plt.plot(x, y)
plt.show()
```

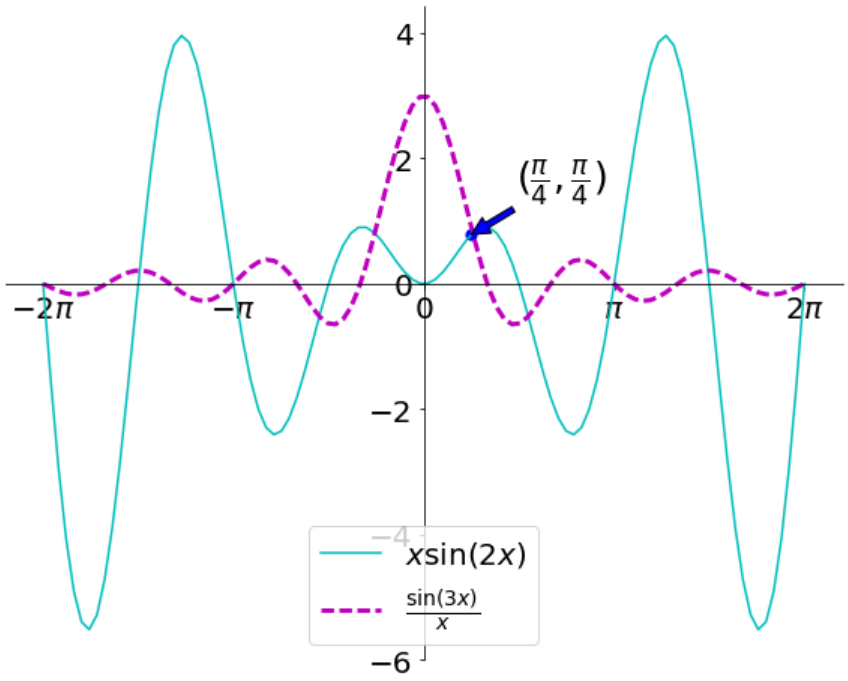


발전된 버전

- 그림판 사이즈 변경
- 2개 자료 함께 그리기
- 선, 마커 포맷 변경
- 범례(Legend) 표시
- 좌표축 옮기기
- 눈금에 수식 표시
- 눈금 폰트 크기 변경
- 수식을 포함한 주석 달기

In [17]:

```
from numpy import pi, sin
import numpy as np
import matplotlib.pyplot as plt

# data
x = np.linspace(-2*pi, 2*pi, 100, endpoint=True)
y1 = x*sin(2*x)
y2 = sin(3*x)/x

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

# plot with marker and line
ax.plot(x, y1, 'c-', label=r'$x\sin(2x)$')
ax.plot(x, y2, 'm--', linewidth=3, label=r'$\frac{\sin(3x)}{x}$')

# Legend
ax.legend(loc='lower center', fontsize=20)

# move axes
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['bottom'].set_position(('data', 0))
ax.spines['left'].set_position(('data', 0))

# set xticks with math formula
ax.set_xticks([-2*pi, -pi, 0, pi, 2*pi])
ax.set_xticklabels([r'$-2\pi$', r'$-\pi$', 0, r'$\pi$', r'$2\pi$'])

# set tick font sizes
ax.tick_params('both', labelsize=20)

# annotation
t = pi/4
ax.scatter([t], [t*sin(2*t)], 50, color='b')
ax.annotate(r'$(\frac{\pi}{4},\frac{\pi}{4})$',
            xy=(t, t*sin(2*t)),
            xycoords='data',
            textcoords='offset points',
            xytext=(30,30),
            fontsize=25,
            arrowprops=dict(facecolor='b'))

plt.show()
```



다른 파이썬 그래픽 라이브러리들

- mpld3 (http://mpld3.github.io) - Matplotlib in the web browsers using JavaScript
- Bokeh (https://bokeh.pydata.org) - Interactive visualization in the web browsers for presentation
- Plotly (https://plot.ly) - Interactive charts and dashboards to share online
- HoloViews (http://holoviews.org) - Data analysis and visualization seamless and simple

참고 자료

- Matplotlib Gallery (https://matplotlib.org/gallery.html)
- Python Course (http://www.python-course.eu)
- Scipy Lecture Notes (http://www.scipy-lectures.org)