파이썬 라이브러리 기본

학습 목표

- 딥러닝 기본을 학습하기 위한 파이썬 기본 및 기본 라이브러리를 이해한다.

- matplotlib, seaborn (시각화)
- numpy (수치 해석, 선형대수 등)
- pandas (데이터 처리)

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

학습 내용

- with 구문
- 리스트, 함수에 대한 이해
  - [num for n in range(10)]
  - {n:i for i, n in enumerate(char)}
- 여러개의 변수 넘겨받기 _
- numpy의 array()
- numpy의
  - reduce_mean(), log(), equal(), add(), argmax(), transpose(), matmul(), reshape()
- matplotlib 의 이해
  - 여러개의 그래프 그려보기
  - plt.subplots()의 이해
  - imshow() 이해
  - savefig() 이해
  - set_axis_off() 이해

```
하고자 하는 것.
try:
    실행1
finally:
    끝난이후에 실행
```

사용예

- file을 열고 사용한 이후에 닫는 것을 보장한다.
- 혹시 예외가 발생되어도 f파일은 자동으로 닫힌다.

```
with open("x.txt") as f:
  data = f.read()
  do something
```

리스트, 함수에 대한 이해

- [num for n in range(10)]
- {n:i for i, n in enumerate(char)}

In [6]:

```
a = [0,1,2,3,4,5,6,7,8,9]
a
```

Out[6]:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [7]:

```
a = [i for i in range(10)]
a
```

Out[7]:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [8]:

```
# 각각의 문자에 인덱스를 지정하고 싶다.
char = ['a', 'b', 'c', 'd', 'e']
for n, i in enumerate(char):
    print(n,i)
```

```
0 a
1 b
2 c
3 d
4 e
```

In [11]:

```
dic_n = {i:n for n, i in enumerate(char)}
dic_n
```

Out[11]:

```
{'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4}
```

In [12]:

```
### 여러개의 변수 넘겨받기. 단, 숫자는 큰 의미 없음.
for n, i in enumerate(char):
    _, n2, i = n, n*n, i
    print(n2, i)
```

```
0 a
1 b
4 c
9 d
16 e
```

In [ ]:

```
* numpy의 array()
* numpy의
  * reduce_mean(), log(), equal(), add(), argmax(), transpose(), matmul(), reshape()
```

numpy 알아보기

In [13]:

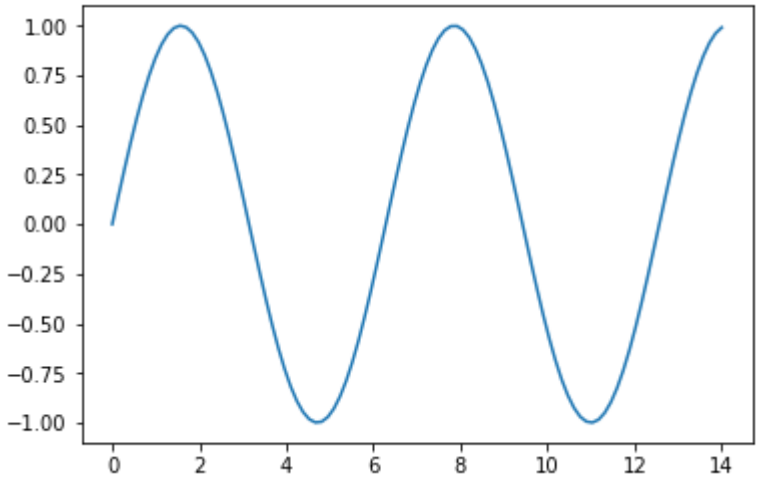
```
# x의 값을 이용하여 y값 (sin) 구하기
x = np.linspace(0,14,100)
y = np.sin(x)
```

In [14]:

```
# 그래프 그리기
plt.plot(x,y)
```

Out[14]:

```
[<matplotlib.lines.Line2D at 0x1513134a4a8>]
```



In [16]:

```
n = np.array([1,2,3,4,5])
n
```

Out[16]:

```
array([1, 2, 3, 4, 5])
```

In [17]:

```
print(type(n))
```

```
<class 'numpy.ndarray'>
```

- 2005년 Travis Oliphant가 발표한 수치해석용 패키지
- ndarray는 Numpy의 핵심인 다차원 행렬 자료구조 클래스이다.
- 파이썬이 제공하는 List자료형과 동일한 출력 형태를 갖는다.
- Numpy의 행렬 연산은 C로 구현된 반복문 사용으로 Python반복문에 비해 속도가 빠르다

Python의 리스트와 Numpy ndarray의 비고

Python List

- 여러가지 타입의 원소
- linked List 구현한 구조
- 메모리 용량이 크고 속도가 느림
- 벡터와 연산 불가

Numpy ndarray

- 동일 타입의 원소
- contiguous memory layout
  - (연속적인 메모리 배치(같은 타입) - 속도 향상
- 메모리 최적화, 속도 향상
- 벡터와 연산 가능

In [18]:

```
a = [1,2,3]
a*2
```

Out[18]:

```
[1, 2, 3, 1, 2, 3]
```

In [22]:

```
## 벡터 연산
a = np.array([1,2,3])
a * 2
```

Out[22]:

```
array([2, 4, 6])
```

In [30]:

```
a = np.arange(1000000)
%time a2 = a**2
```

```
Wall time: 3.64 ms
```

In [29]:

```
L = range(1000000)
%time L2 = [i**2 for i in L]
```

```
Wall time: 913 ms
```

In [31]:

```
### 다차원 생성하기
a1 = [1,2,3]
a2 = [ [11,12,13],
       [21,22,23],
       [31,32,33]]
a2
```

Out[31]:

```
[[11, 12, 13], [21, 22, 23], [31, 32, 33]]
```

In [34]:

```
b1 = np.array([1,2,3])
b2 = np.array([ [11,12,13],
                [21,22,23],
                [31,32,33]])
b2
```

Out[34]:

```
array([[11, 12, 13],
       [21, 22, 23],
       [31, 32, 33]])
```

In [37]:

```
print("행렬의 차원 : ", b1.ndim, b2.ndim)
print("행렬의 크기 : ", b1.shape, b2.shape)
```

```
행렬의 차원 :  1 2
행렬의 크기 :  (3,) (3, 3)
```

데이터 접근

In [39]:

```
a2[0]
```

Out[39]:

```
[11, 12, 13]
```

In [40]:

```
a2[0][1]
```

Out[40]:

```
12
```

In [43]:

```
print(b2[0])
print(b2[0][1])
print(b2[0,1])
```

```
[11 12 13]
12
12
```

In [44]:

```
### 전체 데이터 중의 짝수만 뽑기
b2[ b2 % 2 ==0]
```

Out[44]:

```
array([12, 22, 32])
```

In [45]:

```
### 10을 곱해보기
b2 * 10
```

Out[45]:

```
array([[110, 120, 130],
       [210, 220, 230],
       [310, 320, 330]])
```

matplotlib 의 이해

- 여러개의 그래프 그려보기
- plt.subplots()의 이해
- imshow() 이해
- savefig() 이해
- set_axis_off() 이해

In [46]:

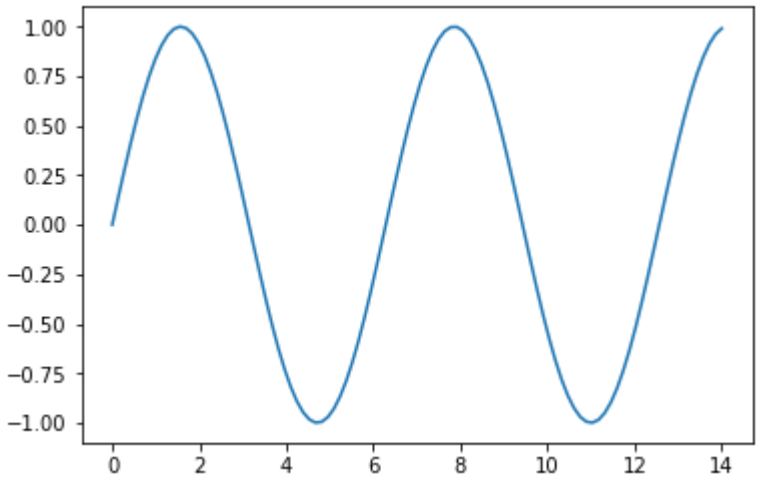
```
# x의 값을 이용하여 y값 (sin) 구하기
x = np.linspace(0,14,100)
y = np.sin(x)
```

In [47]:

```
# 그래프 그리기
plt.plot(x,y)
```

Out[47]:

```
[<matplotlib.lines.Line2D at 0x15134f8e0b8>]
```



그래프에 범례 표시해보기

In [48]:

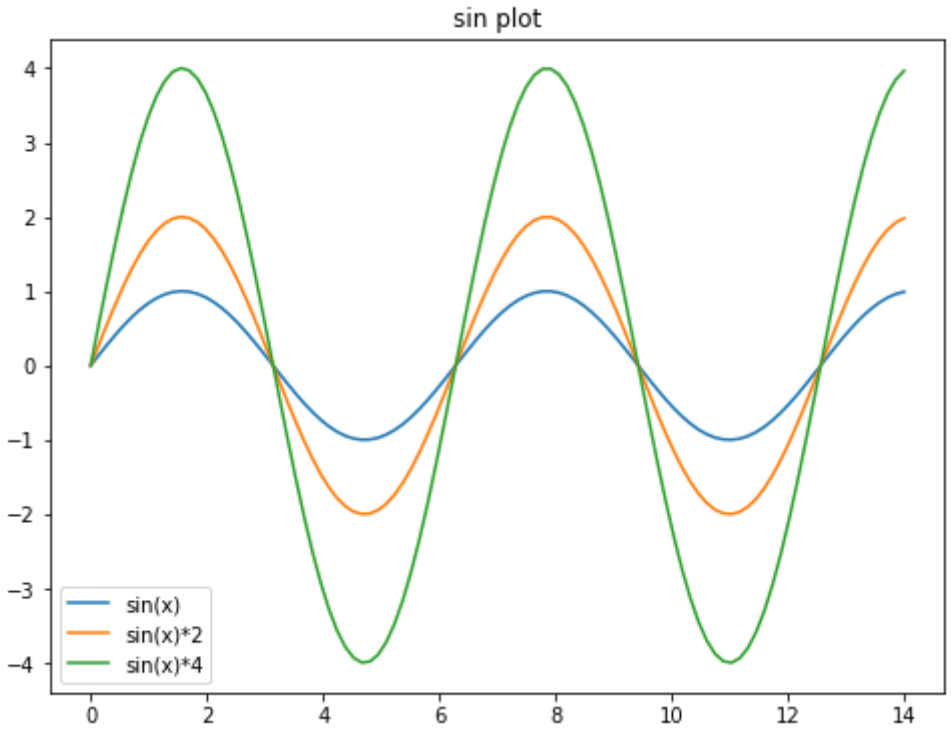
```
y1 = np.sin(x)
y2 = np.sin(x) * 2
y3 = np.sin(x) * 4
```

In [49]:

```
plt.figure(figsize=(8,6))
plt.plot(x, y1, label="sin(x)")
plt.plot(x, y2, label="sin(x)*2")
plt.plot(x, y3, label="sin(x)*4")
plt.legend()
plt.title("sin plot")
```

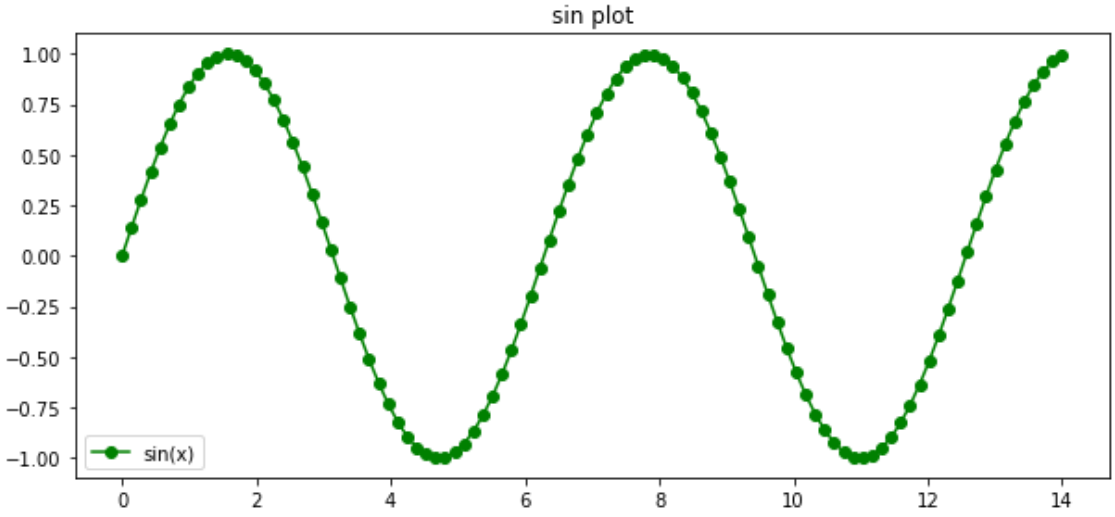
Out[49]:

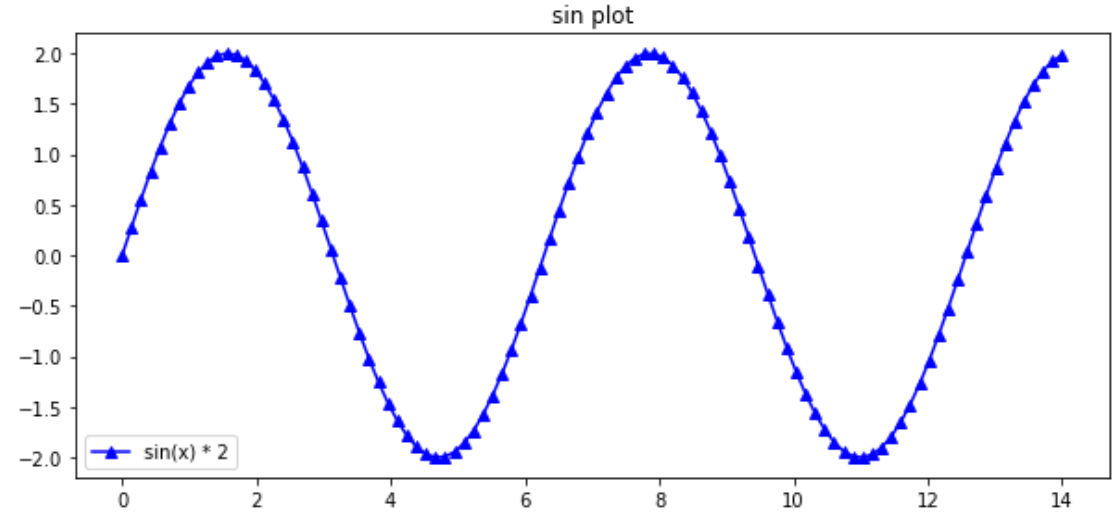
```
Text(0.5,1,'sin plot')
```

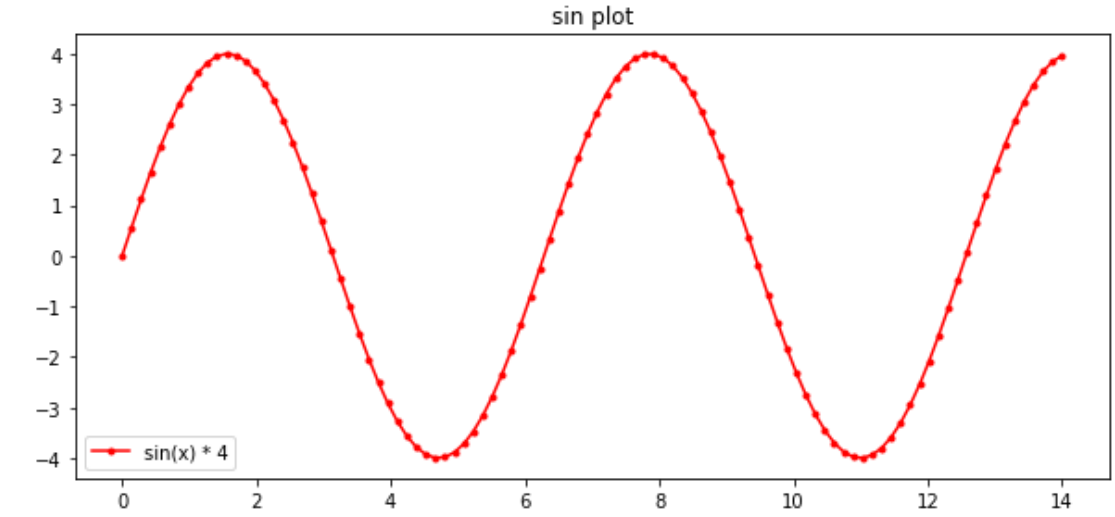


In [61]:

```
plt.figure(figsize=(10,15))
plt.subplot(3, 1, 1)
plt.plot(x, y1, 'go-', label='sin(x)')
plt.title("sin plot")
plt.legend()

plt.subplot(3, 1, 2)
plt.plot(x, y2, 'b^-', label='sin(x) * 2')
plt.title("sin plot")
plt.legend()

plt.subplot(3, 1, 3)
plt.plot(x, y3, 'r.-', label='sin(x) * 4')
plt.title("sin plot")
plt.legend()
plt.show()
```

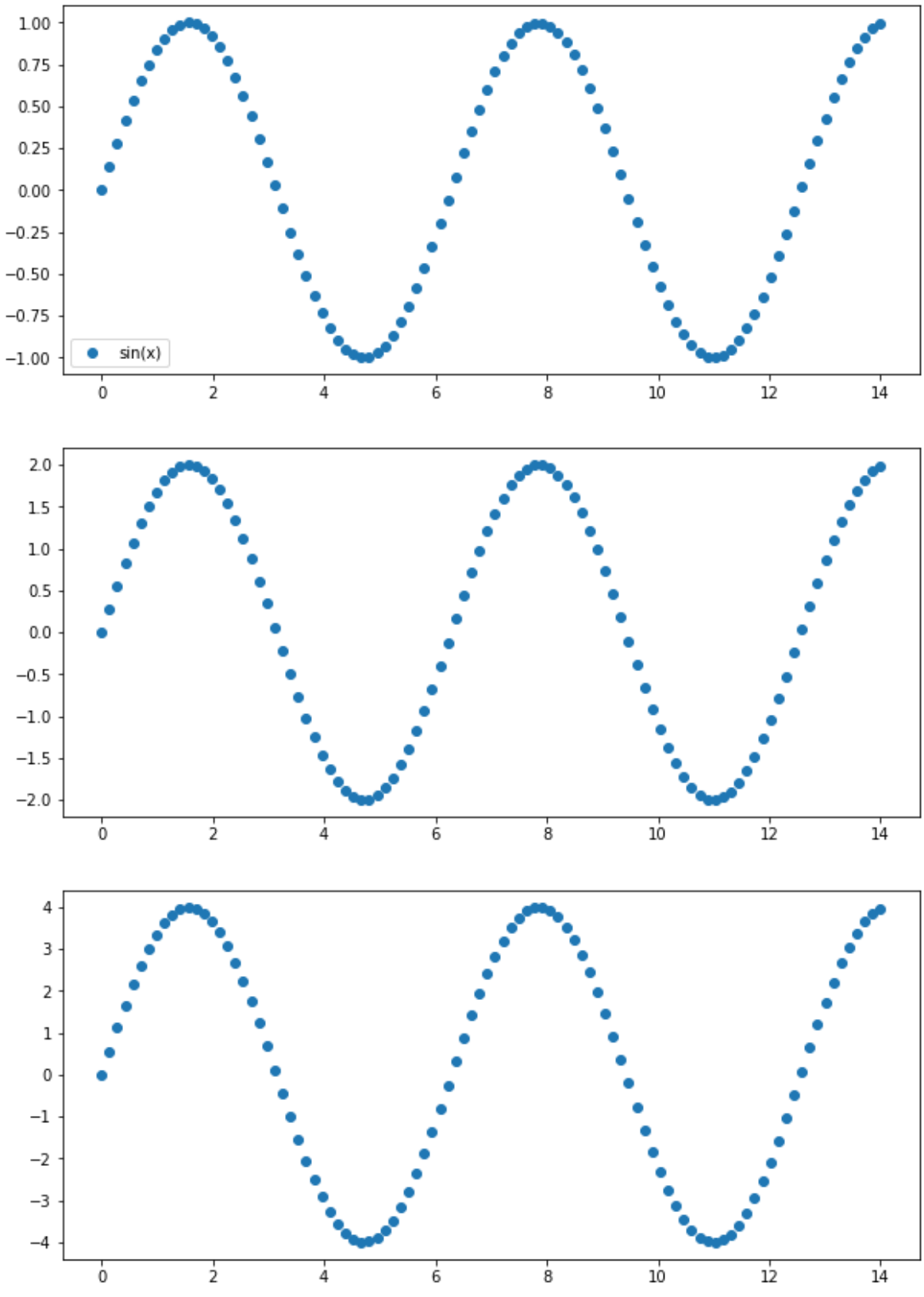

sin plot
sin(x)


sin plot
sin(x) * 2


sin plot
sin(x) * 4

In [69]:

```
fig, ax = plt.subplots(3, 1, figsize=(10,15))

ax[0].plot(x, y1, 'o', label='sin(x)')
ax[0].legend()
ax[1].plot(x, y2, 'o', label='sin(x)')
ax[2].plot(x, y3, 'o', label='sin(x)')
plt.show()
```



imshow() 이해

- 이미지의 표시해주기

In [73]:

```
import urllib.request

url = "https://pythonstart.github.io/web/dog01.jpg"
savename = 'dog.jpg'

urllib.request.urlretrieve(url,savename)
print("저장완료")
```

저장완료

'ls'은(는) 내부 또는 외부 명령, 실행할 수 있는 프로그램, 또는
배치 파일이 아닙니다.

In [77]:

```
!dir dog*
```

```
 C 드라이브의 볼륨에는 이름이 없습니다.
 볼륨 일련 번호: 6CB1-CD77

 C:₩Users₩front₩Documents₩00_TOTO_MLDL_CLASS 디렉터리

2020-02-17  오후 03:10         6,847,926 dog.jpg
               1개 파일           6,847,926 바이트
               0개 디렉터리  100,387,610,624 바이트 남음
```

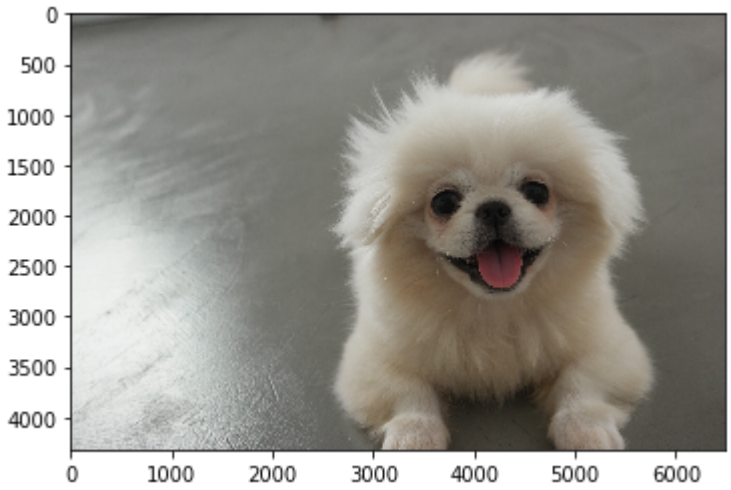
In [79]:

```
fig, ax = plt.subplots()
image = plt.imread("dog.jpg")  # 이미지 읽기
print(image)
ax.imshow(image)
```

```
[[[117 123 119]
  [118 124 120]
  [118 124 120]
  ...
  [116 113 108]
  [118 113 109]
  [117 112 108]]

 [[117 123 119]
  [118 124 120]
  [119 125 121]
  ...
  [117 114 109]
  [117 114 109]
  [117 114 109]]

 [[117 122 118]
  [118 123 119]
  [118 124 120]
  ...
  [116 113 108]
  [115 112 107]
  [116 113 108]]

 ...

 [[186 186 188]
  [186 186 188]
  [187 187 189]
  ...
  [ 83  80  75]
  [ 82  79  74]
  [ 80  77  72]]

 [[184 184 184]
  [184 184 184]
  [186 186 186]
  ...
  [ 81  78  73]
  [ 81  78  73]
  [ 80  77  72]]

 [[181 181 179]
  [181 181 179]
  [184 184 182]
  ...
  [ 80  77  70]
  [ 81  78  71]
  [ 79  76  69]]]
```

Out[79]:

```
<matplotlib.image.AxesImage at 0x1513542c3c8>
```



In [85]:

```
### set_axis_off() 이해     - 축을 없애기
### savefig() 이해          - 파일로 저장
ax.set_axis_off()
ax.imshow(image)
fig.savefig('Plot_without_axes.png')
```

In [83]:

```
!dir Plot_*
```

```
 C 드라이브의 볼륨에는 이름이 없습니다.
 볼륨 일련 번호: 6CB1-CD77

 C:₩Users₩front₩Documents₩00_TOTO_MLDL_CLASS 디렉터리

2020-02-17  오후 03:33           133,049 Plot_without_axes.png
               1개 파일             133,049 바이트
               0개 디렉터리  100,385,861,632 바이트 남음
```

Seaborn에 대해 알아보기

In [86]:

```
import seaborn as sns
import pandas as pd
```

In [87]:

```
tips = sns.load_dataset("tips")  ## tips의 데이터 셋 불러오기
tips.head()  ## 앞의 데이터만 살펴보기  tips.tail() - 뒤의 데이터 셋
```

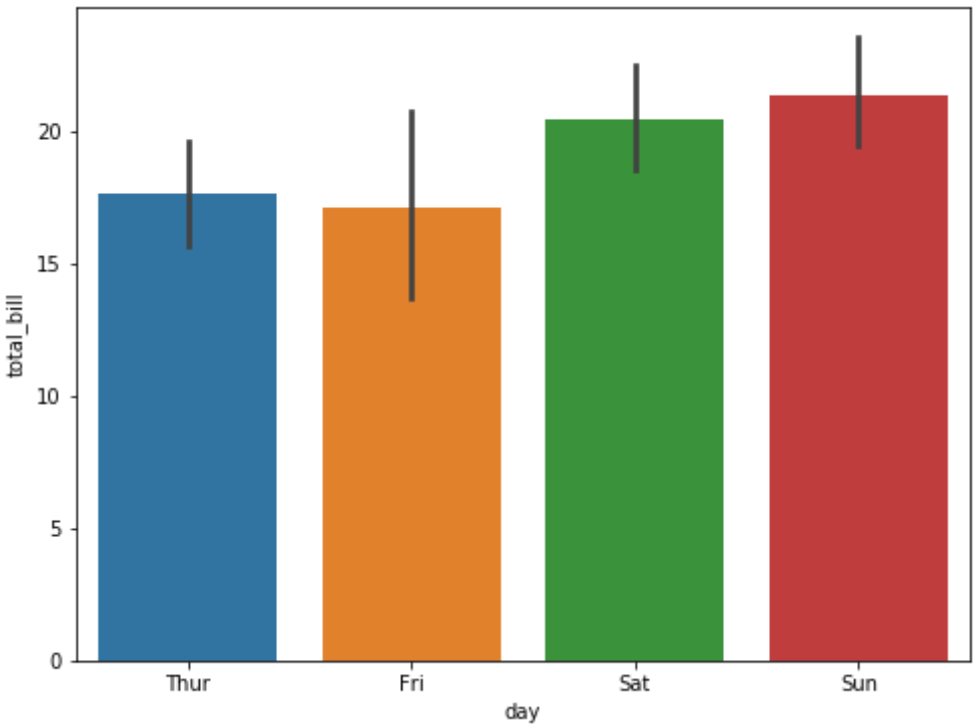
Out[87]:

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

In [88]:

```
### 요일별 식사 금액은 얼마나 될까?
plt.figure(figsize=(8,6))
sns.barplot(x="day", y="total_bill", data=tips)
plt.show()
```

```
C:₩ProgramData₩Anaconda3₩lib₩site-packages₩scipy₩stats₩stats.py:1713: FutureWarning:
Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tup
le(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array
index, `arr[np.array(seq)]`, which will result either in an error or a different res
ult.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

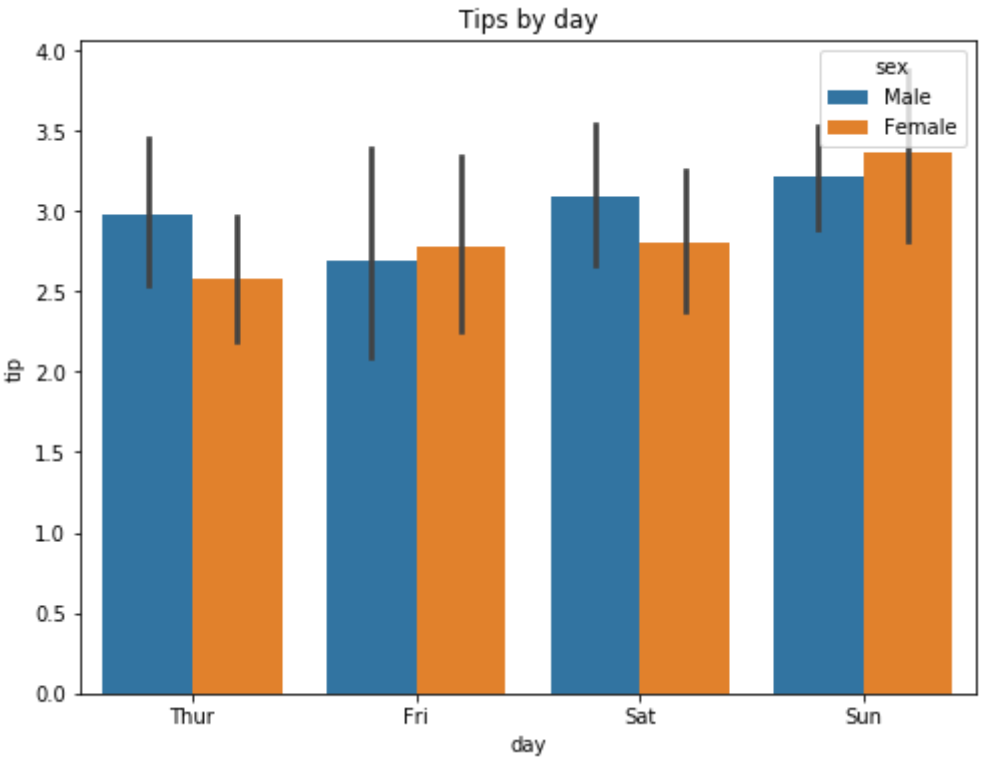


(실습) 요일별 Tips은 얼마나 될까?

In [89]:

```
# 요일별 tip은 남성과 여성은 어떠할까?
plt.figure(figsize=(8,6))
sns.barplot(x="day", y="tip", hue="sex", data=tips)  # hue를 이용하여 구분.
plt.title("Tips by day")
plt.show()
```

```
C:₩ProgramData₩Anaconda3₩lib₩site-packages₩scipy₩stats₩stats.py:1713: FutureWarning:
Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tup
le(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array
index, `arr[np.array(seq)]`, which will result either in an error or a different res
ult.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



pandas 기본

- 데이터를 불러와 csv 또는 excel 데이터셋 만들기

데이터 준비 및 csv 파일 생성

In [90]:

```
iris = sns.load_dataset("iris")
iris.head()
```

Out[90]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [91]:

```
## csv 데이터 셋, excel 데이터 셋 만들기
iris.to_csv("iris.csv", index=False)
iris.to_excel("iris.xlsx", index=False)
```

In [92]:

```
!ls   # 경로안의 데이터 셋 확인
```

```
'ls'은(는) 내부 또는 외부 명령, 실행할 수 있는 프로그램, 또는
배치 파일이 아닙니다.
```

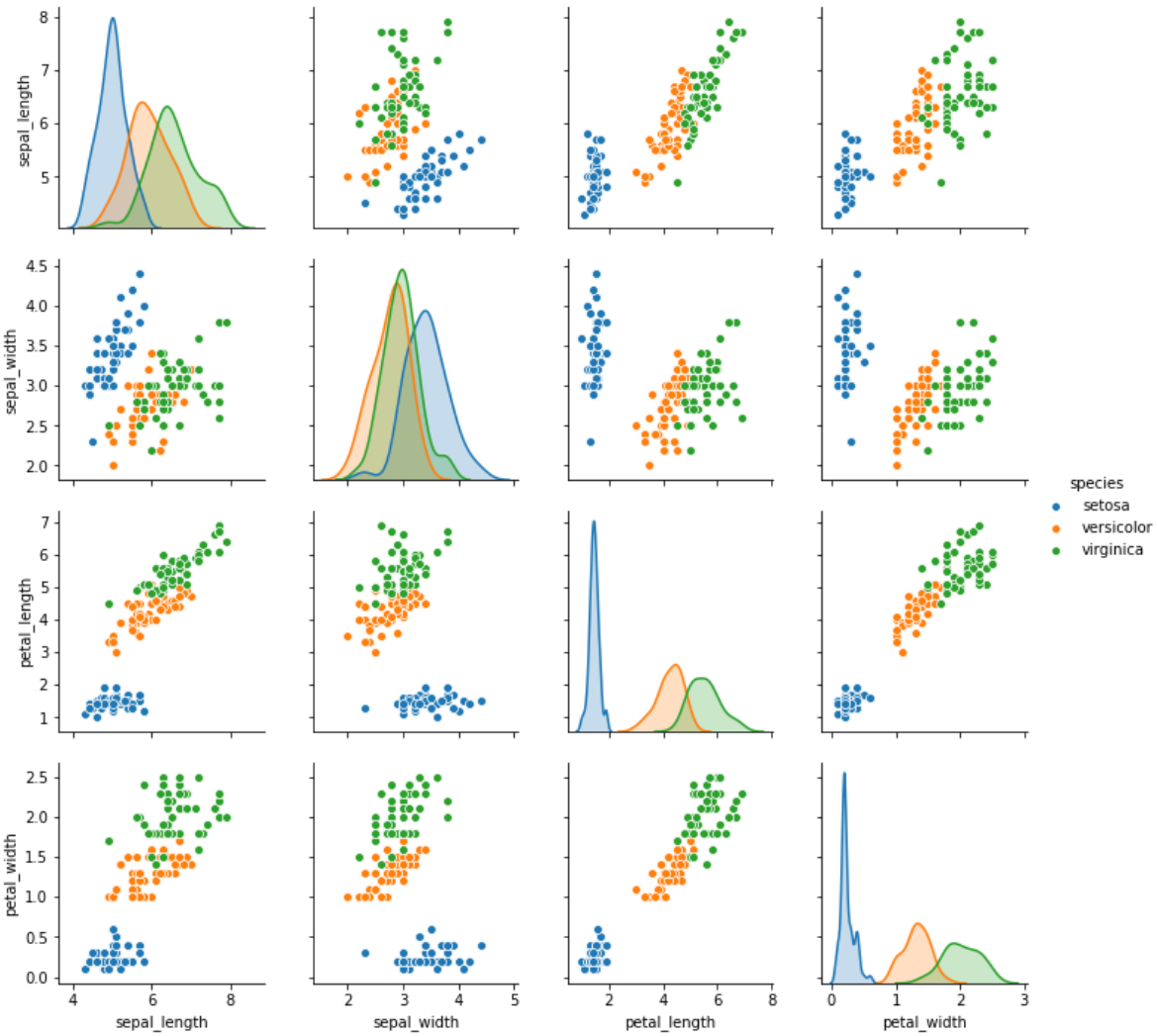
In [93]:

```
## 산점도 그래프 확인
sns.pairplot(iris, hue="species")
```

```
C:₩ProgramData₩Anaconda3₩lib₩site-packages₩scipy₩stats₩stats.py:1713: FutureWarning:
Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tup
le(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array
index, `arr[np.array(seq)]`, which will result either in an error or a different res
ult.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[93]:

```
<seaborn.axisgrid.PairGrid at 0x15135882320>
```



1-2 Pandas에 대해 알아보기

- 리스트, 딕셔너리 간단히 만들어보기
- Pandas 불러오기
- 행과 열 확인 shape
- 컬럼명 확인 columns
- 정보 확인 info()
- 요약값 확인 describe()
- 몇행만 확인하기 head(), tail()
- Series 자료형, DataFrame 자료형 만들기
- 원하는 열 선택(하나)
- 원하는 열 선택(여러개)
- 원하는 행 선택(하나)
- 원하는 행 선택(여러개)
- 새로운 컬럼 생성

In [94]:

```
# 리스트
myfood = ['banana', 'apple', 'candy']
print(myfood[0])
print(myfood[1])
print(myfood[2])
print(myfood[1:3])
```

```
banana
apple
candy
['apple', 'candy']
```

In [95]:

```
for item in myfood:
    print(item)
```

```
banana
apple
candy
```

In [96]:

```
# 딕셔너리
dict1 = {'one':'하나', 'two':"둘", 'three':'셋'}
dict2 = {1:"하나", 2:"둘", 3:"셋"}
dict3 = {'col1':[1,2,3], 'col2':['a','b','c']}
print(dict1)
print(dict2)
print(dict3)
```

```
{'one': '하나', 'two': '둘', 'three': '셋'}
{1: '하나', 2: '둘', 3: '셋'}
{'col1': [1, 2, 3], 'col2': ['a', 'b', 'c']}
```

In [28]:

```
print(dict1['one'])
print(dict2[2])
print(dict3['col2'])
```

```
하나
둘
['a', 'b', 'c']
```

In [101]:

```
dat = pd.read_csv("iris.csv")
dat.shape  # 데이터의 행열 (크기)
```

Out[101]:

```
(150, 5)
```

In [103]:

```
dat.head()  ## 앞의 데이터 보기
```

Out[103]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [104]:

```
dat.tail()  ## 뒤의 데이터 보기
```

Out[104]:

	sepal_length	sepal_width	petal_length	petal_width	species
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

In [105]:

```
dat.columns   ## 데이터의 컬럼보기
```

Out[105]:

```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

In [108]:

```
dat.describe()  ## 데이터의 요약값 보기
```

Out[108]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [109]:

```
dat.info()  # 데이터의 자료형 요약 보기
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
sepal_length    150 non-null float64
sepal_width     150 non-null float64
petal_length    150 non-null float64
petal_width     150 non-null float64
species         150 non-null object
dtypes: float64(4), object(1)
memory usage: 5.9+ KB
```

In [111]:

```
dat.isnull().sum()  # 데이터의 결측치 보기
```

Out[111]:

```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

In [ ]:

```

```