Lecture 8. Matplotlib 개요

기초 데이터 분석

Recap : Pandas 를 사용한 가시화

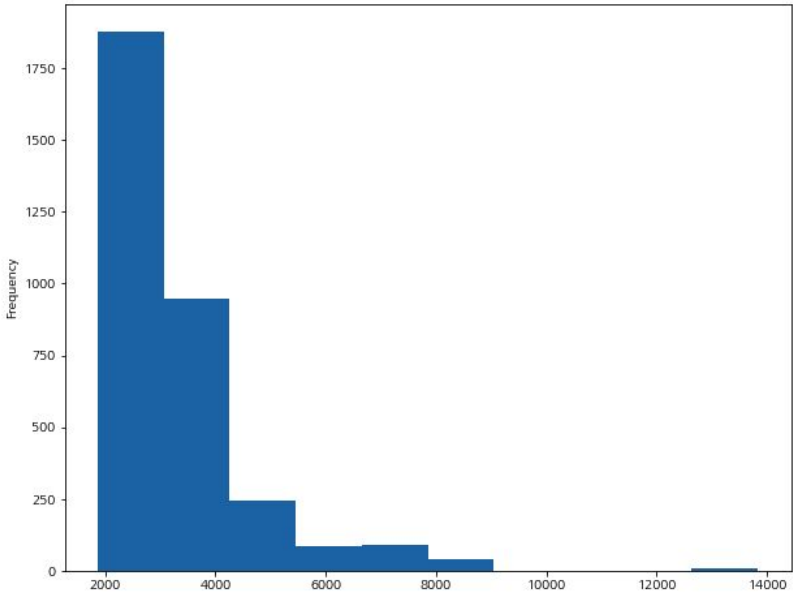
Pandas - Plot()

- DataFrame에 있는 분석 결과를 효과적으로 전달하는 방법
- Pandas는 plot() 함수를 통해 간단하게 시각화 가능
- Line, Bar, Histogram 등 기본적인 차트를 생성 가능 : kind = ‘line’
- 여러개의 그래프를 한번에 plot하는 다중 그래프

Matplotlib vs Pandas.plot()

Pandas.plot() : 사용이 간편하다

- 하지만, 축의 크기를 키우고 싶다면?
- bar들의 간격을 더 줄이고 싶다면?
- 색상을 주황색으로 바꾸고 싶다면?

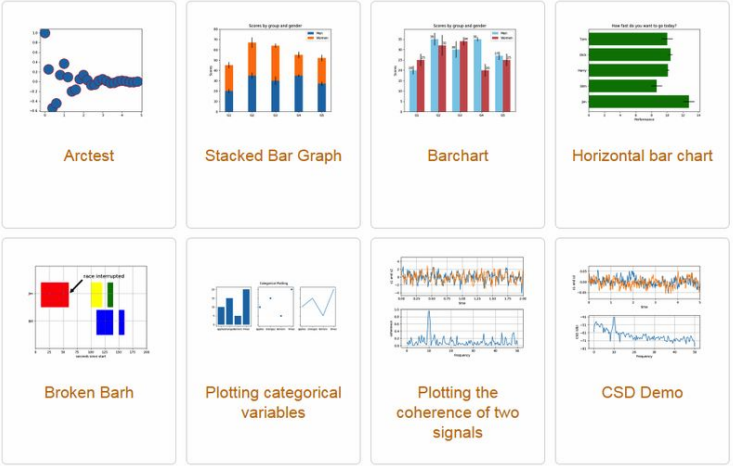


- Matplotlib을 활용하면 Pandas Plot() 차트를 **세부적으로 조정 가능**
- 색상, 글자 크기 등 **가시성이 좋은 자료**를 생성
- 다중 그래프, 축 값 등의 변경 가능

Matplotlib 이란?

Matplotlib

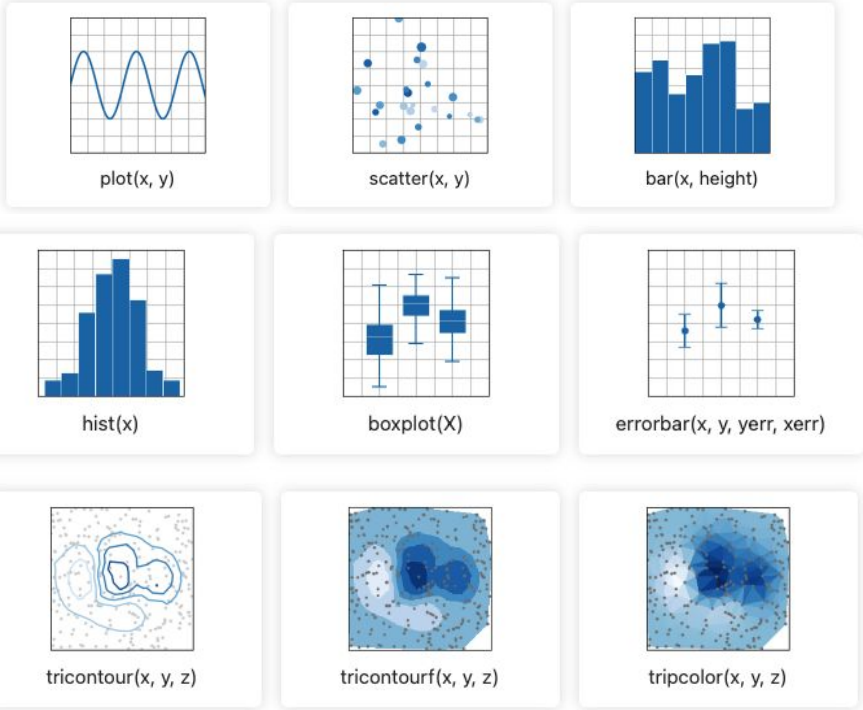
- **데이터 시각화와 2D 그래프 생성**을 위한 파이썬 라이브러리
- Pandas 데이터 프레임을 활용하여 여러가지 차트를 쉽게 생성 가능
- 여러가지 형식의 차트를 지원하며 커스텀이 가능



Matplotlib

Matplotlib로 생성 가능한 Plot

- **Matplotlib은 Matlab에서 생성가능한 여러 타입의 그래프를 지원**
- **기본차트** : 선, 산점도, 바 그래프 등
- **통계차트** : 히스토그램, 박스 플롯, 에러바 등
- 그 외 : 3D, 좌표 데이터, 배열과 벡터 등



Matplotlib 기본 사용법

- **설치 및 환경 설정**
  - pip install matplotlib 을 통해 설치 가능
- **matplotlib 불러오기 & pyplot 서브패키지 사용**
  - import matplotlib.pyplot as plt
- **주피터 노트북 환경에서 노트북에 차트를 저장하는 경우**
  - `%matplotlib inline`
- **기본적인 함수의 호출은 pyplot에 그래프_이름()을 호출하는 식으로 작동**
  - 예시 1) plt.plot() : default 는 line
  - 예시 2) plt.scatter() : 산점도
- **그래프 함수 호출 이후, 최종 적으로 차트를 출력하기 위해 plt.show() 사용**

Matplotlib 기본 사용법

- **설치 및 환경 설정**
  - pip install matplotlib 을 통해 설치 가능
- **matplotlib 불러오기 & pyplot 서브패키지 사용**
  - import matplotlib.pyplot as plt
- **주피터 노트북 환경에서 노트북에 차트를 저장하는 경우**
  - `%matplotlib inline`
- **기본적인 함수의 호출은 pyplot에 그래프_이름()을 호출하는 식으로 작동**
  - 예시 1) plt.plot() : default 는 line
  - 예시 2) plt.scatter() : 산점도

Matplotlib 기본 사용법

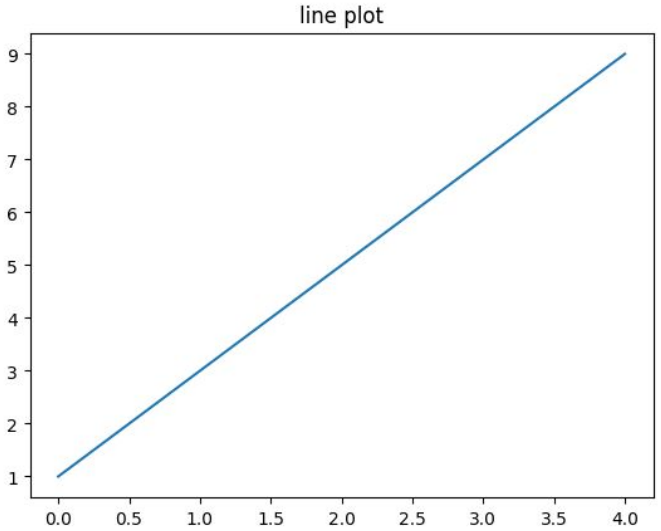
- 그래프 함수 호출 이후, 최종 적으로 차트를 출력하기 위해 **plt.show()** 사용
  - plt.plot([1,4,9,16,25]) # Line 차트 생성을 위해 array 넣기
  - plt.show() # 최종 Chart를 출력
- **figure** 단위로 관리 가능, 호출된 플롯 함수는 **figure**에 차트를 생성
  - fig = plt.figure()
- 최종 **figure**는 **savefig()** 함수를 통해 파일로 저장 가능
  - fig.savefig('figure.png') # 'figure.png' 파일로 저장

In [3]:
```
fig = plt.figure() # pyplot의 figure 객체 생성
plt.plot([1,2,3,4,5]) # Line 플롯으로 플롯 생성
plt.show() # 생성된 플롯 출력
plt.savefig('figure.png') # 생성된 플롯 파일로 저장
```

Line Plot

- 데이터들을 선으로 연결한 **Line Plot**
- **Pandas** 내장 시각화와 유사하게 **plot()** 함수 호출을 통해 사용
  - 함수 호출 : plt.plot([1,2,3,4])
  - 차트 출력 : plt.show()

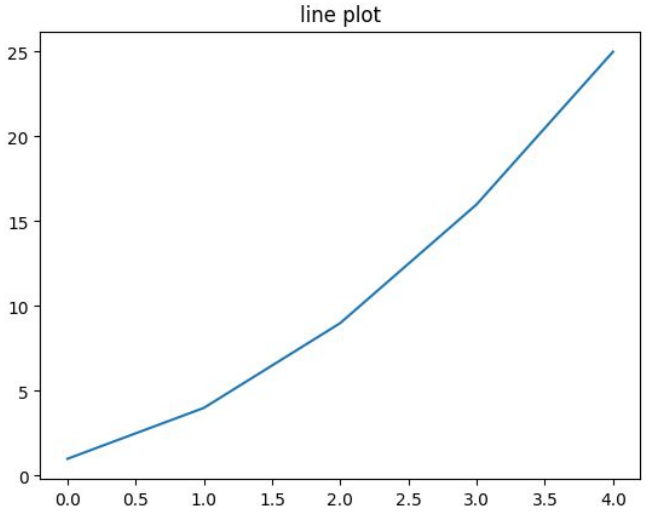
In [3]:
```
plt.title('line plot') # 차트 제목 설정
plt.plot([1,3,5,7,9]) # Line 차트 생성을 위해 array 넣기
plt.show() # 최종 Chart를 출력
```



Line Plot

- 데이터들을 선으로 연결한 **Line Plot**
- **Pandas** 내장 시각화와 유사하게 **plot()** 함수 호출을 통해 사용
  - 함수 호출 : plt.plot([1,2,3,4])
  - 차트 출력 : plt.show()

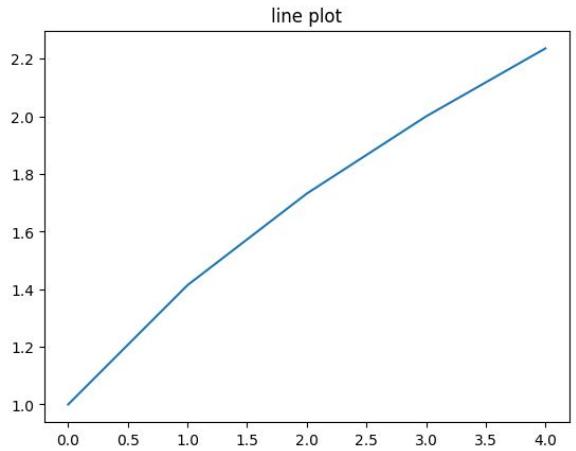
In [3]:
```
plt.title('line plot') # 차트 제목 설정
plt.plot([1,4,9,16,25]) # Line 차트 생성을 위해 array 넣기
plt.show() # 최종 Chart를 출력
```



Line Plot

- 데이터들을 선으로 연결한 **Line Plot**
- **Pandas** 내장 시각화와 유사하게 **plot()** 함수 호출을 통해 사용
  - 함수 호출 : plt.plot([1,2,3,4])
  - 차트 출력 : plt.show()

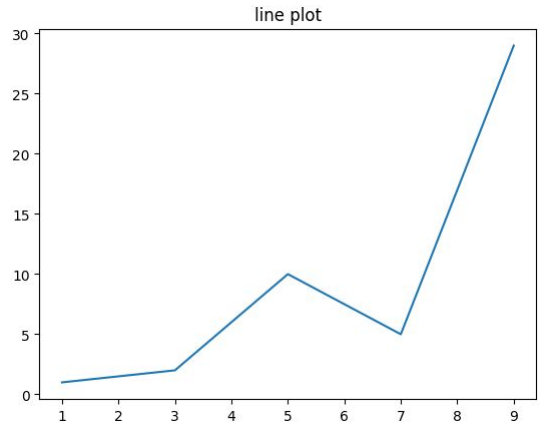
In [3]:
```
plt.title('line plot') # 차트 제목 설정
plt.plot(np.array([1,1.414, 1.732, 2, 2.236])) # Line 차트 생성을 위해 numpy array 넣기
plt.show() # 최종 Chart를 출력
```



Line Plot

- **plot() 함수 내부에는 x값, 혹은 x,y 값을 넣어줄 수 있다**
- 하나의 배열만 입력 : **x는 1~N까지 자동으로 지정해서 플롯 생성**
- 2개의 동일한 길이의 배열 입력 : **각 x에 따른 y 값들을 선으로 잇는 플롯 생성**

In [3]:
```
plt.title('line plot') # 차트 제목 설정
plt.plot([1,3,5,7,9],[1,2,10,5,29]) # Line 차트 생성을 위해 array 넣기
plt.show() # 최종 Chart를 출력
```

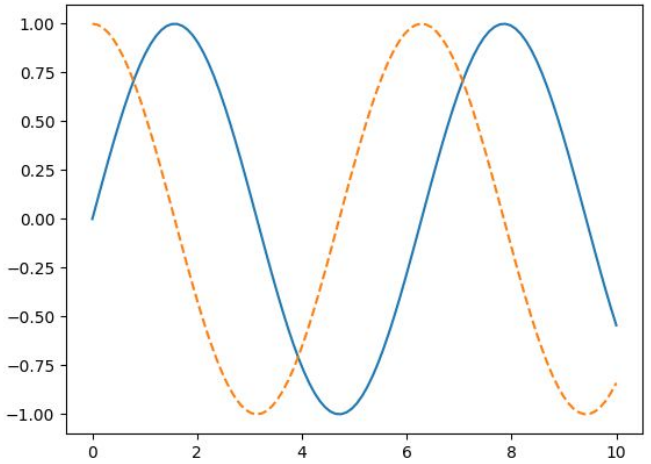


Line Plot

- **plot() 함수 내부에는 x값, 혹은 x,y 값을 넣어줄 수 있다**
- 하나의 배열만 입력 : **x는 1~N까지 자동으로 지정해서 플롯 생성**
- 2개의 동일한 길이의 배열 입력 : **각 x에 따른 y 값들을 선으로 잇는 플롯 생성**

In [3]:
```
import numpy as np
x = np.linspace(0, 10, 100)

fig = plt.figure()
plt.plot(x, np.sin(x), '-')
plt.plot(x, np.cos(x), '--');
plt.show()
```



Line Plot

그 외의 함수 인자들

- c / color : Line의 색상을 설정한다.
  - 여러 Plot이 한 Figure에 있을 경우 자동으로 다른 색상으로 지정됨
  - 하지만 임의로 색상을 지정하는 경우 color 인자 사용

In [3]:
```
x = np.linspace(0, 10, 100)
fig = plt.figure()
plt.plot(x, np.sin(x), color='blue') # 파란색 라인 생성
plt.plot(x, np.cos(x), color='m') # 마젠타 색상 라인 생성
plt.show()
```

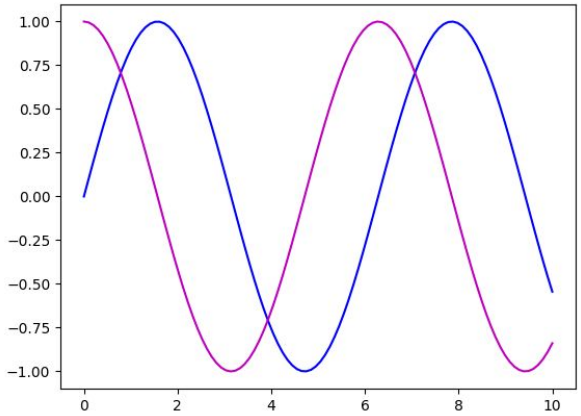
character	color
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

Line Plot

그 외의 함수 인자들

- c / color : Line의 색상을 설정한다.
  - 여러 Plot이 한 Figure에 있을 경우 자동으로 다른 색상으로 지정됨
  - 하지만 임의로 색상을 지정하는 경우 color 인자 사용

In [3]:
```
x = np.linspace(0, 10, 100)
fig = plt.figure()
plt.plot(x, np.sin(x), color='blue') # 파란색 라인 생성
plt.plot(x, np.cos(x), color='m') # 마젠타 색상 라인 생성
plt.show()
```

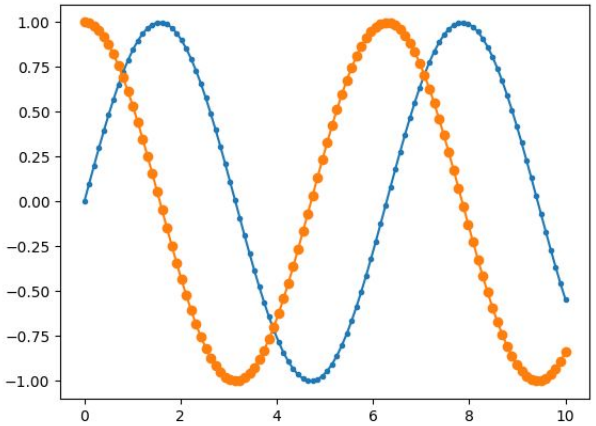


Line Plot

그 외의 함수 인자들

- **marker**
- 각 data point 의 표시를 커스텀 한다.
- Options : . , o v x + 등등

In [3]:
```
x = np.linspace(0, 10, 100)
fig = plt.figure()
plt.plot(x, np.sin(x), marker='.') # 점 마커 사용
plt.plot(x, np.cos(x), marker='o') # 원형 마커 사용
plt.show()
```

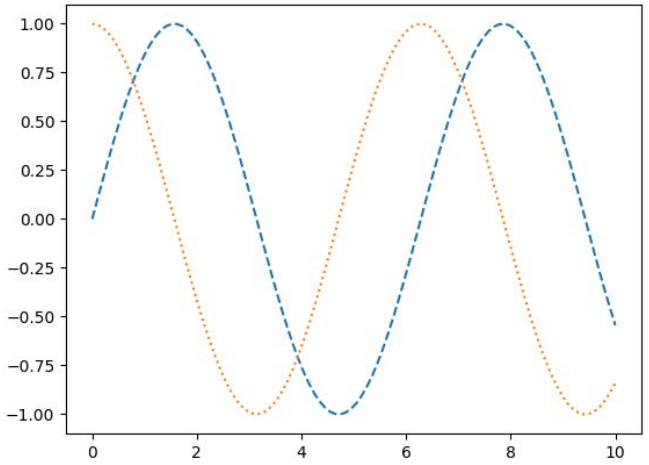


Line Plot

그 외의 함수 인자들

- **linestyle : 선의 모양을 설정**
- **options : '-'(실선, 기본값)**, '--'(대시선), '-.'(대시&실선), ':'(점선)

In [3]:
```
x = np.linspace(0, 10, 100)
fig = plt.figure()
plt.plot(x, np.sin(x), linestyle='--') # 대시선 사용
plt.plot(x, np.cos(x), linestyle=':') # 점선 사용
plt.show()
```

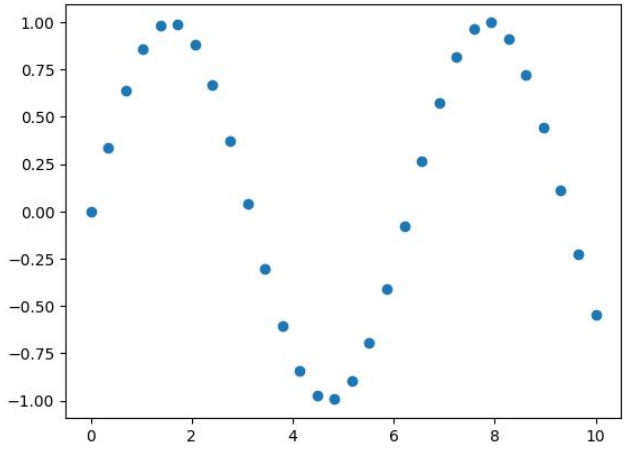


Scatter Plot

Scatter Plot (산점도) : 데이터 포인트를 하나의 점으로 나타내는 플롯

- plt.scatter() 함수를 호출하여 사용
- 입력으로는 x,y 값을 받아오며, x,y는 동일한 길이의 배열
- 각 x와 y 값에 대응되는 점을 플롯한다
- 기존 Plot() 함수로 동일한 결과물 생성도 가능

In [3]:
```
x = np.linspace(0, 10, 30)
y = np.sin(x)
plt.scatter(x, y); # Scatter Plot
```



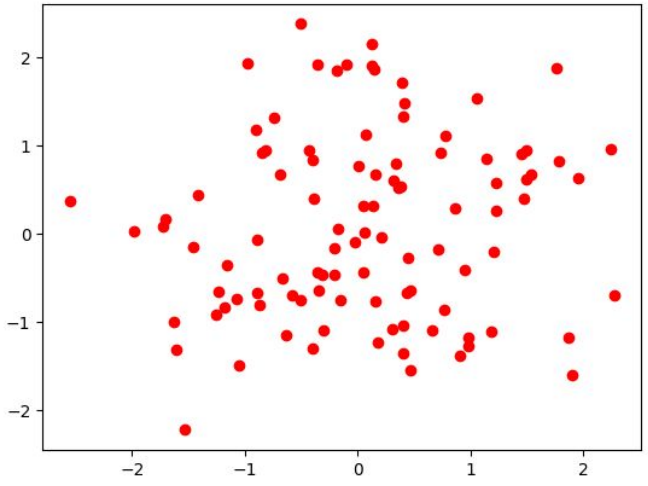
Scatter Plot

Scatter Plot (산점도) : 데이터 포인트를 하나의 점으로 나타내는 플롯

함수 인자

- **c** : 각 점들의 색상 지정
- 하나의 색상으로 지정하거나
- x,y 와 같은 길이의 배열 입력시, 각 점의 색상 지정

In [3]:
```
rng = np.random.RandomState(0)
x = rng.randn(100) # 랜덤한 x값 100개 생성
y = rng.randn(100) # 랜덤한 y값 100개 생성
plt.scatter(x,y,c = 'red')
```



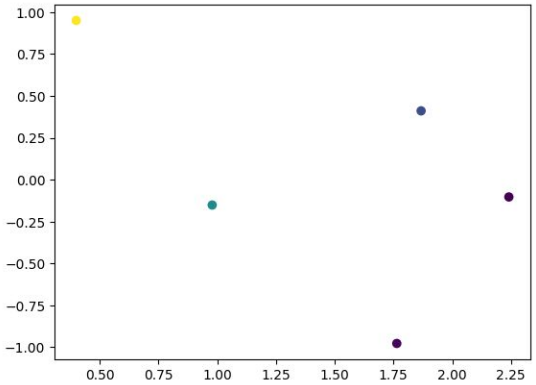
Scatter Plot

Scatter Plot (산점도) : 데이터 포인트를 하나의 점으로 나타내는 플롯

함수 인자

- **c** : 각 점들의 색상 지정
- 하나의 색상으로 지정하거나
- x,y 와 같은 길이의 배열 입력시, 각 점의 색상 지정

In [3]:
```
rng = np.random.RandomState(0)
x = rng.randn(5) # 랜덤한 x값 5개 생성
y = rng.randn(5) # 랜덤한 y값 5개 생성
colors = rng.randn(5) # 랜덤한 색상값 5개 생성
plt.scatter(x,y, c = colors)
```



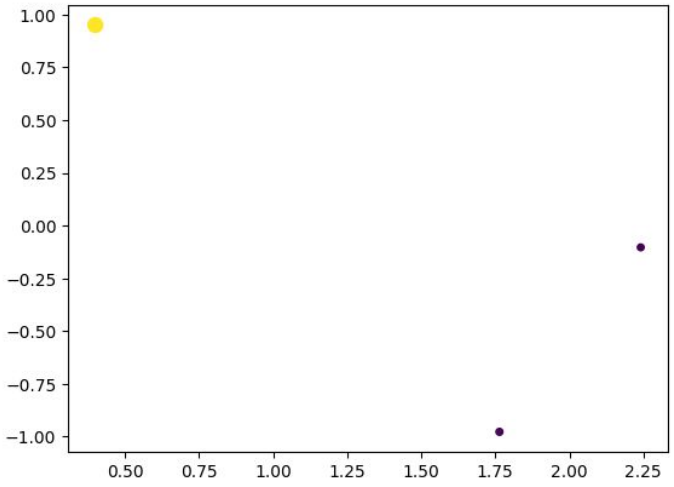
Scatter Plot

Scatter Plot (산점도) : 데이터 포인트를 하나의 점으로 나타내는 플롯

함수 인자

- **s** : 각 점들의 반지름 설정
- 하나의 크기로 지정하거나
- x,y 와 같은 길이의 배열 입력시, 각 점의 크기 지정

In [3]:
```
rng = np.random.RandomState(0)
x = rng.randn(5) # 랜덤한 x값 5개 생성
y = rng.randn(5) # 랜덤한 y값 5개 생성
colors = rng.randn(5) # 랜덤한 색상값 5개 생성
sizes = rng.randn(5)*50 # 랜덤한 크기값 5개 생성
plt.scatter(x,y, s=sizes, c=colors)
```

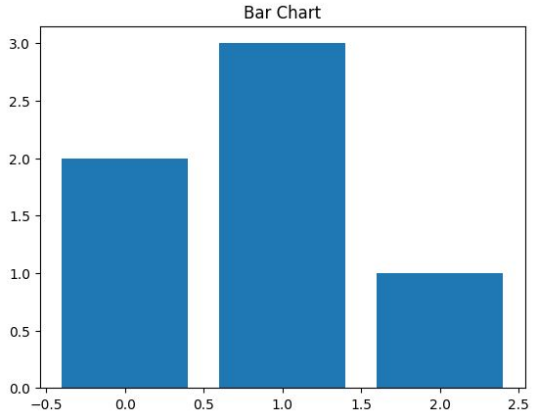


Bar Plot

Bar Plot : 데이터 포인트의 수치를 막대로 표현

- **plt.bar()** 함수를 호출하여 사용
- 입력으로는 **x, height** 값을 받아오며, x,height는 동일한 길이의 배열
- 각 x 값에 대응되는 높이의 막대를 표현
- 추후 배울 tick 옵션을 활용해 문자열을 x 축으로 사용 가능

In [3]:
```
y = [2, 3, 1]
x = np.arange(len(y))
plt.title("Bar Chart")
plt.bar(x, y)
```



Bar Plot

Bar Plot : 데이터 포인트의 수치를 막대로 표현

Options

- width : 막대의 너비를 설정
- bottom : 막대의 시작점을 지정
- color : 막대 색상 지정 가능

In [3]:
```
rng = np.random.RandomState(0)
x = [1,2,3,4] # 랜덤한 x값 100개 생성
y = rng.randn(4) # 랜덤한 y값 100개 생성
w = rng.randn(4) # 랜덤한 width값 100개 생성
plt.bar(x,y,width=w,color='orange')
```