Lecture 10. Matplotlib 활용(1)

기초 데이터 분석

Recap : Matplotlib Plot

Plot 함수들

- **Plot() 함수 : Line Plot**
- **Scatter() : 산점도**
- **Bar() : 바 그래프**

그외 pyplot 관련 함수

- **subplots()**
- **title()**
- **xlabel(), xlim()**
- **avxline(), ahxline()**
- **legend(), grid(True)**

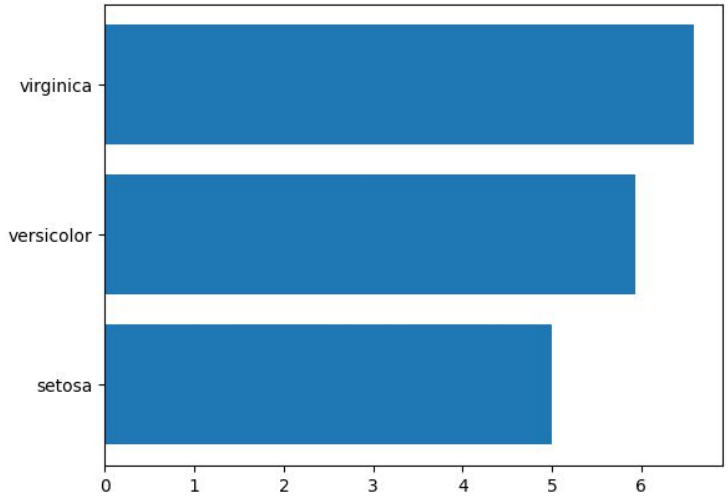
Today : Matplotlib의 활용

- 수평 막대 그래프 (Barh)
- 누적 막대 그래프 (Stacked Bar)
- 박스 그래프 (Box)
- 히스토그램 (Histogram)
- 커널밀도함수 (KDE)
- 바이올린 플롯
- Axes3D

수평 막대 그래프 (barh)

- 수직 형태의 Bar 그래프 사용이 불가하능 할 경우
- 범주별 데이터를 수평 막대로 표현
- barh() 함수를 사용
- bar() 함수와 다르게 y와 x 순서대로 받아옴

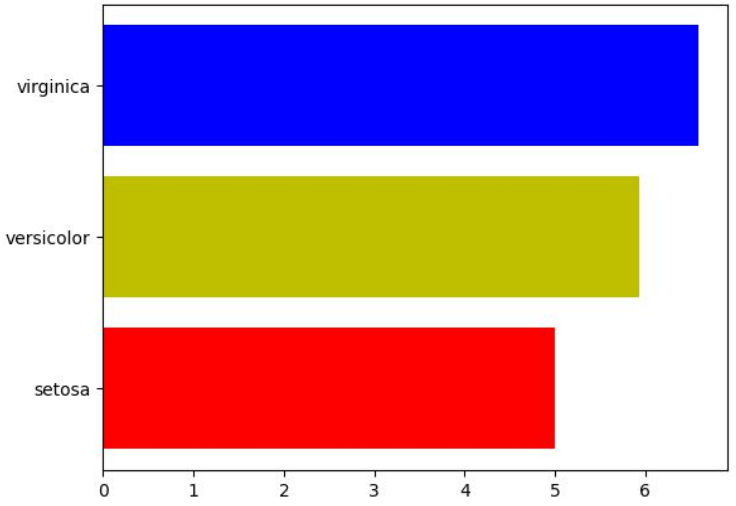
```
# iris 종에 따른 sepal_length의 평균
x = iris.groupby('species').sepal_length.mean().values
# iris 종 목록
y = iris.groupby('species').sepal_length.mean().index
plt.barh(y,x) # barh
```



수평 막대 그래프 (barh)

- 수직 형태의 **Bar** 그래프 사용이 불가능 할 경우
- 범주별 데이터를 수평 막대로 표현
- **barh()** 함수를 사용
- bar() 함수와 다르게 **y와 x** 순서대로 받아옴
- bar() 과 동일하게 **color** 인자를 변경가능

```
# iris 종에 따른 sepal_length의 평균
x = iris.groupby('species').sepal_length.mean().values
# iris 종 목록
y = iris.groupby('species').sepal_length.mean().index
color = ['r','y','b'] # 색상 순서
plt.barh(y,x,color=color) # barh
```

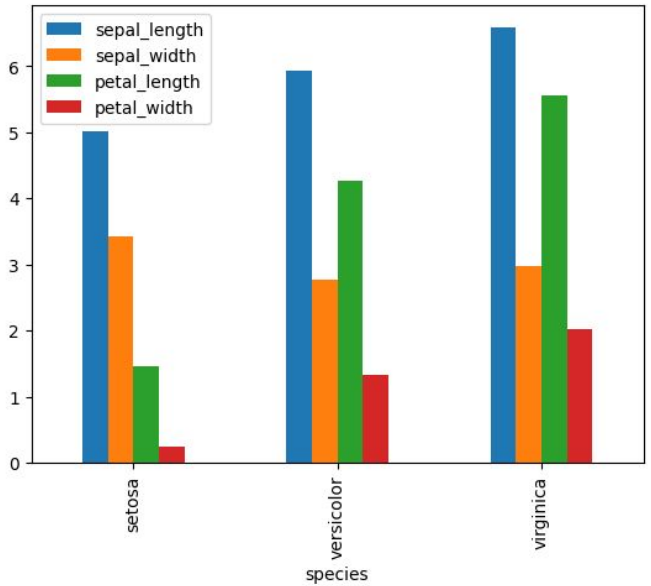


누적 막대 그래프

- Bor 그래프에서 하나의 bar 가 몇가지 category로 나누어지는 경우
- Pandas.plot.bar() 을 활용
- 여러 범주의 데이터를 쌓은 Stacked Bar 형태로 표현

- 기존 iris species 정보를 포함하는 sepal_length를 표현

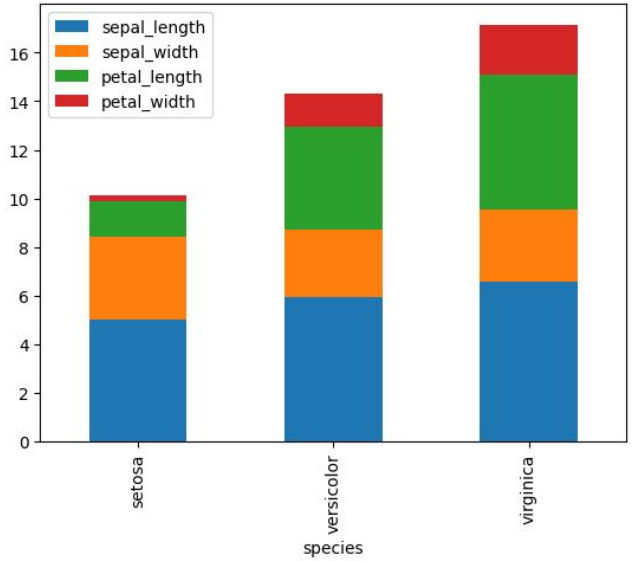
In [3]:
```
grouped_iris = iris.groupby('species').mean().reset_index()
# Stacked = False (default)
grouped_iris.plot.bar(x='species')
```



누적 막대 그래프

- Bor 그래프에서 하나의 bar 가 몇가지 category로 나누어지는 경우
- Pandas.plot.bar() 을 활용
- 여러 범주의 데이터를 쌓은 Stacked Bar 형태로 표현
- Stacked = True

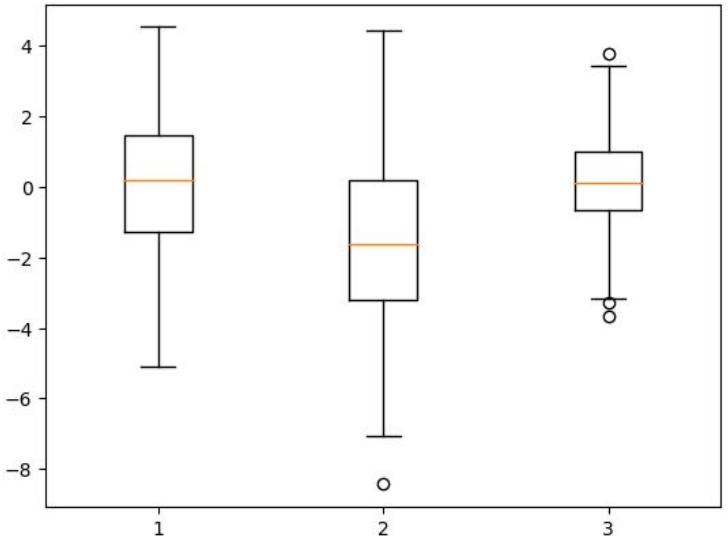
In [3]:
```
grouped_iris = iris.groupby('species').mean().reset_index()
# Stacked = True
grouped_iris.plot.bar(x='species',stacked=True)
```



박스 플롯 (Box Plot)

- 특정 카테고리의 수치 데이터의 범위를 표현할때 주로 사용
- 정보 : 최소/최대값, 사분위수(Q1~Q4), Outlier, Median
- box() 함수 : 각 box들의 값들의 목록을 받아옴

In [3]:
```
np.random.seed(0)

# 데이터 생성
data_a = np.random.normal(0, 2.0, 100)
data_b = np.random.normal(-1.5, 2.5, 200)
data_c = np.random.normal(0.3, 1.3, 300)

#box 플롯
plt.boxplot([data_a, data_b, data_c])
```

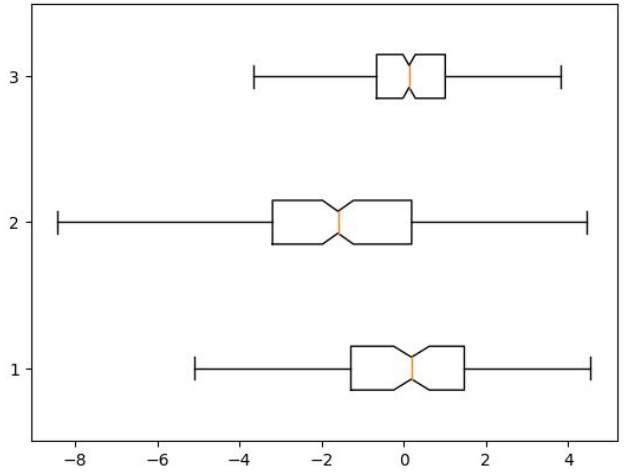


박스 플롯 (Box Plot)

- **Argument**

Notch : True로 설정시 95% 신뢰구간을 출력 (whis=2.5)

vert : False 로 설정시 수평 박스 차트로 변형

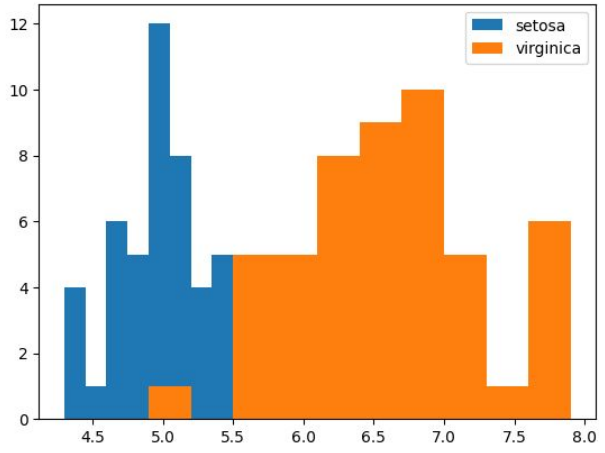


In [3]:
```
np.random.seed(0)

# 데이터 생성
data_a = np.random.normal(0, 2.0, 100)
data_b = np.random.normal(-1.5, 2.5, 200)
data_c = np.random.normal(0.3, 1.3, 300)

#box 플롯
plt.boxplot([data_a, data_b, data_c],notch=True,whis=2.5,vert=False)
```

히스토그램 (Histogram)

- 수치형 데이터의 분포를 Count로 표현할때
- plt.hist(x) : 수치형 데이터 배열을 input으로 받아옴
- Argument
  - bins : histogram 칸마다의 간격
  - label : 여러 histogram을 쌓고, 각 이름을 정할때
  - histtype : ‘bar’, ‘barstacked’, ‘step’, ‘stepfilled’

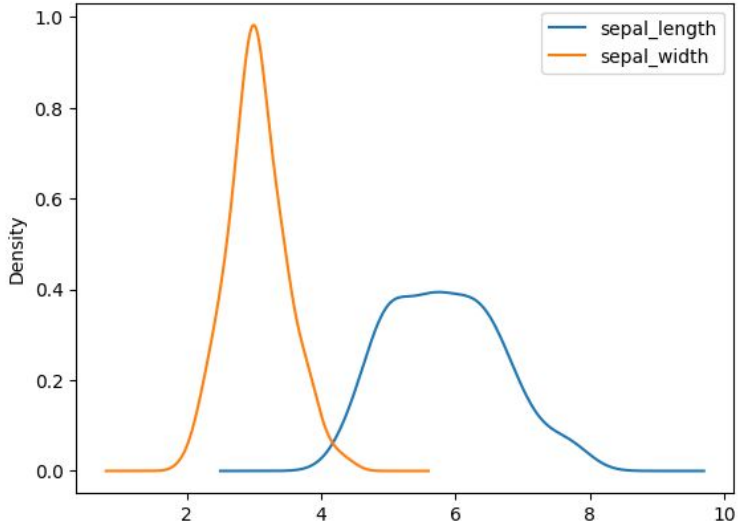
In [3]:
```
# setosa 종의 sepal_length
setosa = iris[iris.species=='setosa'].sepal_length
# virginica 종의 sepal_length
virginica = iris[iris.species=='virginica'].sepal_length
# Setosa 의 히스토그램
plt.hist(setosa, label='setosa')
# Virginica 의 히스토그램
plt.hist(virginica, label='virginica')
plt.legend()
```



커널 밀도 함수

- 수치형 데이터의 분포를 연속 함수로 보여주는 경우
- 커널 밀도의 값을 통해 분포를 나타낸다
- **Pandas.plot.kde() 함수 사용**
- **Argument**
  - **secondary_y : True로 설정시 y축 두개로 분포를 나타냄**

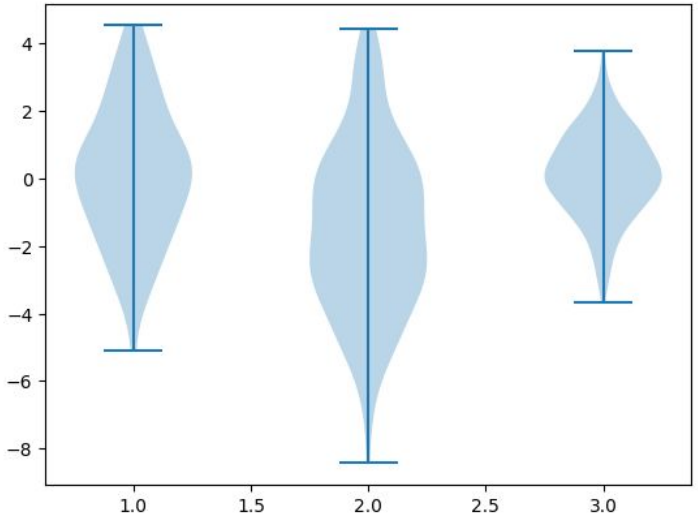
In [3]:
```
# Sepal_width에 대한 kde
iris.sepal_length.plot.kde(label='sepal_length')
# Sepal_width에 대한 kde
iris.sepal_width.plot.kde(label='sepal_width')

plt.legend()
plt.show()
```



바이올린 플롯

- Box 플롯과 유사하지만, 분포에 대한 정보를 더 자세히 나타내는 경우
- violinplot() 함수를 사용 : 인자로 받아오는 정보는 box()와 동일

In [3]:
```
np.random.seed(0)

# 데이터 생성
data_a = np.random.normal(0, 2.0, 100)
data_b = np.random.normal(-1.5, 2.5, 200)
data_c = np.random.normal(0.3, 1.3, 300)

#box 플롯
plt.violinplot([data_a, data_b, data_c])
```



Axes3D

- 기존 pyplot의 2차원 그래프 외에 3차원 그래프가 필요한 경우
- mpl_toolkits.mplot3d 에서 Axes3D 사용
- add_subplot()의 인자 projection = '3d' 로 설정
- 기존 plot 함수에 3개 차원의 데이터 입력

In [3]:

```
# Axes3D 라이브러리 불러오기
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize=(6, 6))
# Subplot 3d로 설정
ax = fig.add_subplot(111, projection='3d')
s_w = iris.sepal_width
s_l = iris.sepal_length
p_w = iris.petal_width

# 3개 차원의 데이터 입력
ax.scatter(s_w,s_l,p_w)
```