Lecture 11. Matplotlib 활용(2)

기초 데이터 분석

Today : Matplotlib 활용 (2)

- Axes3D (2)
- Colormap
- Error Bar
- Confidence Interval
- Contour
- Hex

Axes3D

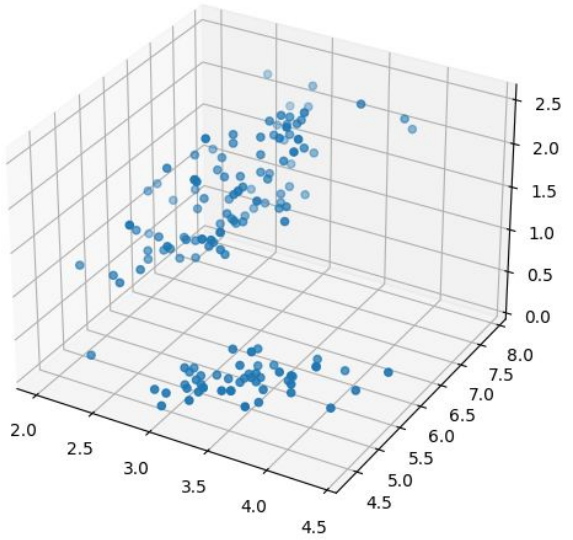
- 기존 pyplot의 2차원 그래프 외에 3차원 그래프가 필요한 경우
- mpl_toolkits.mplot3d 에서 Axes3D 사용
- add_subplot()의 인자 projection = '3d' 로 설정
- 기존 plot 함수에 3개 차원의 데이터 입력

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)
```



Axes3D +@

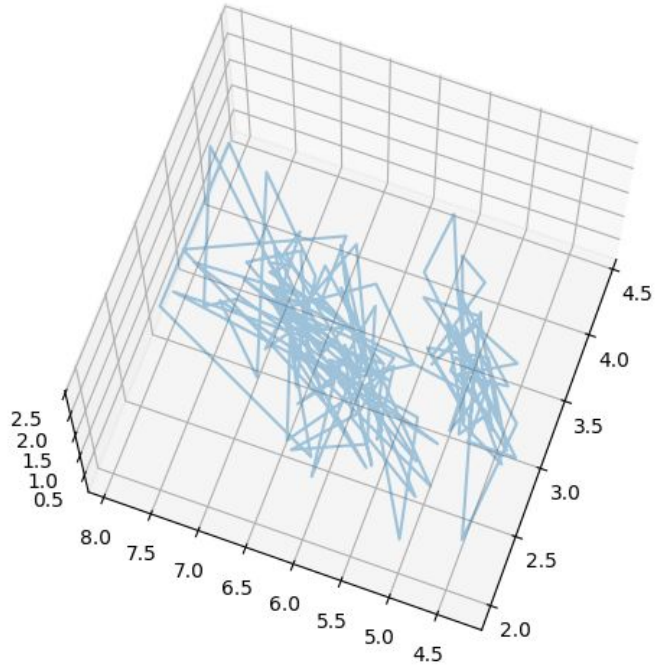
- 시점 각도 조절 : view_init() 함수 사용
- 투명도 조절 : plot 당시 alpha 값을 조정
- scatter 외에도 다른 plot 사용 가능

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.plot(s_w,s_l,p_w, alpha=0.2)
ax.view_init(elev=70.,azim=200)
```

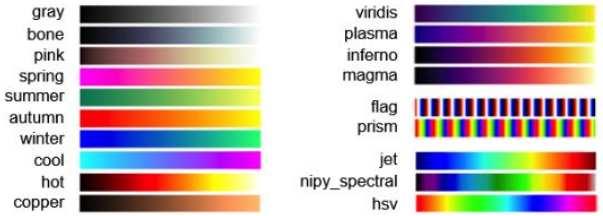


컬러매핑 (Color Map)

- 매번 직접 색상을 지정하는 과정이 번거롭다
- 특정 색상 계열을 선택해 자동으로 색상 지정
- 수치 데이터를 기반으로 **Mapping** 함수가 색상을 **returr**

컬러매핑 (Color Map)

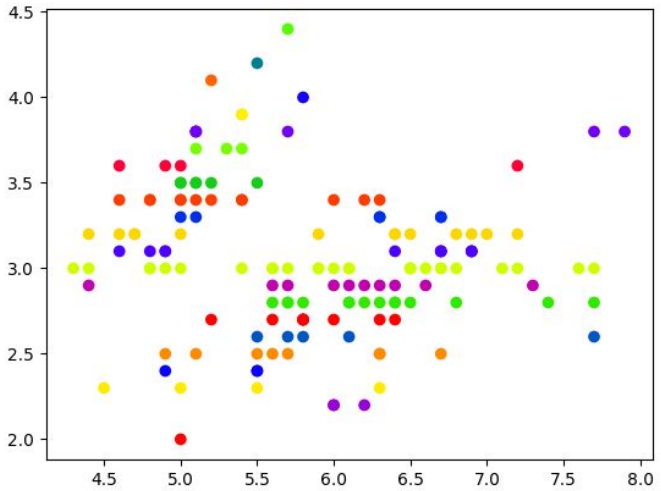
- 사전 정의된 컬러맵을 호출하여 수치들이 컬러맵를 따르도
- plt.컬러맵_이름() 의 형식으로 호출 : ex) plt.spring()
- 컬러맵의 종류는 plt.colormaps() 호출로 확인



In [3]:

```
length = iris.sepal_length
width = iris.sepal_width
# width 기반 color mapping
plt.scatter(length,width,c=width)

plt.prism() # prism 컬러맵 적용
```

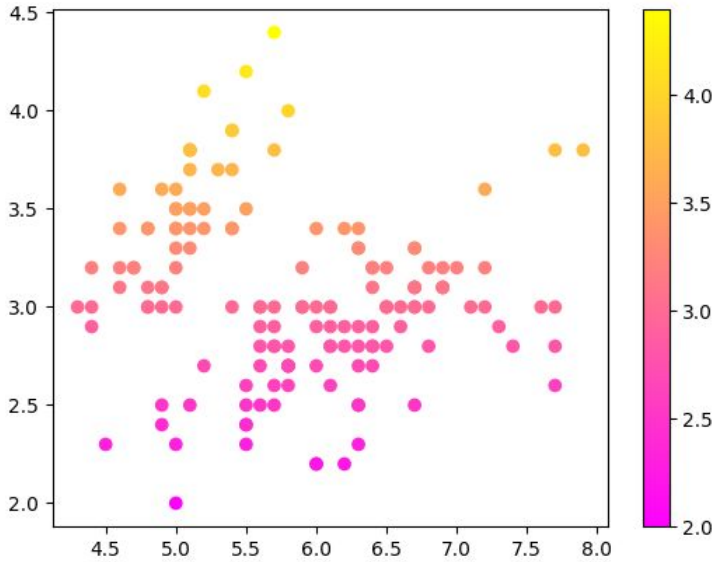


컬러바 (ColorBar)

- 색상 매핑의 range를 표현하는 Colorbar
- plt.colorbar() 를 호출하여 컬러파 추가

In [3]:
```
length = iris.sepal_length
width = iris.sepal_width
# width 기반 color mapping
plt.scatter(length,width,c=width)

plt.spring() # prism 컬러맵 적용
plt.colorbar() # Colorbar 표시
```

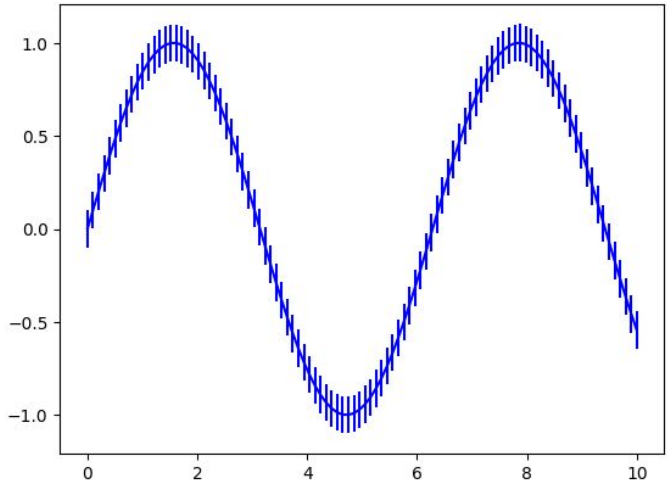


에러바 Error Bar

- 데이터의 편차를 표현하는 **Error Bar**
- 수치형 데이터의 편차를 표현하여 그래프의 신뢰도를 늘릴 수 있다

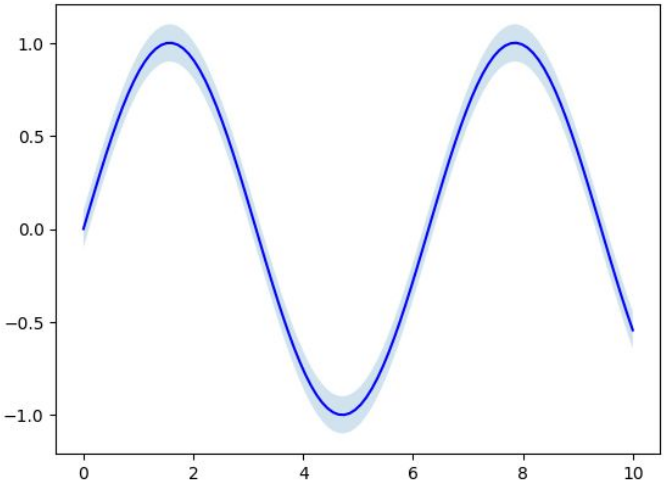
In [3]:
```
x = np.linspace(0, 10, 100)
fig = plt.figure()
yerr = [0.1]*100 # 모두 0.1 크기의 errorbar 생성

# yerr 설정
plt.errorbar(x, np.sin(x), yerr = yerr, color='blue')
plt.show()
```



Confidence Interval

- Error Bar와 유사하게 신뢰구간을 표현하기 위해 사용
- plt.fill_between(x, y_bottom, y_top) 형식으로 사용
- alpha 값을 지정하여 투명도 조절 가능



In [3]:
```
x = np.linspace(0, 10, 100)
fig = plt.figure()
yerr = 0.1 # 모두 0.1 크기의 errorbar 생성
plt.plot(x, np.sin(x), color='blue')

# interval 설정
plt.fill_between(x, np.sin(x)-yerr, np.sin(x)+yerr, alpha=0.2)

plt.show()
```

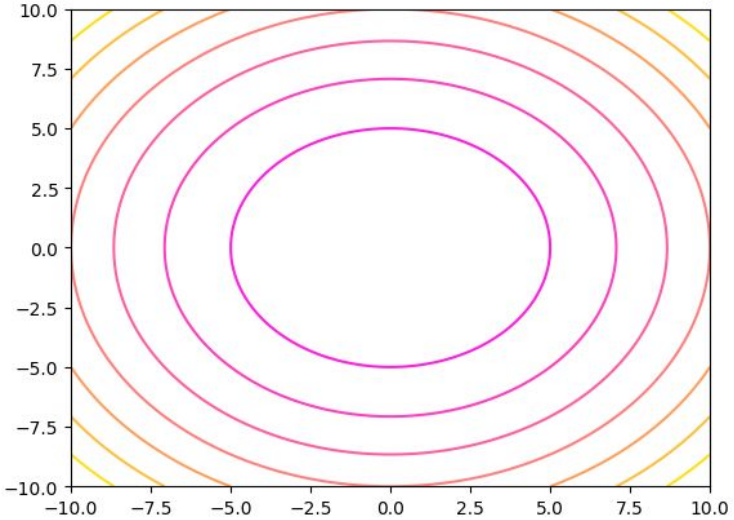
등치선 플롯 (contour)

- 같은 level에 있는 수치형 데이터를 연결한 등치선 플롯
- x,y축의 mesh grid가 필요 : np.meshgrid(x,y)
- plt.contour(xmesh, ymesh, zmesh) 사용
- colorbar와 함께 사용 가능

In [3]:
```
def f(x,y): # z level 를 결정하는 함수
return x**2 + y**2

x = np.linspace(-10,10,100)
y = np.linspace(-10,10,100)
xmesh, ymesh = np.meshgrid(x,y) # x,y mesh 형성
zmesh = f(xmesh,ymesh) # z mesh 형성

plt.contour(xmesh,ymesh,zmesh) # Contour 그리기
plt.colorbar()
```



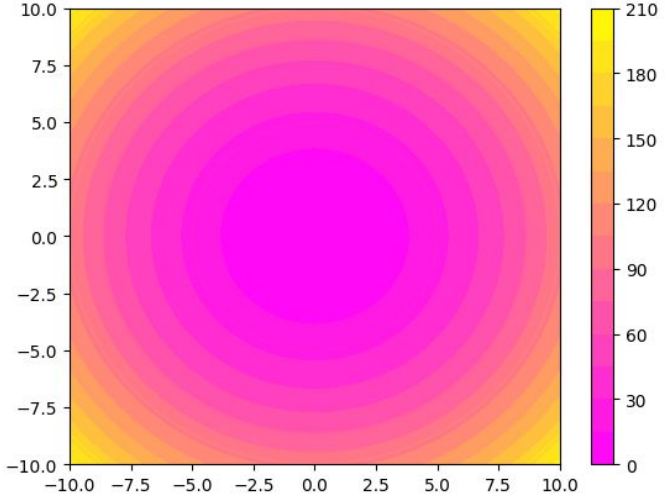
등치선 플롯 (contour)

- contourf() 함수 : 등치선 사이에 값에 따른 색상 추가
- contourf() 에게 colormap 적용하여 색상 할당
- contour() 과 contourf() 동시에 적용 가능

In [3]:
```
def f(x,y): # z level 를 결정하는 함수
return x**2 + y**2

x = np.linspace(-10,10,100)
y = np.linspace(-10,10,100)
xmesh, ymesh = np.meshgrid(x,y) # x,y mesh 형성
zmesh = f(xmesh,ymesh) # z mesh 형성

plt.contour(xmesh,ymesh,zmesh) # Contour 그리기
plt.contourf(xmesh,ymesh,zmesh, levels=15, cmap="spring") # Contourf 그리기
plt.colorbar()
```



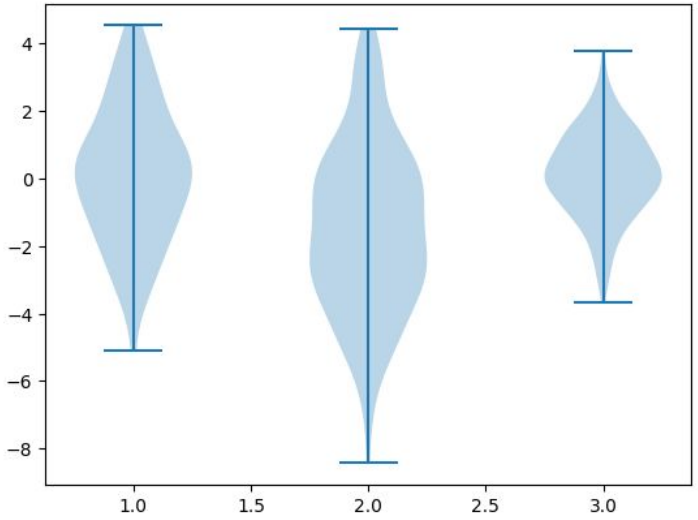
바이올린 플롯

- 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])
```



실전 연습

Seaborn dataset과 지금까지 배운 matplotlib을 활용해 가시화 해봅시다

1. **seaborn의 sns.load_dataset() 을 통해 데이터 로드**
2. **지금까지 배웠던 여러 plot들로 각 column들의 관계를 표현해봅시다**
   a. **line, bar**
   b. **box**
   c. **histogram, kde, hex**
   d. **Axes3d**
   e. **Contour**
   f. **Subplot() …**