ch.3 Visualizing Data

Ch.3 데이터 시각화를 위한 두 가지 주요 용도

데이터 시각화를 위한 두 가지 주요 용도:

- 데이터를 탐색하려면
- 데이터 통신 방법
- 데이터 시각화는 책을 읽을 가치가 있는 풍부한 연구 분야입니다.

Matplotlib

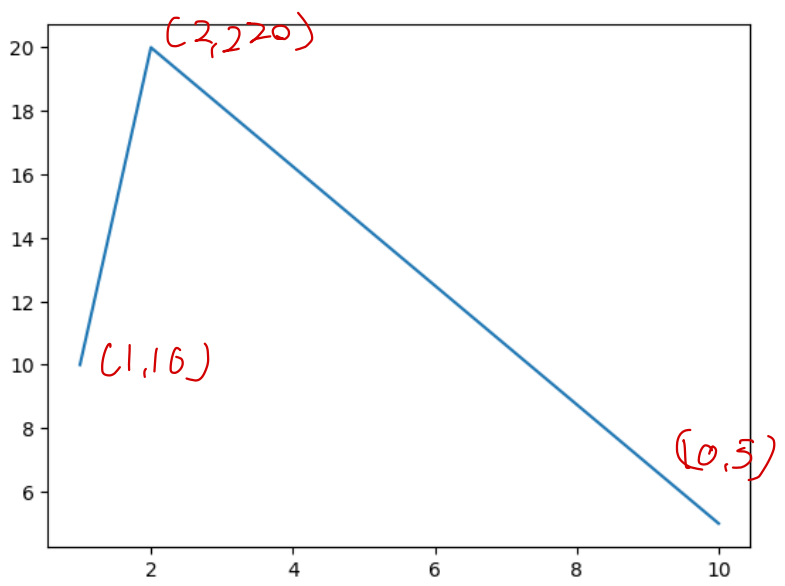
- 널리 사용됨
- 단순 막대 차트, 선 차트 및 산점도에 적합
- 매트플롯 lib.pyplot 모듈

```
import matplotlib.pyplot as plt
```

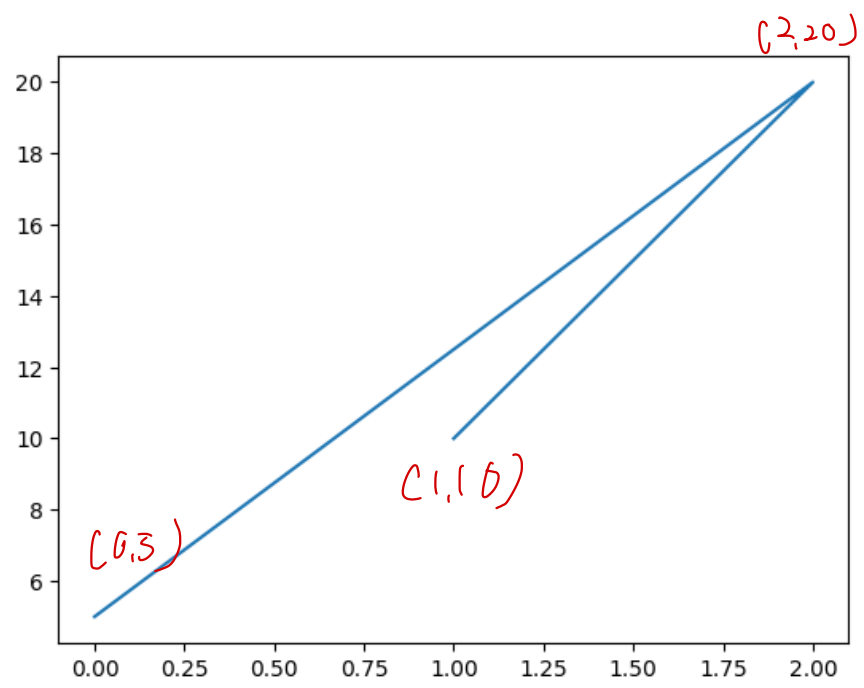
- 이 코드는 (1, 10), (2, 20), 그리고 (10, 5) 세 점을 연결하는 선을 그립니다. 첫 번째 리스트 [1, 2, 10]은 x축 좌표를 나타내고, 두 번째 리스트 [10, 20, 5]는 y축 좌표를 나타냅니다.

`[1, 2, 10]`의 x 좌표와 `[10, 20, 5]`의 y 좌표를 가지는 세 개의 점을 이어 선 그래프를 생성

```
plt.plot([1,2,10],[10,20,5])
plt.show()
```

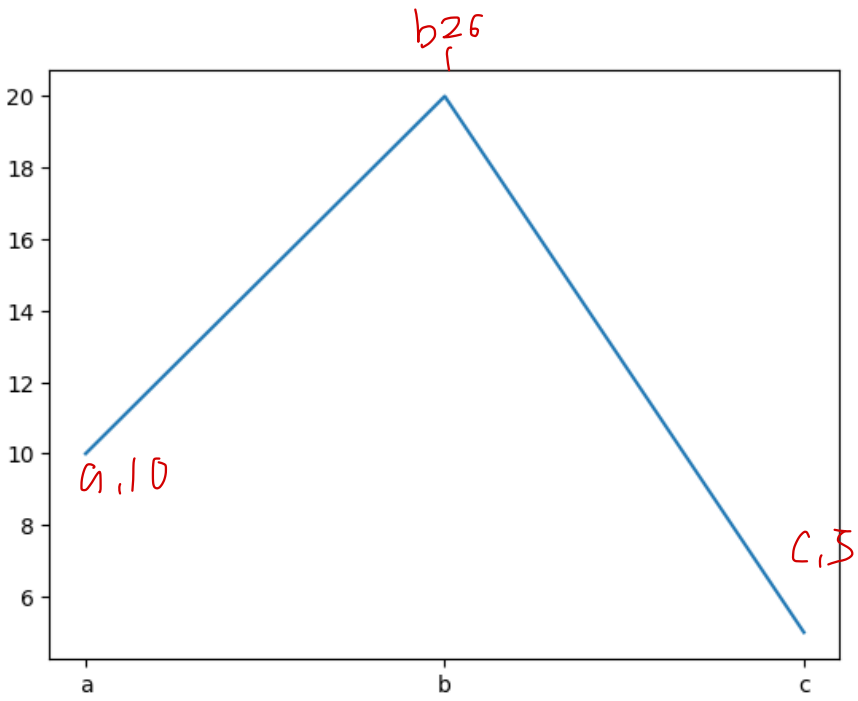


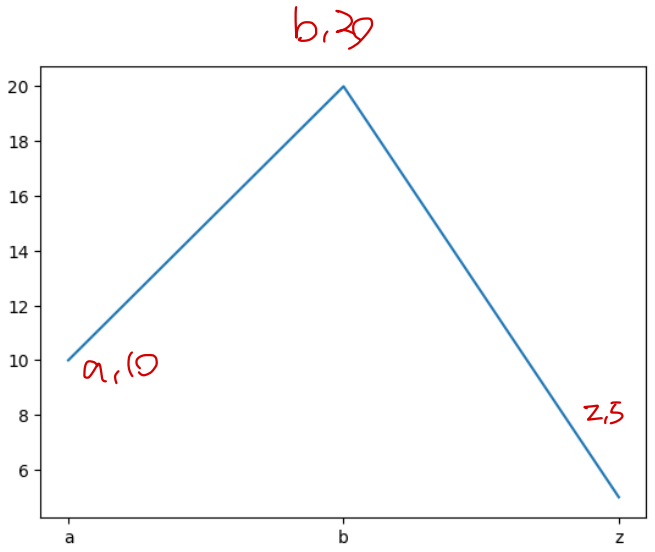

```
plt.plot([1,2,0],[10,20,5])
plt.show()
```



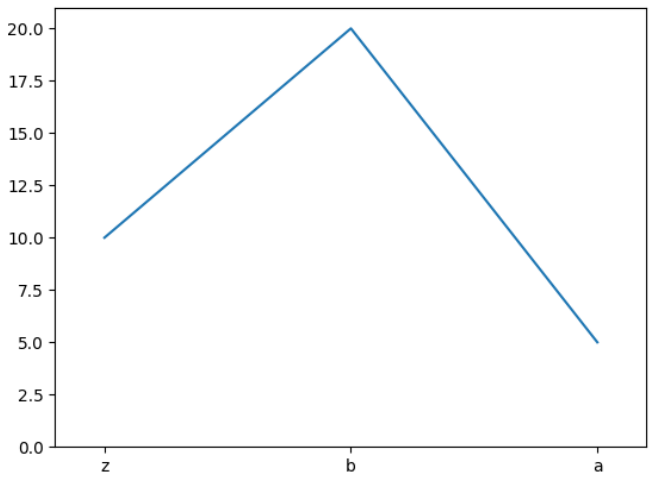
```
plt.plot(['a','b','c'],[10,20,5])
plt.show()
```

```
# just rank
plt.plot(['a','b','z'],[10,20,5])
plt.show()
```






```
# x축과 y축의 범위 설정
plt.axis([-0.2, 2.2, 0, 21])

# 선 그래프 생성
plt.plot(['z', 'b', 'a'], [10, 20, 5])

# 생성한 그래프를 화면에 표시
plt.show()
```

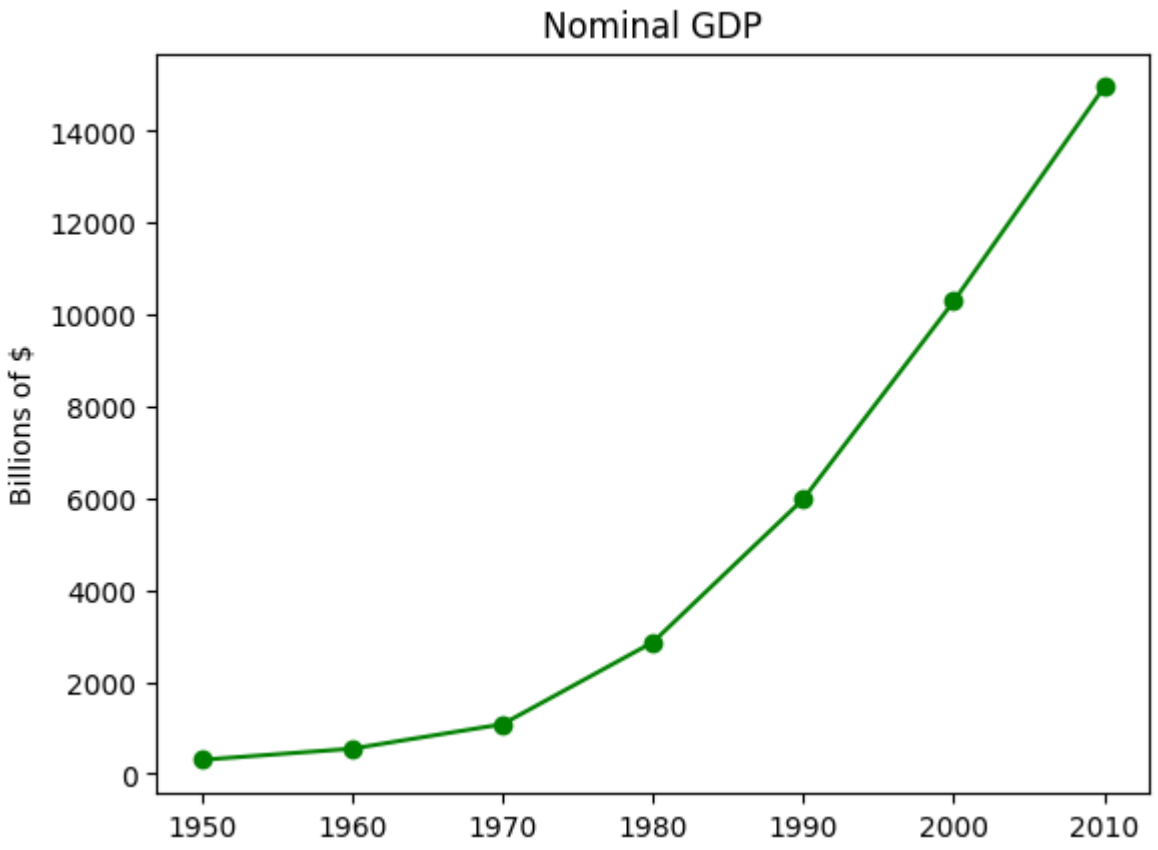


1. `years` 리스트에는 연도 데이터를, `gdp` 리스트에는 해당 연도의 GDP 데이터를 저장합니다.
2. `plt.plot()` 함수를 사용하여 라인 차트를 생성합니다. x축에는 연도 (`years`)를, y축에는 GDP (`gdp`)를 나타냅니다. 라인의 색깔은 녹색으로 지정되었고, 마커는 'o' (원형)로, 선 스타일은 'solid'로 설정되었습니다.
3. `plt.title()` 함수를 사용하여 차트에 제목을 추가합니다. 제목은 "Nominal GDP"로 지정되었습니다.
4. `plt.ylabel()` 함수를 사용하여 y축에 레이블을 추가합니다. 여기서는 "Billions of $"로 설정되었습니다.
5. `plt.show()` 함수를 호출하여 차트를 화면에 표시합니다.

```
def make_chart_simple_line_chart():
    # 연도 데이터를 담은 리스트
    years = [1950, 1960, 1970, 1980, 1990, 2000, 2010]
    # 해당 연도의 GDP 데이터를 담은 리스트
    gdp = [300.2, 543.3, 1075.9, 2862.5, 5979.6, 10289.7, 14958.3]

    # 라인 차트 생성: x축에는 연도, y축에는 GDP
    # 색깔은 녹색, 마커는 'o' (원형), 선 스타일은 실선으로 설정
    plt.plot(years, gdp, color='green', marker='o', linestyle='solid')

    # 차트에 제목 추가
    plt.title("Nominal GDP")

    # y축에 레이블 추가
    plt.ylabel("Billions of $")

    # 생성한 차트를 화면에 표시
    plt.show()
```

```
# 함수 호출
make_chart_simple_line_chart()
```

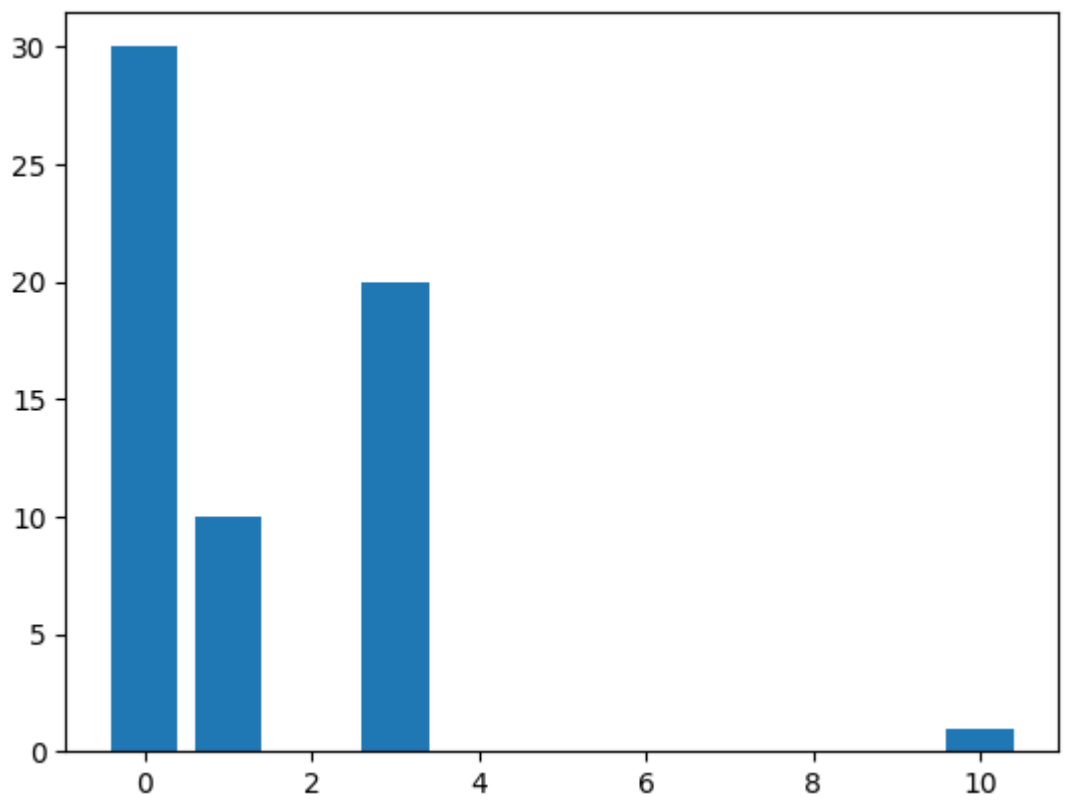
```
make_chart_simple_line_chart()
```



Bar Charts

- 막대 차트는 일부 개별 항목 집합에서 일부 수량이 어떻게 다른지 보여주고 싶을 때 좋은 선택입니다.

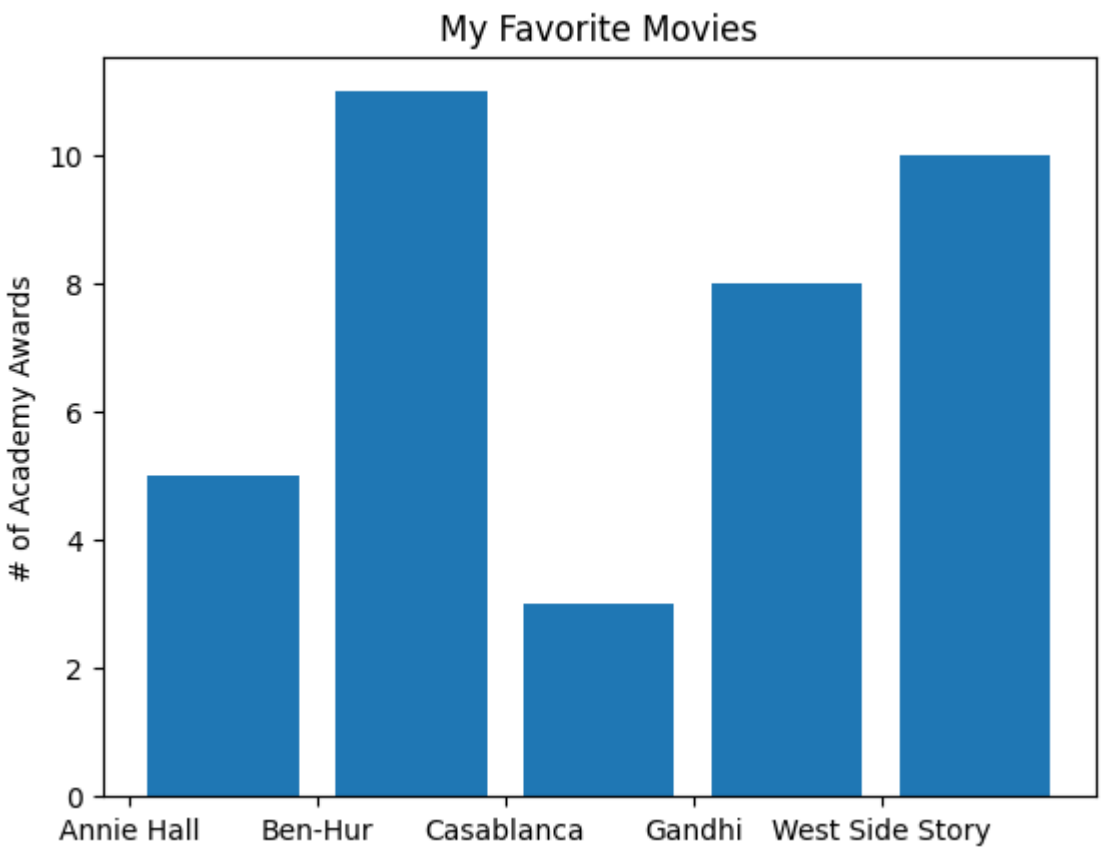
```
plt.bar([1, 3, 0, 10],[10,20,30,1])
plt.show()
```



- `movies` 리스트는 다섯 개의 영화 제목을 포함하고 있습니다.
- `num_oscars` 리스트는 각 영화가 수상한 아카데미 상의 수를 나타냅니다.
- `xs` 리스트는 각 막대의 x좌표를 조정하여 막대가 중앙에 위치하도록 합니다.

- `plt.bar()` 함수를 사용하여 막대 그래프를 생성합니다. x축은 영화 제목을, y축은 아카데미 상의 개수를 나타냅니다.
- `plt.ylabel()` 함수는 y축에 레이블 "# of Academy Awards"를 추가합니다.
- `plt.title()` 함수는 차트에 제목 "My Favorite Movies"를 추가합니다.
- `plt.xticks()` 함수는 x축 레이블로 영화 제목을 사용하여 막대의 가운데에 영화 제목을 표시합니다.
- `plt.show()` 함수는 생성된 차트를 화면에 표시합니다.

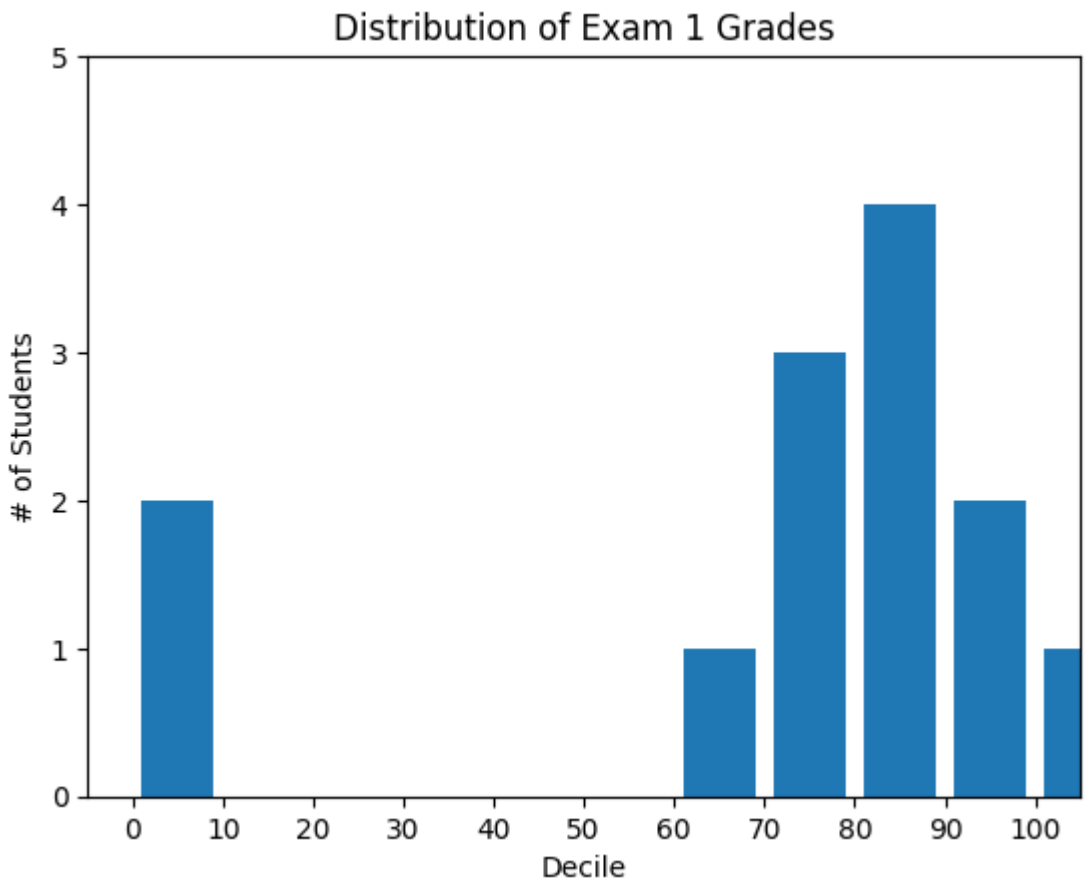
```
def make_chart_simple_bar_chart():
    # 영화 제목과 해당 영화가 수상한 아카데미 상의 개수를 담은 리스트
    movies = ["Annie Hall", "Ben-Hur", "Casablanca", "Gandhi", "West Side Story"]
    num_oscars = [5, 11, 3, 8, 10]

    # 각 막대가 중앙에 위치하도록 하기 위해 왼쪽 좌표에 0.1을 추가
    xs = [i + 0.5 for i, _ in enumerate(movies)]
    # 막대 그래프 생성: x축은 영화 제목, y축은 아카데미 상 개수
    plt.bar(xs, num_oscars)

    # y축 레이블 추가
    plt.ylabel("# of Academy Awards")
    # 차트 제목 추가
    plt.title("My Favorite Movies")

    # 막대의 가운데에 영화 제목을 표시하기 위해 x축 레이블 설정
    plt.xticks([i for i, _ in enumerate(movies)], movies)

    # 생성한 차트를 화면에 표시
    plt.show()

# 함수 호출
make_chart_simple_bar_chart()
```



- 막대 차트는 값이 분포되는 방식을 시각적으로 탐색하기 위해 버킷 숫자 값의 히스토그램을 표시하는 데에도 좋은 선택이 될 수 있습니다

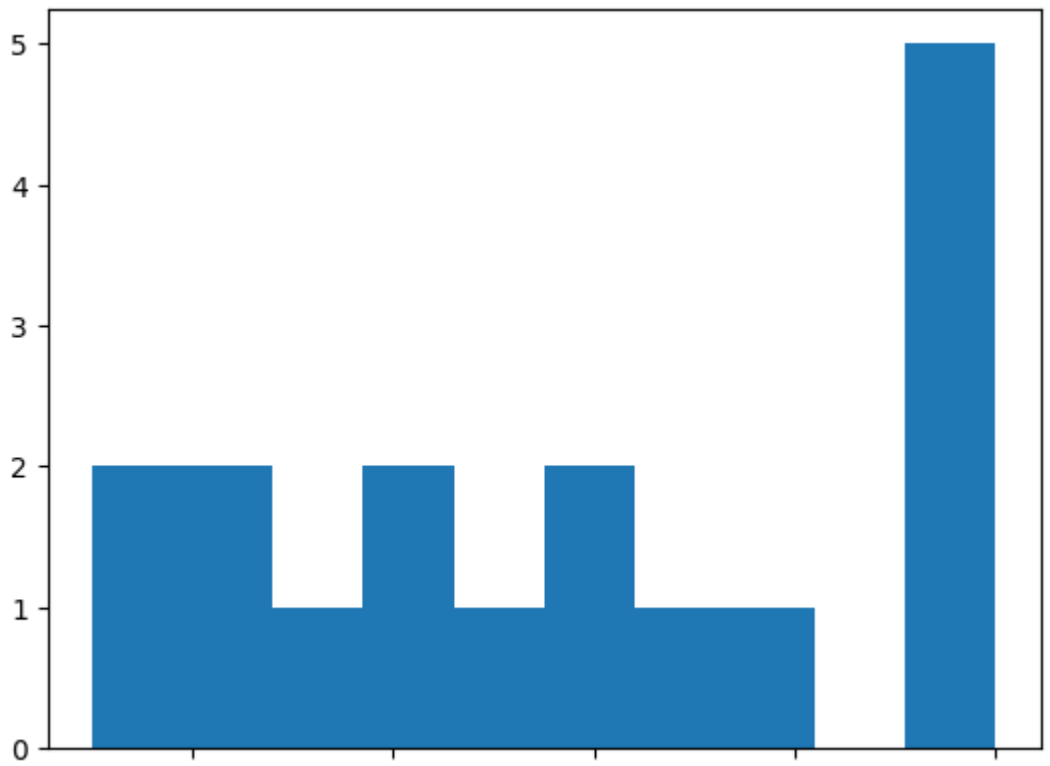
```
from collections import Counter

def make_chart_histogram():
```

```
    # 학생들의 시험 성적을 나타내는 리스트
    grades = [83, 95, 91, 87, 70, 0, 85, 82, 100, 67, 73, 77, 0]

    # 성적을 10의 배수에 해당하는 구간으로 변환하는 람다 함수
    decile = lambda grade: grade // 10 * 10
    # Counter 객체를 사용하여 각 구간에 속하는 학생 수를 세어 저장
    histogram = Counter(decile(grade) for grade in grades)

    # 막대 그래프 생성: 각 막대는 특정 구간에 속하는 학생 수를 나타냄
    plt.bar([x + 5 for x in histogram.keys()],  # 막대를 왼쪽으로 5만큼 이동
            histogram.values(),                 # 각 막대의 높이 지정
            8)                                  # 막대의 너비 지정

    # x축 범위 설정: -5부터 105까지, y축 범위 설정: 0부터 5까지
    plt.axis([-5, 105, 0, 5])
    # x축 레이블 설정: 0부터 100까지 10의 배수로 표시
    plt.xticks([10 * i for i in range(11)])
    # x축 레이블 지정
    plt.xlabel("Decile")
    # y축 레이블 지정
    plt.ylabel("# of Students")
    # 차트 제목 지정
    plt.title("Distribution of Exam 1 Grades")

    # 생성한 히스토그램을 화면에 표시
    plt.show()

# 함수 호출
make_chart_histogram()
```




```
# 데이터
data = [1, 2, 3, 4, 5, 6, 7, 8, 1, 2, 4, 6, 10, 10, 10, 10, 10]

# 히스토그램 생성
plt.hist(data, bins=10)
```

```
# 생성한 히스토그램을 화면에 표시
plt.show()
```

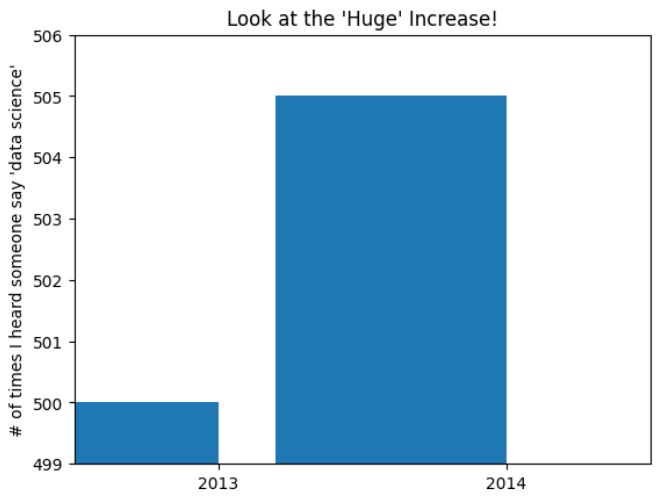


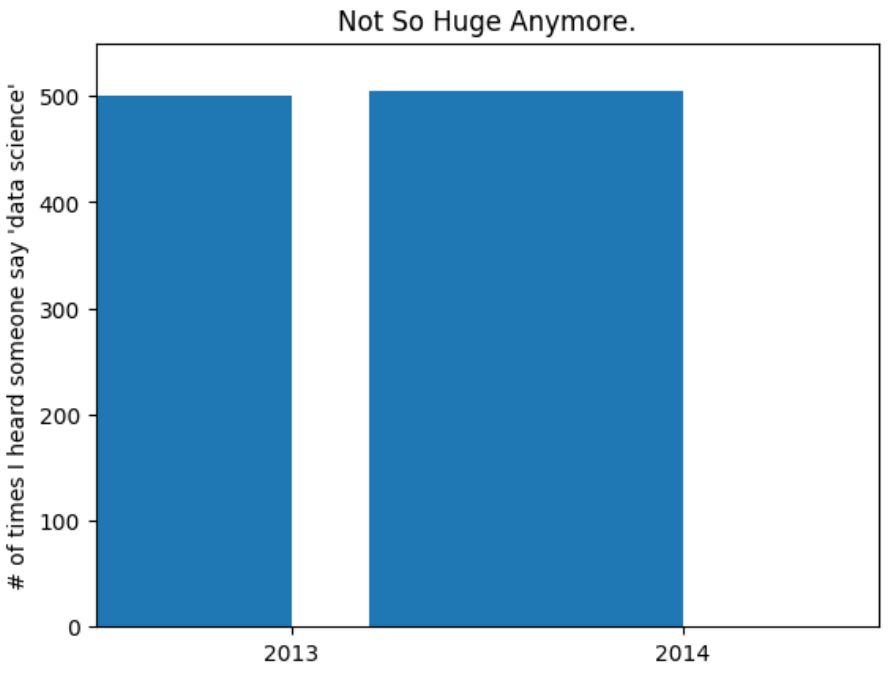
Misleading bar chart

```
def make_chart_misleading_y_axis(mislead=True):
    # "data science"이라는 문구가 언급된 횟수
    mentions = [500, 505]
    # 연도
    years = [2013, 2014]

    # 막대 그래프 생성
    plt.bar([2012.6, 2013.6], mentions, 0.8)
    # x축에 연도 레이블 추가
    plt.xticks(years)
    # y축 레이블 추가
    plt.ylabel("# of times I heard someone say 'data science'")

    if mislead:
        # y축을 오도하여 500 이상의 부분만 보이게 함
        plt.axis([2012.5, 2014.5, 499, 506])
        # 오도한 차트에 대한 제목 추가
        plt.title("Look at the 'Huge' Increase!")
    else:
        # 오도하지 않은 정상적인 차트
        plt.axis([2012.5, 2014.5, 0, 550])
        # 정상적인 차트에 대한 제목 추가
        plt.title("Not So Huge Anymore.")

    # 생성한 차트를 화면에 표시
    plt.show()
```

```
make_chart_misleading_y_axis()
```

```
make_chart_misleading_y_axis(mislead=False)
```

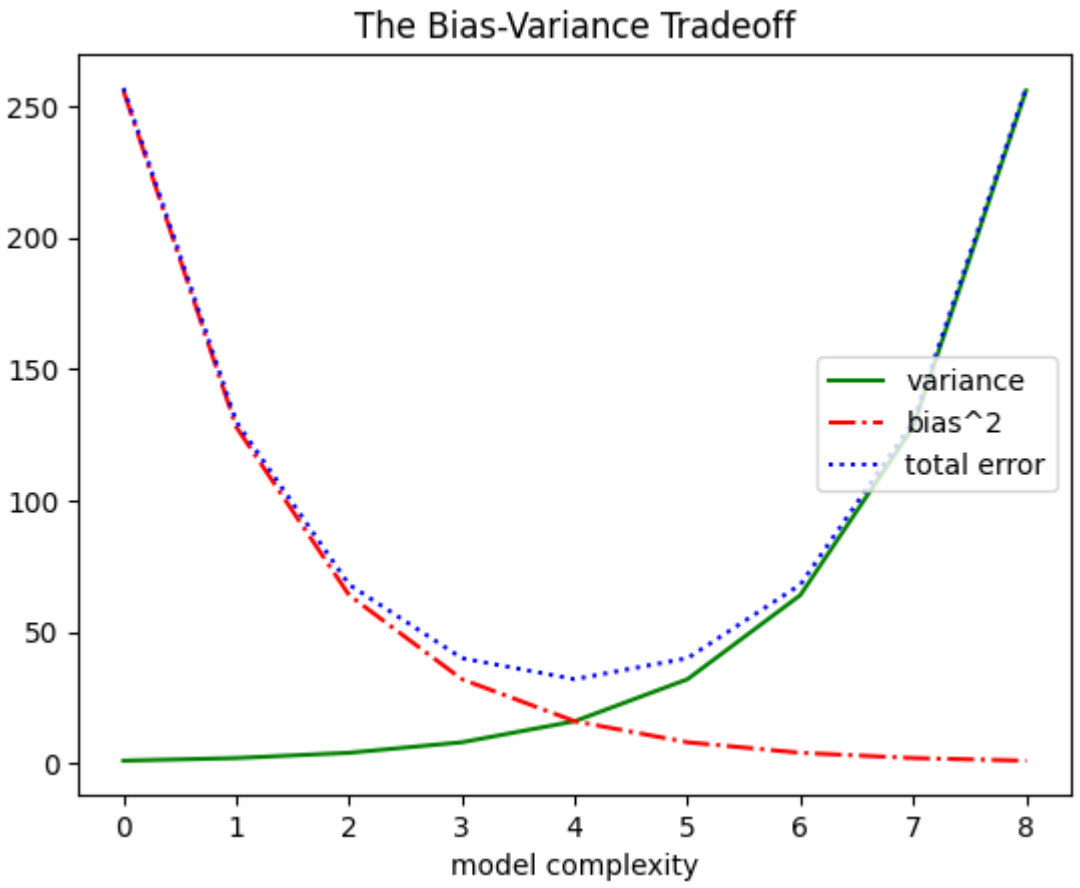




Line Charts

- plt.plot()을 사용한 선형 차트
- **트렌드를 보여주기 위한 좋은 선택**

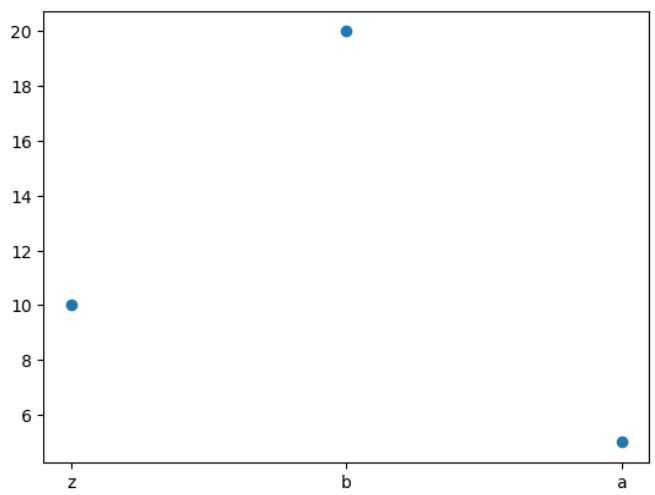
```
def make_chart_several_line_charts():
    # 분산
    variance = [1, 2, 4, 8, 16, 32, 64, 128, 256]
    # 편향 제곱
    bias_squared = [256, 128, 64, 32, 16, 8, 4, 2, 1]
    # 총 오차
    total_error = [x + y for x, y in zip(variance, bias_squared)]

    xs = range(len(variance))

    # 여러 개의 plt.plot 호출로 동일한 차트에 여러 시리즈를 표시할 수 있음
    # 각 시리즈에 레이블을 할당했으므로 범례가 자동으로 생성됨
    plt.plot(xs, variance, 'g-', label='variance')  # 녹색 실선
    plt.plot(xs, bias_squared, 'r-.', label='bias^2')  # 빨간색 점선
    plt.plot(xs, total_error, 'b:', label='total error')  # 파란색 점선

    # 범례 위치 설정: loc=5는 "오른쪽 상단"
    plt.legend(loc=5)
    plt.xlabel("model complexity")  # x축 레이블 설정
    plt.title("The Bias-Variance Tradeoff")  # 차트 제목 설정
    plt.show()  # 생성한 차트를 화면에 표시

# 함수 호출
make_chart_several_line_charts()
```



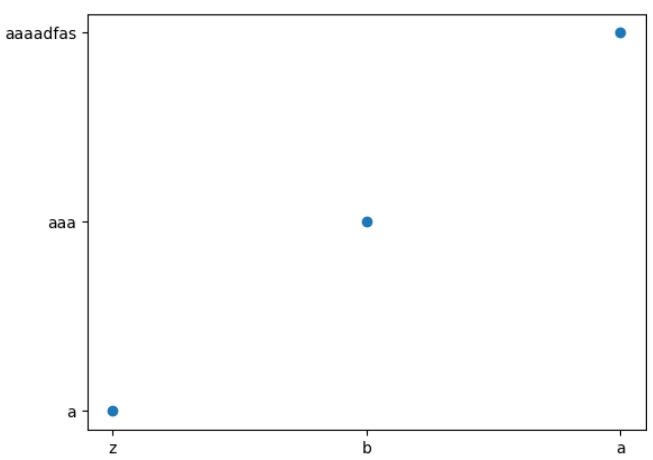
Scatter plots

- 두 쌍의 **데이터 집합 사이의** 관계를 시각화하기 위한 올바른 선택입니다.
- 사용자가 보유한 친구 수와 매일 사이트에서 사용하는 시간(분) 간의 관

```
plt.scatter(['z','b','a'],[10,20,5])
plt.show()
```

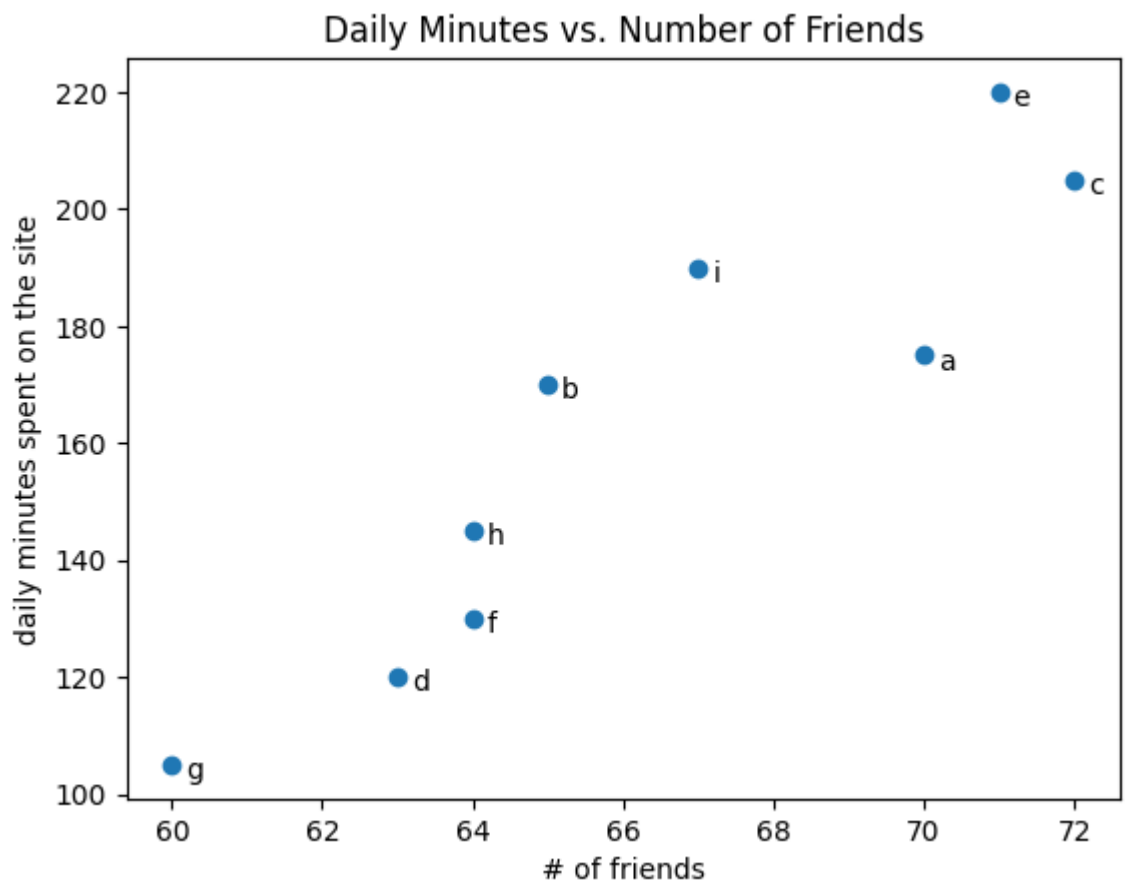



```
plt.scatter(['z','b','a'],['a','aaa', 'aaaadf
# plt.axis()
plt.show()
```




```
def make_chart_scatter_plot():
    # 친구 수
    friends = [70, 65, 72, 63, 71, 64, 60, 64, 67]
    # 사이트에서 보내는 일일 시간
    minutes = [175, 170, 205, 120, 220, 130, 105, 145, 190]
    # 각 점에 대한 레이블
    labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']

    # 산점도 생성
    plt.scatter(friends, minutes)

    # 각 점에 레이블 추가
    for label, friend_count, minute_count in zip(labels, friends, minutes):
        plt.annotate(label,
```

```
                        xy=(friend_count, minute_count),  # 점과 레이블을 동시에 표시
                        xytext=(5, -5),  # 약간의 오프셋 설정
                        textcoords='offset points')

    # 차트 제목 설정
    plt.title("Daily Minutes vs. Number of Friends")
    # x축 레이블 설정
    plt.xlabel("# of friends")
    # y축 레이블 설정
    plt.ylabel("daily minutes spent on the site")
    # 생성한 차트를 화면에 표시
    plt.show()

# 함수 호출
make_chart_scatter_plot()
```

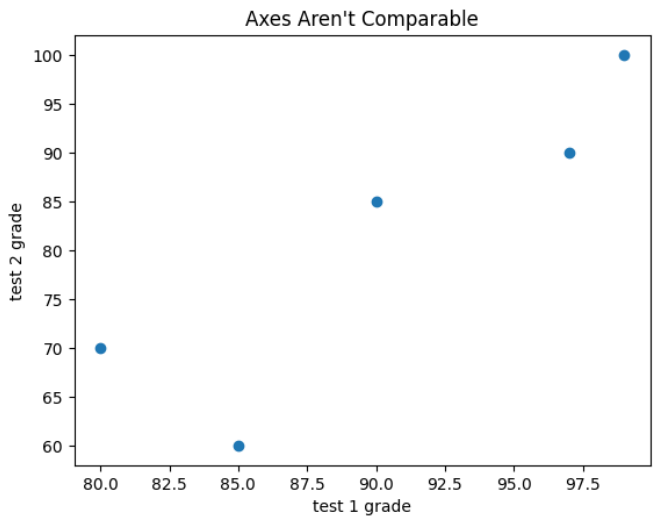


- 유사한 변수를 산포하는 경우 matplotlib에서 척도를 선택하도록 하면 오해의 소지가 있는 그림이 나타날 수 있습니다

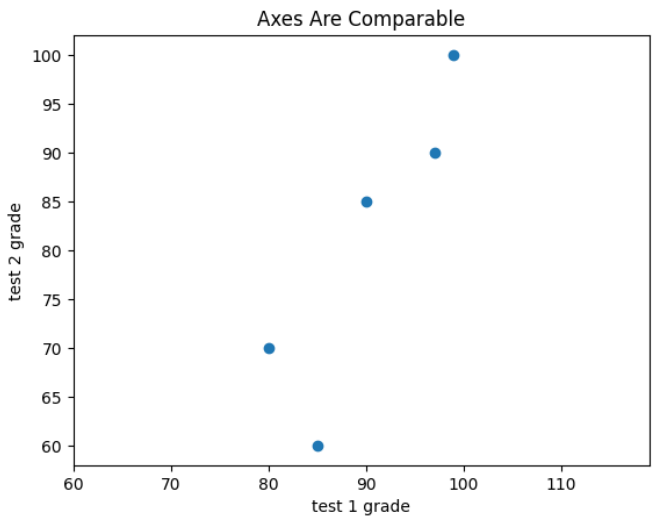
```
def make_chart_scatterplot_axes(equal_axes=False):
    # 시험 1 점수
    test_1_grades = [99, 90, 85, 97, 80]
    # 시험 2 점수
    test_2_grades = [100, 85, 60, 90, 70]

    # 산점도 생성
    plt.scatter(test_1_grades, test_2_grades)
    # x축 레이블 설정
    plt.xlabel("test 1 grade")
    # y축 레이블 설정
    plt.ylabel("test 2 grade")

    if equal_axes:
        # 축의 척도를 동일하게 설정하여 차트를 생성
        plt.title("Axes Are Comparable")
        plt.axis("equal")
    else:
        # 기본적인 축 설정으로 차트 생성
```

```
        plt.title("Axes Aren't Comparable")

    # 생성한 차트를 화면에 표시
    plt.show()
```

```
make_chart_scatterplot_axes()
```



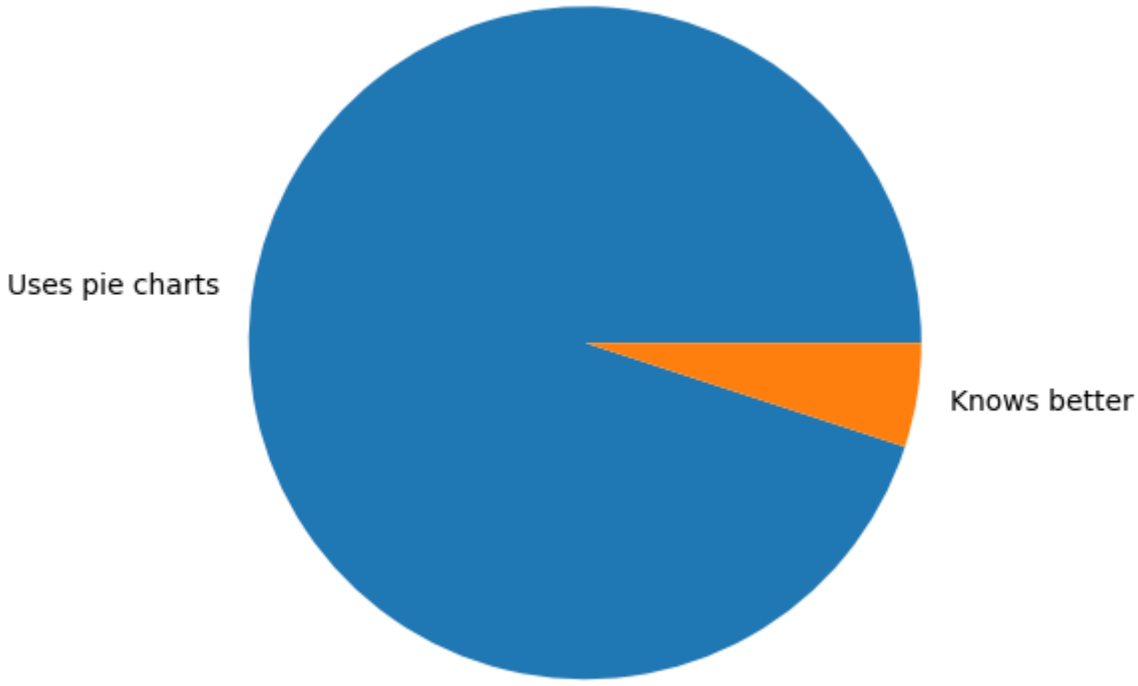

```
make_chart_scatterplot_axes(equal_axes=True)
```



Pie Charts

- **숫자 비율을 설명하기 위해 슬라이스로 나눈 원**

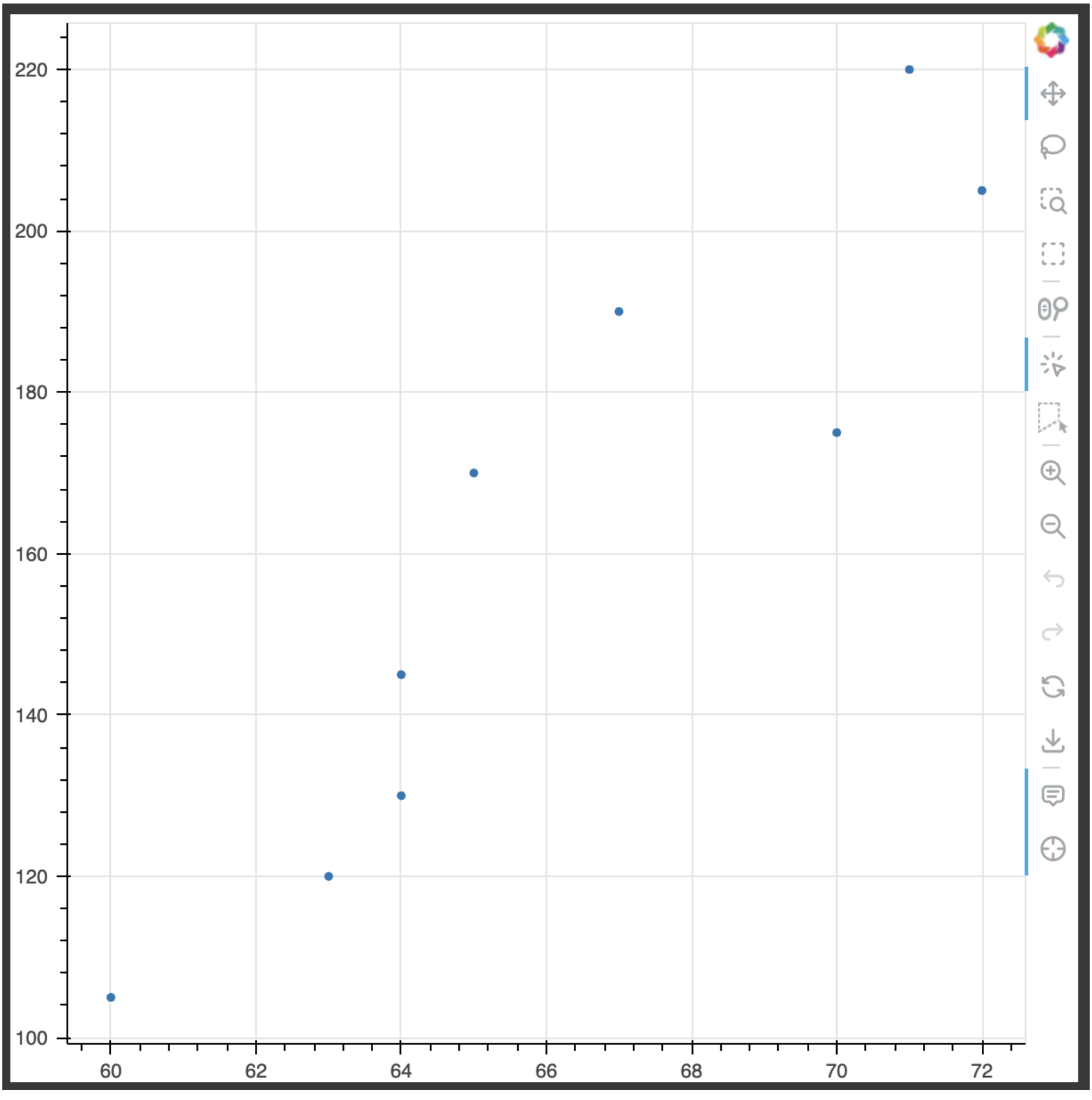
```
def make_chart_pie_chart():
    # 파이 차트 생성: 각 범주의 비율을 나타냄
    plt.pie([0.95, 0.05], labels=["Uses pie charts", "Knows better"])

    # 파이가 원형으로 표시되도록 설정
    plt.axis("equal")
    # 생성한 차트를 화면에 표시
    plt.show()

# 함수 호출
make_chart_pie_chart()
```



Data scientists move to bokeh

- Bokeh는 D3 스타일(대화형)의 시각화를 Python에 도입한 새로운 라이브러리입니다
- https://demo.bokeh.org/movies

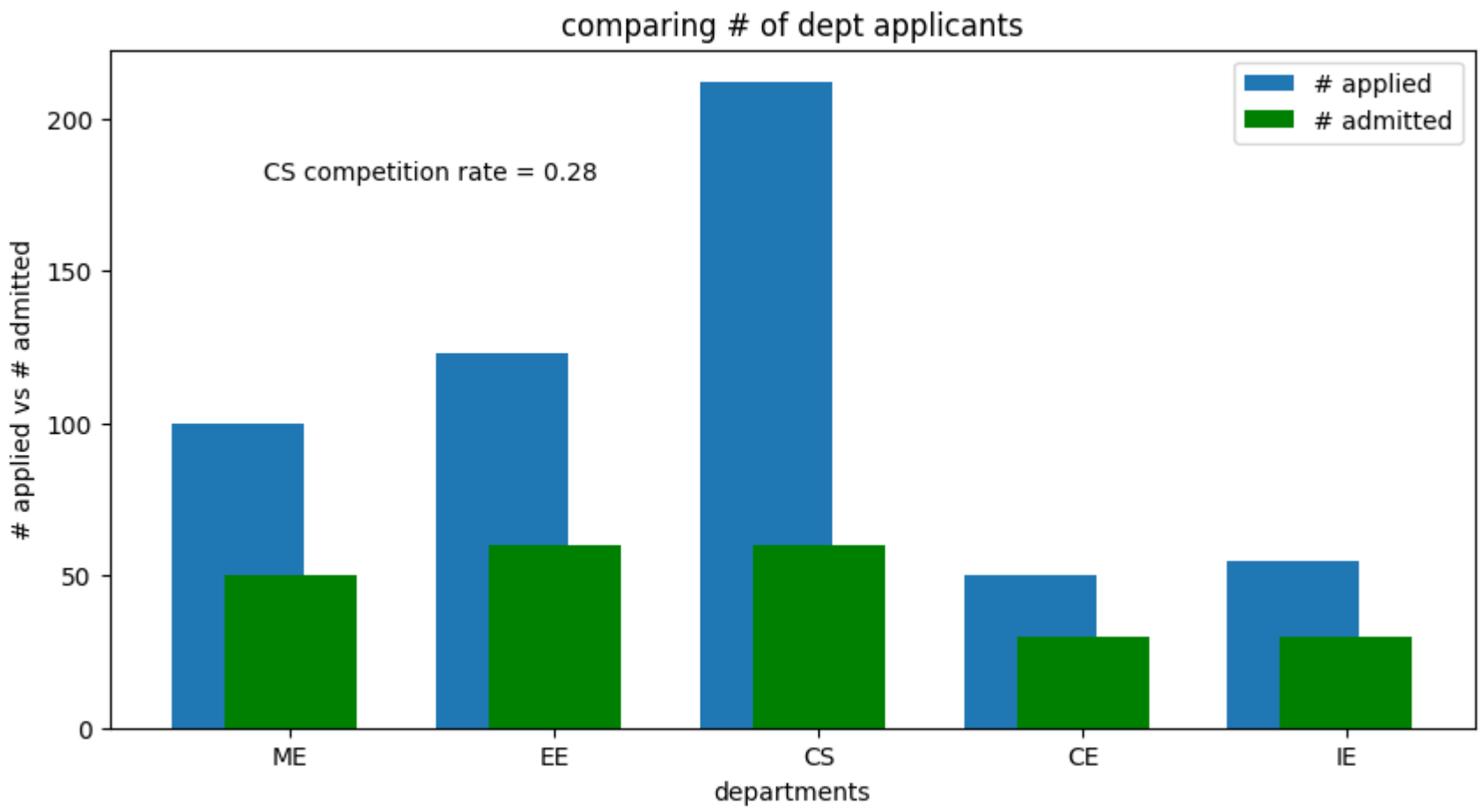
```
# Bokeh Libraries
from bokeh.io import output_notebook
from bokeh.plotting import figure, show
import random

# 데이터
friends = [70, 65, 72, 63, 71, 64, 60, 64, 67]
minutes = [175, 170, 205, 120, 220, 130, 105, 145, 190]

# 주석 처리된 부분은 데이터를 조작하여 추가적인 노이즈를 만드는 부분입니다.
# 이 코드는 현재 주석 처리되어 있습니다.
# friends = [i + 3 * random.random() for i in friends for _ in range(100)]
# minutes = [i + 50 * random.random() for i in minutes for _ in range(100)]

# Jupyter Notebook에서 Bokeh 출력을 사용합니다.
output_notebook()

# 사용할 도구들
TOOLS="hover,crosshair,pan,wheel_zoom,zoom_in,zoom_out,box_zoom,undo,redo,reset,tap,save,box_se

# 일반적인 figure() 객체 설정
fig = figure(tools=TOOLS)

# 산점도 생성
fig.scatter(friends, minutes)

# 생성한 그래프 보기
show(fig)
```



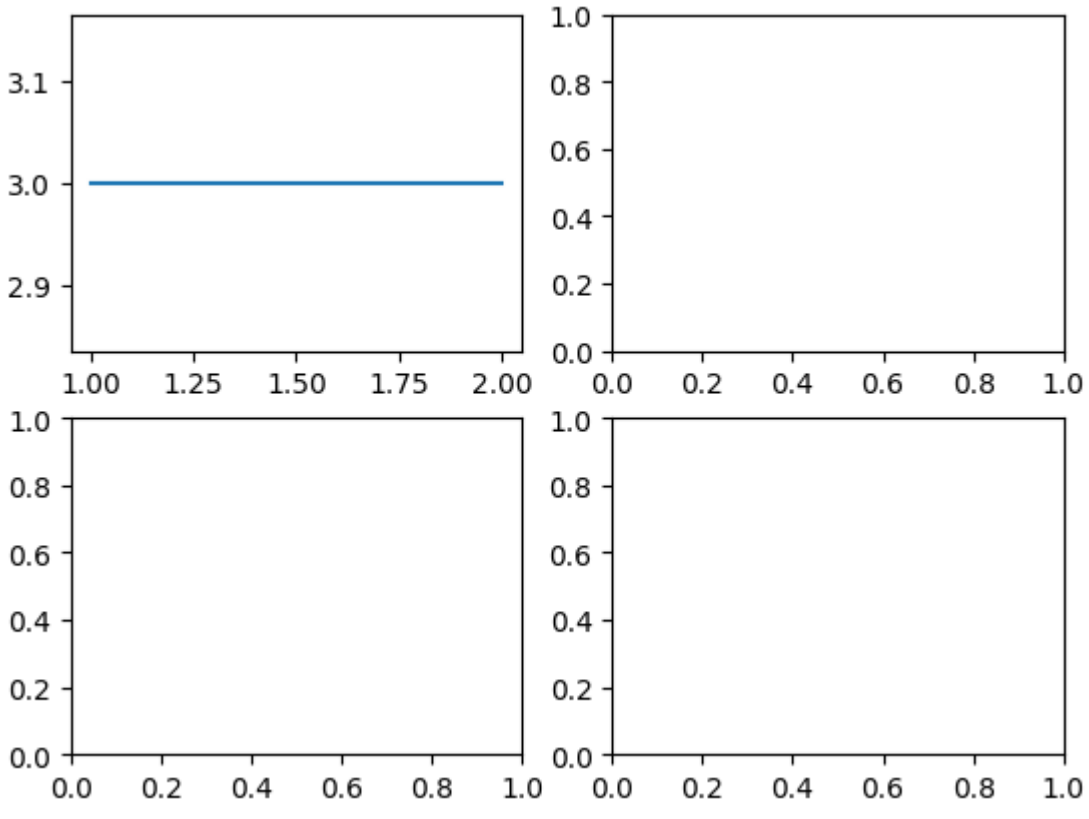
Complete Example

```
import matplotlib.pyplot as plt
import numpy as np

# 그래프 크기 설정
plt.figure(figsize=(10,5))

# 학과 이름과 지원자 수, 입학자 수 데이터
dept_names = ['ME', 'EE', 'CS', 'CE', 'IE']
num_apps = [100, 123, 212, 50, 55]
num_adms = [50, 60, 60, 30, 30]

# 지원자 수 막대 그래프 생성
plt.bar(np.array(range(len(dept_names))) - 0.2, num_apps, width=0.5, label='# applied')
# 입학자 수 막대 그래프 생성
```

```
plt.bar(range(len(dept_names)), num_adms, color='g', width=0.5, label='# admitted')

# x축에 학과 이름 표시
plt.xticks(range(len(dept_names)), dept_names)
# y축 눈금 설정
plt.yticks(range(0, 250, 50))
# 범례 위치 설정
plt.legend(loc=1)
# 차트 제목 설정
plt.title('comparing # of dept applicants')
# x축 레이블 설정
plt.xlabel("departments")
# y축 레이블 설정
plt.ylabel("# applied vs # admitted")

# 차트에 주석 추가 (CS 학과의 경쟁률)
plt.annotate('CS competition rate = {:.2}'.format(num_adms[2]/num_apps[2]), xy=(-0.1, 180))

# 생성한 차트를 화면에 표시
plt.show()
```




```
# 2x2 서브플롯을 가지는 그림 생성
fig, ax = plt.subplots(2, 2)

# 첫 번째 서브플롯에 선 그래프 추가
ax[0, 0].plot([1, 2], [3, 3])

# 결과 표시
plt.show()
```


3.1
3.0
2.9
1.00
1.25
1.50
1.75
2.00
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0