Out[2]:

Click here to toggle on/off the raw code.

🐍 [One Point Tutorial] Visualization I - `matplotlib`

Python을 활용한 데이터 시각화

☝ 시각화 튜토리얼을 들어가기 전,

우리는 데이터 시각화를 왜 해야할까? ¶

1. 데이터를 한 눈에 살펴 볼 수 있다.

2. 데이터 지식이 부족한 사람도 시각화를 통하여 데이터를 쉽게 이해할 수 있다.

3. 데이터에 대한 인사이트 공유가 가능해지기 때문에 데이터 기반의 효율적인 의사 소통이 가능해진다.

4. 데이터 시각화가 쓰일 수 있는 분야는 무궁무진하다.

5. EDA 등 전처리 단계에서 이슈를 빠르게 파악할 수 있다.

6. AUROC 등 머신러닝의 성과 평가에 유용하게 활용된다.

7. ...

1. *matplotlib* 을 배워 보자 !

- **matplotlib 이란?**
  파이썬의 대표적인 데이터 시각화 라이브러리
  핵심적인 시각화 기법을 대다수 적용 가능

학습 목표

`matplotlib` 의 핵심적인 시각화 기법을 이해하고 활용함

목차

1. Line Plot
2. Scatter Plot
3. Bar Plot
4. Histogram
5. Piechart

Import module (모듈 설치 후 불러오기)

matplotlib 모듈이 설치 안 되어 있다면, 설치부터 하기!

```
Current Working Directory is changed.
```

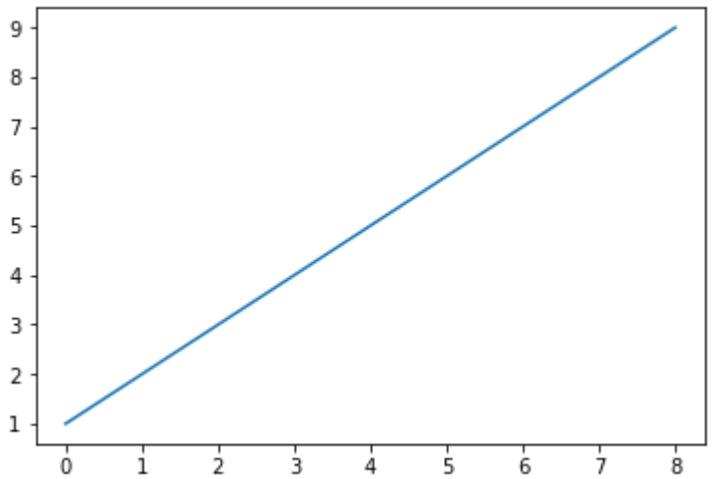
1.1. 가장 간단한 그래프 , Line Plot

가장 간단한 플롯은 선을 그리는 라인 플롯(line plot)이다.
라인 플롯은 데이터가 시간, 순서 등에 따라 어떻게 변화하는지 보여주기 위해 사용한다.

> **사용 방법 :**
> plt. `plot` (x데이터 , y데이터)

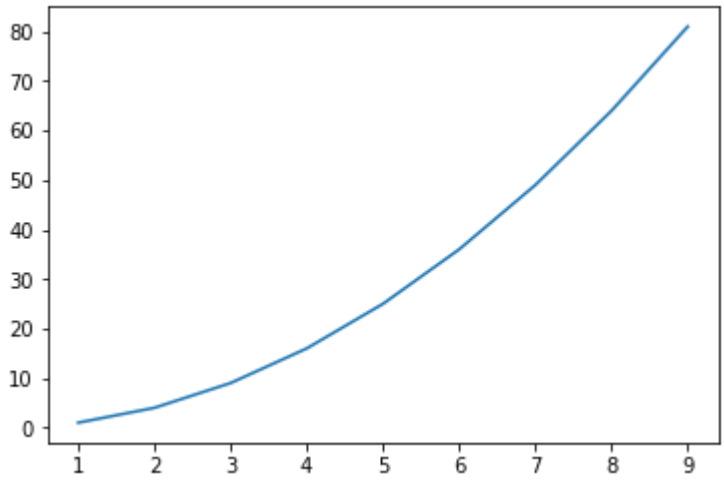
1.1.1. 첫번째 그래프 : 주어진 데이터가 하나일 때 (plot 함수의 데이터 인자가 1개일 때)

이런 경우, x축은 0, 1, 2, 3, ... 으로 자동 지정되고 넣은 데이터는 각 x축에 대응하는 y값이 된다.
확인해보자!



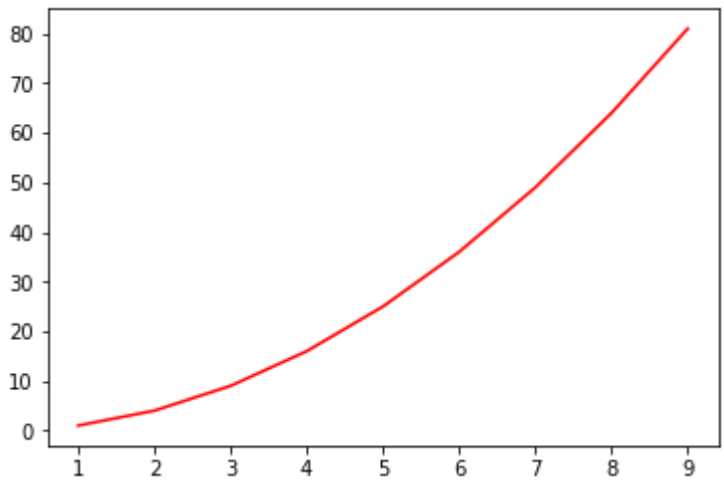
(0, 1), (1, 2), ... , (8, 9)를 지나는 것을 확인할 수 있다.

1.1.2. 두번째 그래프 : x, y축 데이터들이 모두 주어졌을 때



x,y 데이터가 (x, y) 좌표쌍으로 표현되어 그래프에 나타나는 것을 확인할 수 있다.

1.1.3. 세번째 그래프 : 라인의 컬러 바꾸기



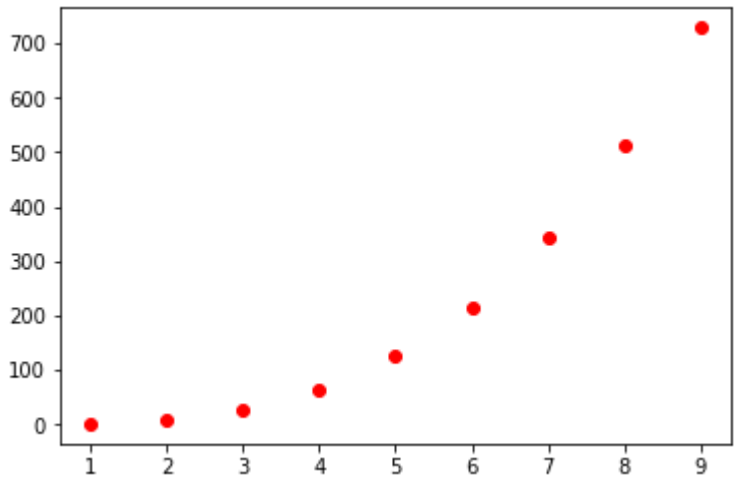
코드에 'r'을 추가한 것만으로 그래프가 빨간색으로 그려진 것을 확인할 수 있다.
'r'은 red의 약자이고, 색을 지정하려면 plot에서 인자를 하나 추가해주면 된다.

색상표

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

1.1.4. 네번째 그래프 : 라인 마커 변경하기

다양한 선의 종류를 선택할 수 있다. default 값은 직선이고, 점으로 표현하는 마커나 점선 등을 원한다면 선택할 수 있다.
선의 종류는 plot에서 인자로 지정하면 되고, 색을 같이 지정하려면 같은 인자에 함께 문자형으로 색을 지정하면 된다.
예시를 살펴보자! 동그란 마커 'o'를 붉은색 'r'로 표현하기 때문에, 세번째 인자를 'or'로 전달하였다.

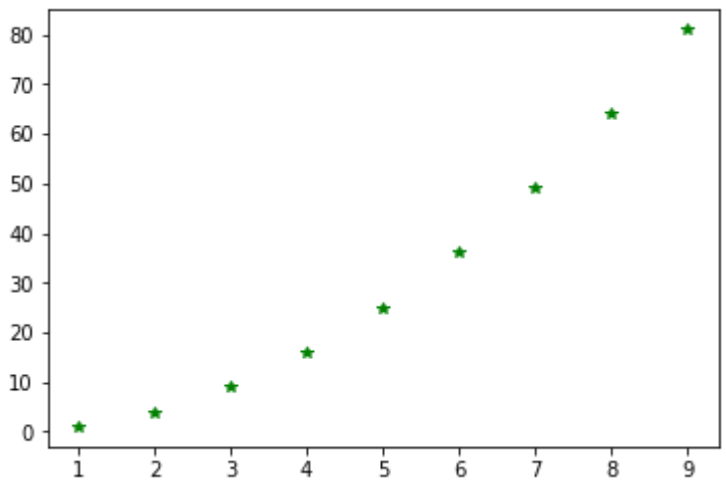


위의 그래프들은 line plot이었는데 이번에는 선이 아닌 점으로 그려진 것을 확인할 수 있다.

마커 종류

문법	종류
'-'	solid line style
'--'	dashed line style
'-.'	dash-dot line style
':'	dotted line style
'.'	point marker
','	pixel marker
'o'	circle marker
'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
'+'	plus marker
'x'	X marker
'D'	diamond marker

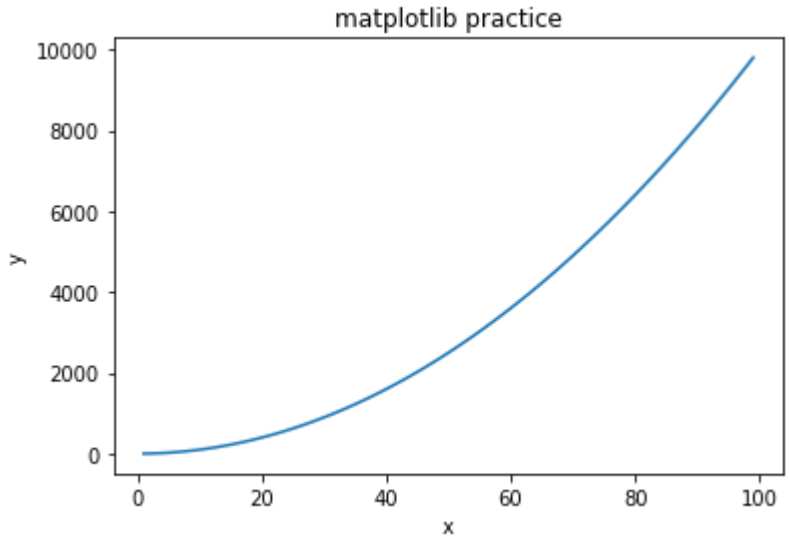
한 번 더 복습해보자.
위 plot의 마커를 초록색 별로 변환해보라.



1.1.5. 다섯번째 그래프 : 축의 이름 / 그래프 제목 달기

사용 방법:

1. x축 이름 : plt.xlabel(x축 이름)
2. y축 이름 : plt.ylabel(y축 이름)
3. 그래프 제목 : plt.title(그래프 제목)



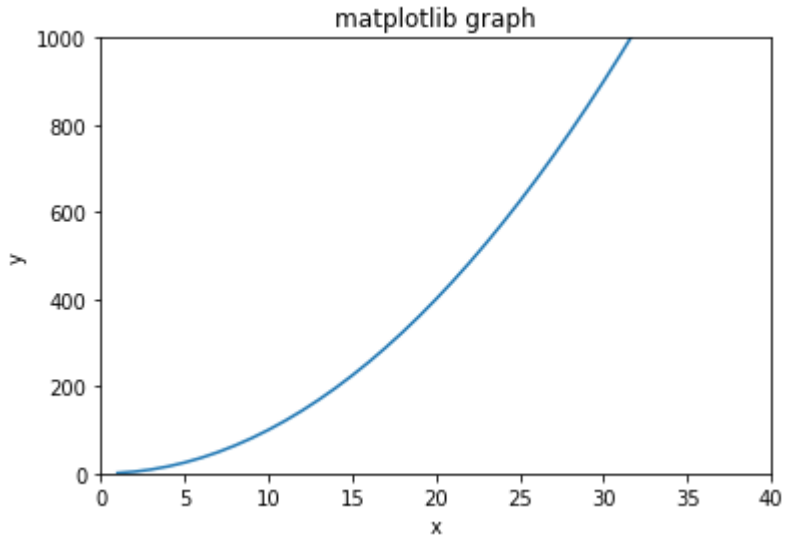
x축 이름/ y축 이름/ 그래프 제목이 생긴 것을 확인할 수 있다 !

1.1.6. 여섯번째 그래프: 구간 설정

x축과 y축의 구간을 설정할 수 있다.

사용 방법:

1. x축 구간 설정 : plt.xlim(시작 범위, 끝 범위)
2. y축 구간 설정 : plt.ylim(시작 범위, 끝 범위)

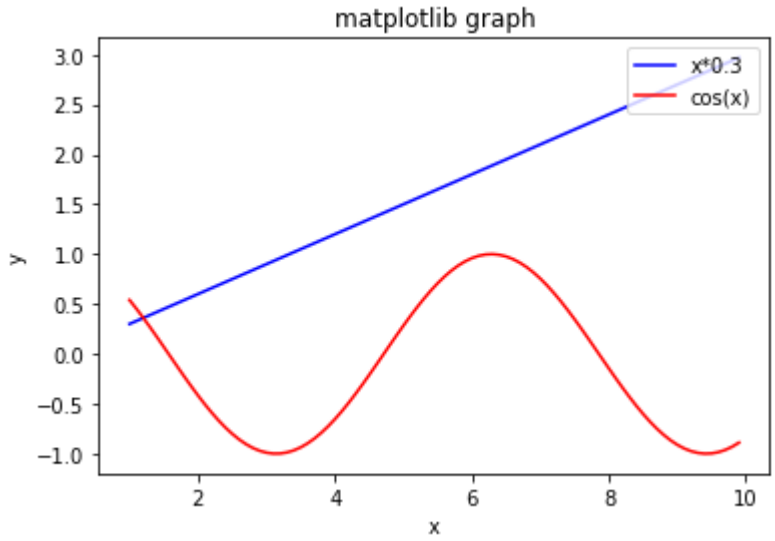


x축은 0에서 40의 범위만, y축은 0에서 1000의 범위만 나타내고 있는 것을 확인할 수 있다.

1.1.7. 일곱번째 그래프: 범주 달기 (legend)

여러 개의 그래프를 같이 그리는 경우가 존재하고, 이 경우 각 그래프를 구분하기 위해 그래프마다 라벨을 부여해 출력
이를 범주(legend)라고 한다.

> **사용 방법:**
> plt.legend(loc = 원하는 위치) #위치 지정해주어야 함



1.1.8. 여덟번째 그래프: 어노테이션 (화살표로 가리키기)

그래프에 화살표를 그린후, 그 화살표에 문자열을 출력하는 기능

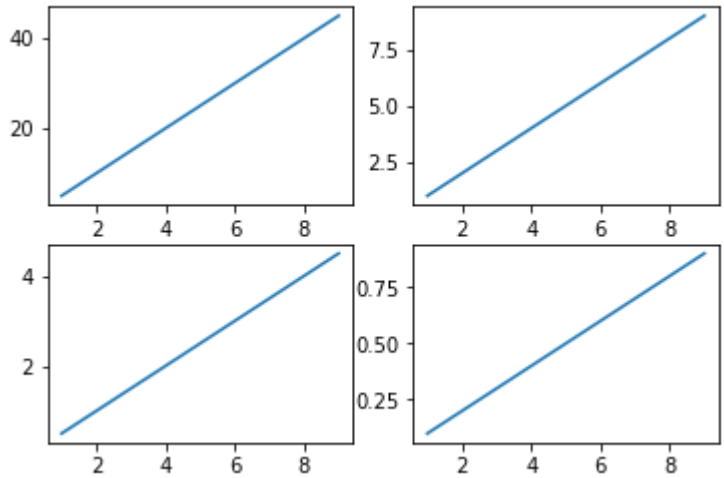
> **사용 방법:**
> plt.annotate(원하는 문자열 , xy(화살표 꼭지 xy좌표), xytext(문자열 xy 좌표), arrowprops(화살표 속성 ex.color))

(1.5, 7.5)의 위치를 가리키는 화살표가 생긴 것을 확인할 수 있다.

1.1.9. 아홉번째 그래프: 서브 플롯 그리기

여러 개의 그래프를 그리고 싶을때 서브플롯을 사용

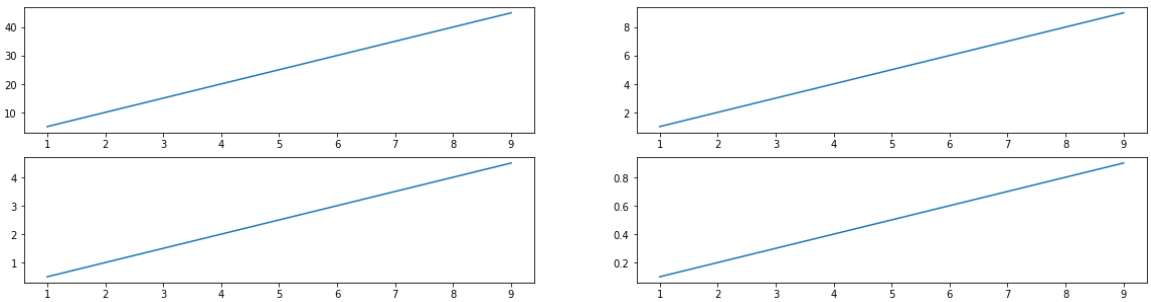
> **사용 방법:**
> plt.subplot(`nrow` , `ncol` , `pos`)



그래프가 그려지는 순서는 행방향으로 진행된다는 것을 알 수 있다.

1.1.10. 열번째 그래프: 그래프 사이즈

> **사용 방법:**
> plt.figure(figsize=(가로 , 세로))



figsize를 어떤 숫자로 지정하느냐에 따라 그래프의 크기가 달라진다.

1.1.11. 열한번째는 팁: plt 기타 스타일 참고

스타일 문자열	약자	의미
color	c	선 색깔
linewidth	lw	선 굵기
linestyle	ls	선 스타일
marker		마커 종류
markersize	ms	마커 크기
markeredgecolor	mec	마커 선 색깔
markeredgewidth	mew	마커 선 굵기
markerfacecolor	mfc	마커 내부 색깔

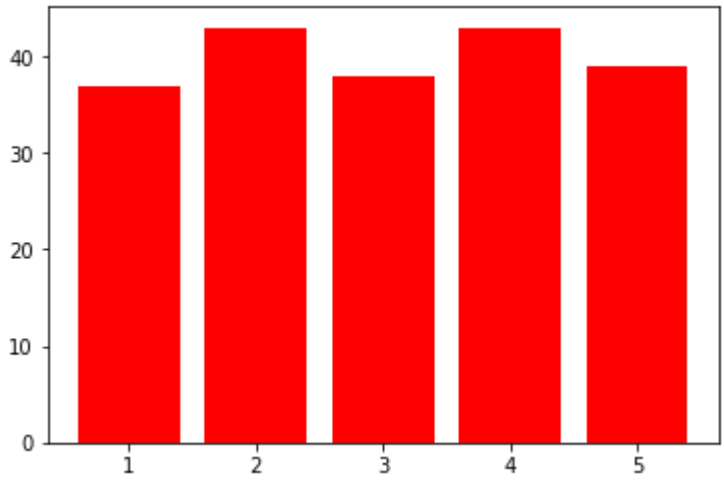
1.2. Bar Plot

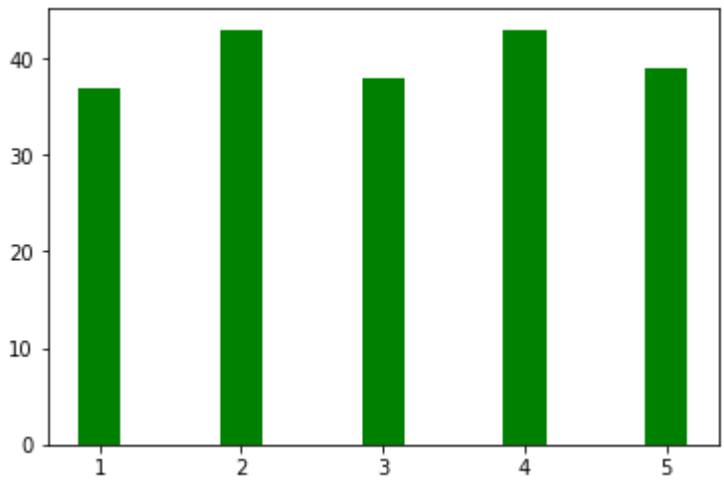
라인 플롯은 ***연속적인 x***에 따른 y의 변화를 살펴보는 데에 효과적이라면, 막대 차트는 ***연속적이지 않은 x***에 따른 y의 변화를 살펴보는 데에 효과적이다.

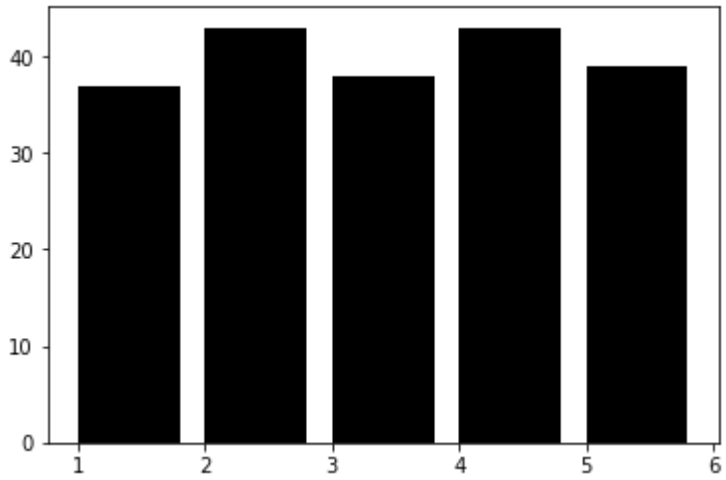
사용 방법:

1. plt.bar(x , height , width , align)
   width: 막대 두께 조절
   align: 'center'와 'edge' (기본값 'center')
   x축 label 막대의 중간 위치(center) 또는 왼쪽 가장자리 위치(edge) 설정
2. 데이터 .plot(kind = 'bar')

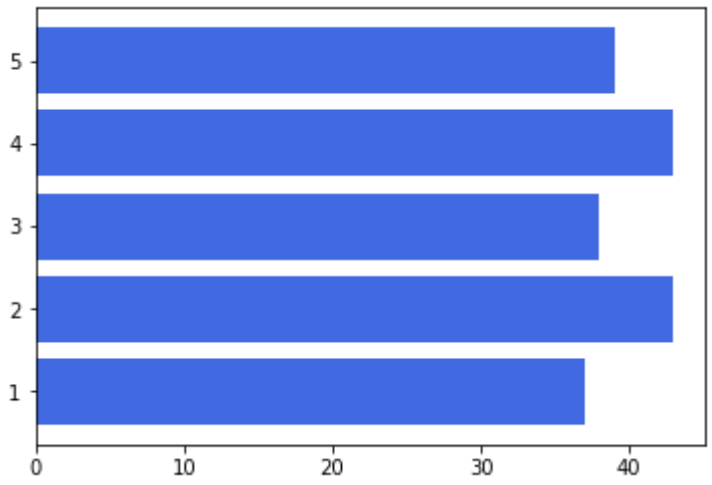
1.2.1. 첫번째 그래프: 첫번째 방법으로 막대 차트 그리기


40
30
20
10
0
1
2
3
4
5

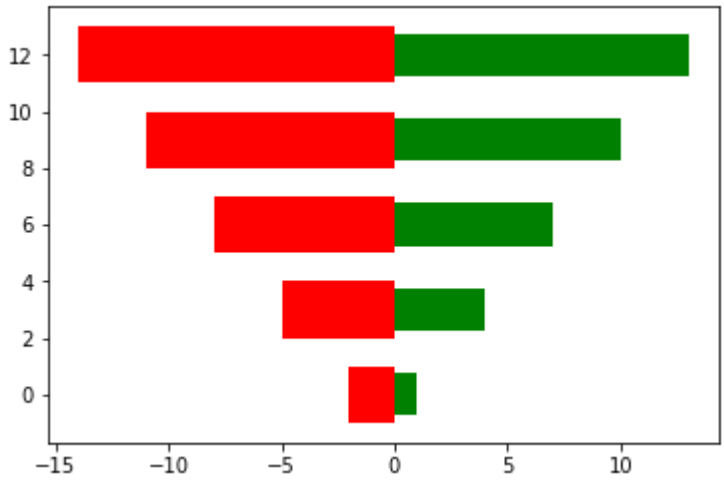

40
30
20
10
0
1
2
3
4
5



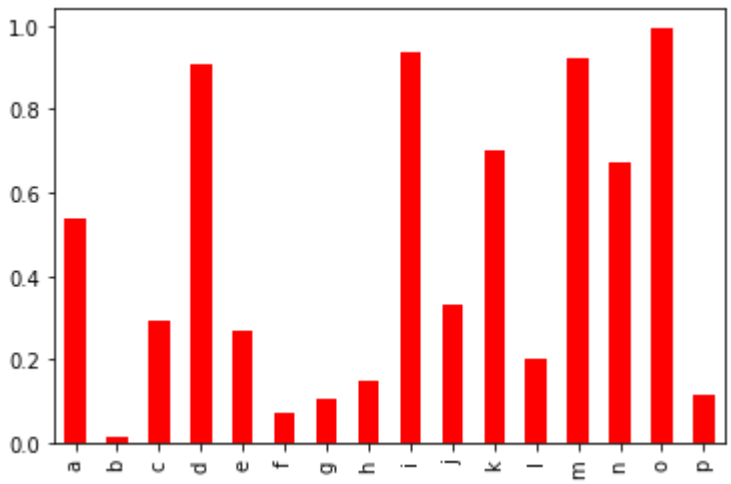
1.2.1. 두번째 그래프: 막대 방향을 수평으로 바꾸기



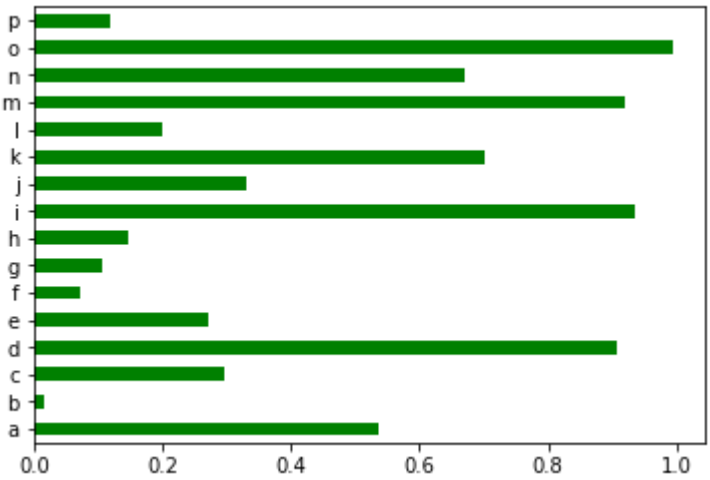
1.2.1. 세번째 그래프: 양쪽으로 막대차트 그리기



1.2.1. 네번째 그래프: 두번째 방법으로 막대 차트 그리기



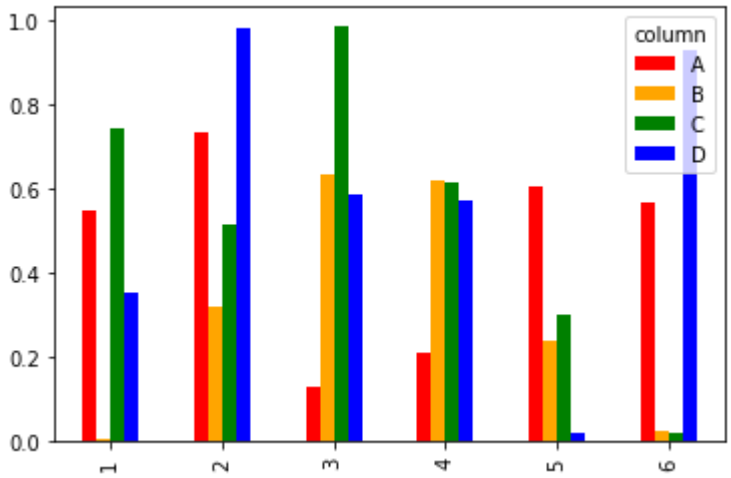
1.2.1. 다섯번째 그래프: 막대 방향 수평으로 바꾸기



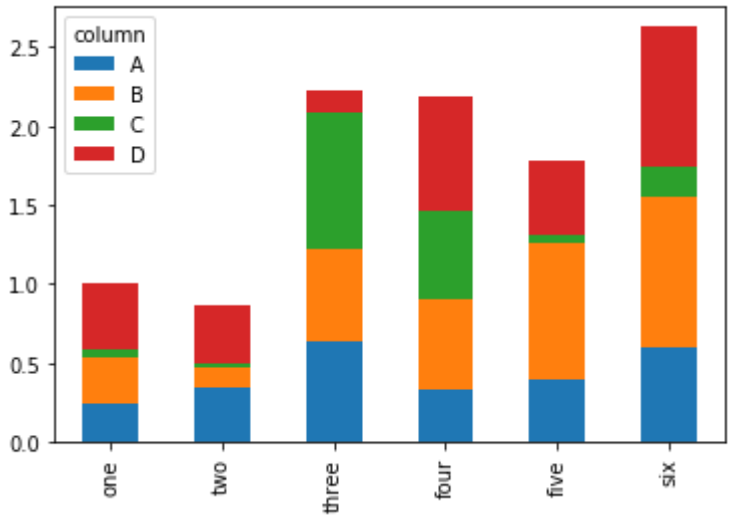
1.2.1. 여섯번째 그래프: data.frame을 그래프로 그리기

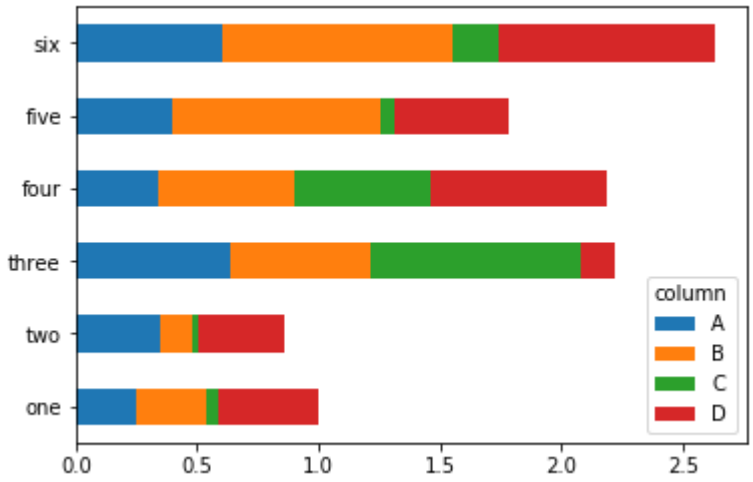
Out[46]:

column	A	B	C	D
1	0.547922	0.005280	0.742668	0.354096
2	0.732152	0.321847	0.513796	0.981808
3	0.129532	0.634703	0.983994	0.585614
4	0.212694	0.622099	0.614847	0.572731
5	0.607056	0.239213	0.303188	0.023149



1.2.1. 일곱번째 그래프: Stacked Bar Chart



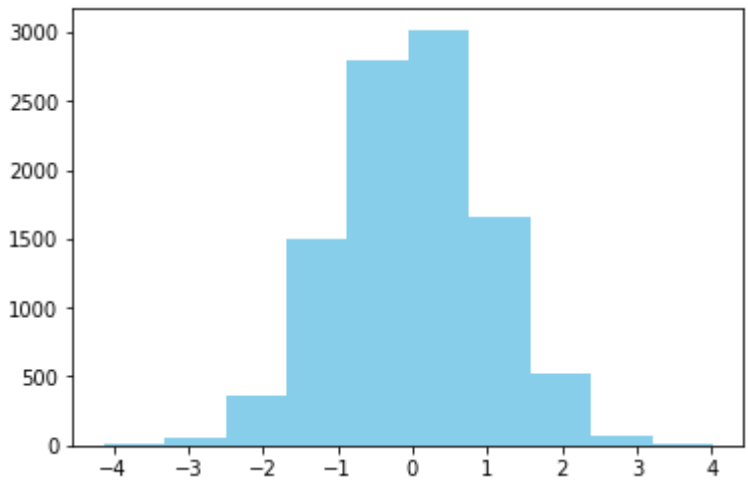


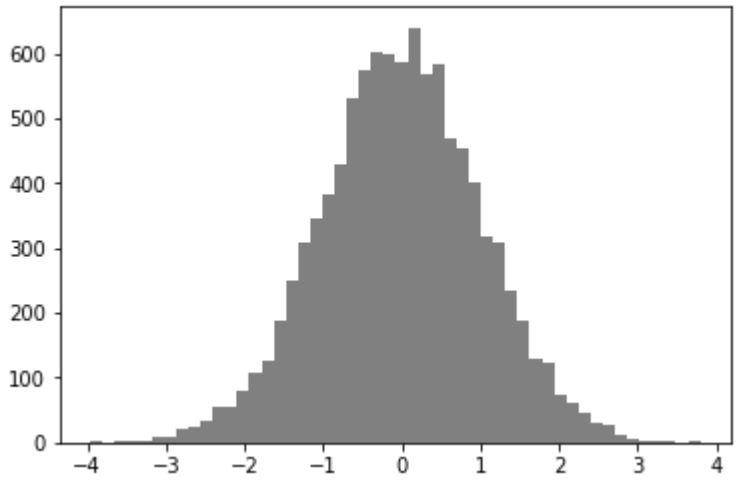
1.3. Histogram

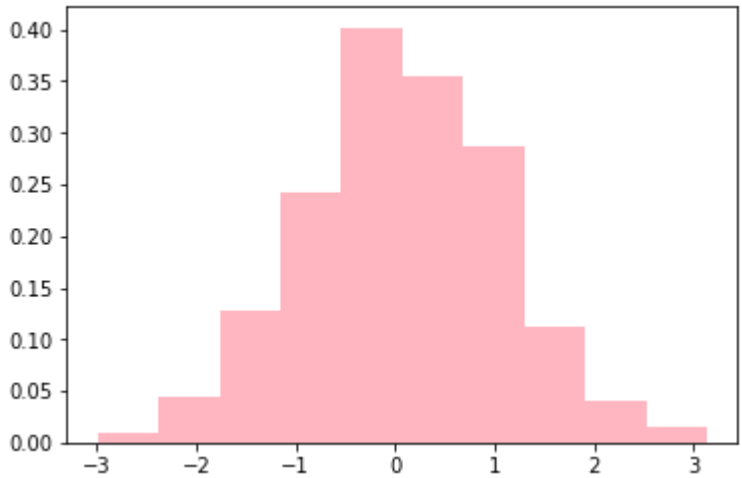
히스토그램은 확률분포의 그래픽적인 표현으로 막대그래프 중 한 종류. 확률분포와 관계가 있으므로 통계적 요소를 나타내기 위해 많이 사용한다.

> **사용 방법:**
> plt.hist(`data` , bins = 숫자 , density = True or False(False가 default))

`bins` 는 구간의 개수를 의미하고, `density=True` 는 빈도수가 아닌 비율로 나타내는 것을 의미한다.







1.4. Pie Chart

1차원의 데이터를 인자로 전달하면 데이터의 상대적 면적을 자동으로 계산하여 그래프를 그려주는 것이 파이차트이다. 단일 범주형 변수 대상으로 구성비를 확인하는 데 유용하다.

> **사용 방법:**
> plt.pie(`data`)