In [1]:
```
import pandas as pd
```

In [2]:
```
#Pandas 의 Dataframe을 생성.
#Pandas 를 이용해 Dataframe 으로 불러오거나 생성.
names = ['Bob', 'Jessica', 'Mary', 'John', 'Mel']
births = [968, 155, 77, 578, 973]
custom = [1, 5, 25, 13, 23232]
```

In [3]:
```
# Zip : 하나로 묶기.
BabyDataSet = list(zip(names,births))
df = pd.DataFrame(data = BabyDataSet, columns=['Names', 'Births'])
```

In [4]:
```
for aa in zip(names, births):
    print(aa)
```

```
('Bob', 968)
('Jessica', 155)
('Mary', 77)
('John', 578)
('Mel', 973)
```

In [5]:
```
BabyDataSet
```

Out[5]:
```
[('Bob', 968), ('Jessica', 155), ('Mary', 77), ('John', 578), ('Mel', 973)]
```

In [6]:
```
df.head()
```

Out[6]:

	Names	Births
0	Bob	968
1	Jessica	155
2	Mary	77
3	John	578
4	Mel	973

In [7]:
```
#데이터프레임의 열 타입 정보를 출력.
print(df.dtypes)
print('----------')

#데이터프레임의 형태 정보.
print(df.index)
print('----------')

#데이터프레임의 열 정보.
print(df.columns)
```

```
Names     object
Births     int64
dtype: object
----------
RangeIndex(start=0, stop=5, step=1)
```

```
----------
Index(['Names', 'Births'], dtype='object')
```

In [8]:
```
df.dtypes # 열에 뭐가 있는지? + 각 열이 어떤 type인지
```

Out[8]:
```
Names     object
Births     int64
dtype: object
```

In [9]:
```
df.index # 행에 대한 정보
```

Out[9]:
```
RangeIndex(start=0, stop=5, step=1)
```

In [10]:
```
df.columns # 열에 뭐가 있는지
```

Out[10]:
```
Index(['Names', 'Births'], dtype='object')
```

In [11]:
```
# 데이터프레임의 특정한 하나의 열을 선택.
df['Names']
```

Out[11]:
```
0        Bob
1    Jessica
2       Mary
3       John
4        Mel
Name: Names, dtype: object
```

In [12]:
```
# 0~3번째 인덱스를 선택. (행)
df[0:3]
```

Out[12]:

	Names	Births
0	Bob	968
1	Jessica	155
2	Mary	77

In [13]:
```
# Births 열이 100보다 큰 데이터를 선택.(조건문)
df[df['Births'] > 100]
```

Out[13]:

	Names	Births
0	Bob	968
1	Jessica	155
3	John	578
4	Mel	973

In [14]:
```
df[df['Births'] > 100].head(2)
```

Out[14]:

	Names	Births
0	Bob	968

	Names	Births
1	Jessica	155

In [15]:
```
# 데이터프레임에서의 평균값을 계산.
# 숫자로 된 열이 Births 뿐이라 Births 만 계산 됨.
df.mean()
```

Out[15]:
```
Births    550.2
dtype: float64
```

In [16]:
```
# 통계정보 요약.
# 위의 Mean 도 describe의 일부라고 할 수 있겠다.
df.describe()
```

Out[16]:

	Births
count	5.000000
mean	550.200000
std	428.424672
min	77.000000
25%	155.000000
50%	578.000000
75%	968.000000
max	973.000000

Numpy

In [17]:
```
import numpy as np
```

In [18]:
```
arr1 = np.arange(15).reshape(3,5) #reshape : 배열의 모양 재설정
print(arr1)
```

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]]
```

In [19]:
```
np.arange(10)
```

Out[19]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [20]:
```
python_array = [0, 1, 2, 3, 4, 5, 6]
np.array(python_array)
```

Out[20]:
```
array([0, 1, 2, 3, 4, 5, 6])
```

In [21]:
```
arr1.shape
```

Out[21]: (3, 5)

In [22]:
```
arr3 = np.zeros((3,4)) # 0뿐인 배열
print(arr3)
```

```
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
```

In [23]:
```
arr3 = np.ones((3,4)) # 1뿐인 배열
print(arr3)
```

```
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]
```

In [24]:
```
arr4 = np.array([
    [1,2,3],
    [4,5,6]
], dtype= np.float64)

arr5 = np.array([
    [7,8,9],
    [10,11,12]
], dtype= np.float64)

# 사칙연산
print("arr4 + arr5 = ")
print(arr4 + arr5, "₩n")
print("arr4 - arr5 = ")
print(arr4 - arr5, "₩n")
print("arr4 * arr5 = ")
print(arr4 * arr5, "₩n")
print("arr4 / arr5 = ")
print(arr4 / arr5, "₩n")
# numpy 사용하는 이유는 vector 나 matrix 연산을 위한 library.
# shape 이 맞아야 계산이 가능.
```

```
arr4 + arr5 =
[[ 8. 10. 12.]
 [14. 16. 18.]]

arr4 - arr5 =
[[-6. -6. -6.]
 [-6. -6. -6.]]

arr4 * arr5 =
[[ 7. 16. 27.]
 [40. 55. 72.]]

arr4 / arr5 =
[[0.14285714 0.25       0.33333333]
 [0.4        0.45454545 0.5       ]]
```

Matplotlib

In [25]:
```
# 이 library를 이용하기 위해서 matplotlib inline를 해줘야한다함.
%matplotlib inline
import matplotlib.pyplot as plt
```

In [26]:
```
y=df['Births']
x=df['Names']
```

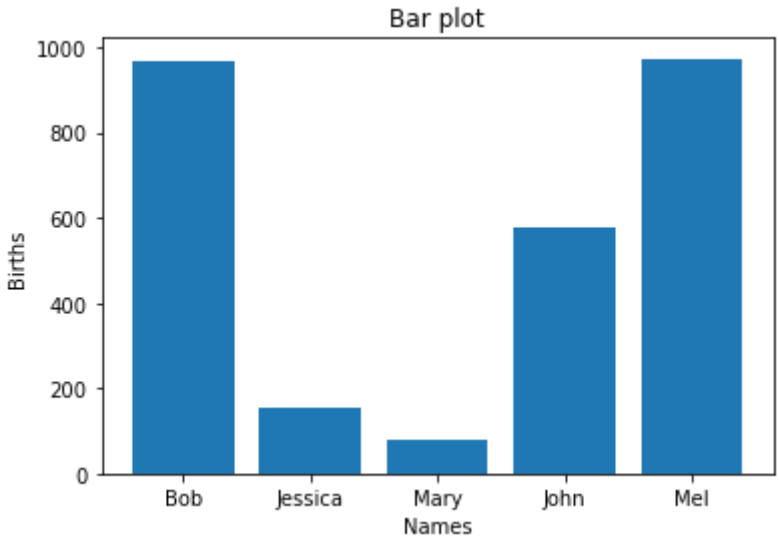
In [27]:
```
print(x)
```

```
0        Bob
1    Jessica
2       Mary
3       John
4        Mel
Name: Names, dtype: object
```

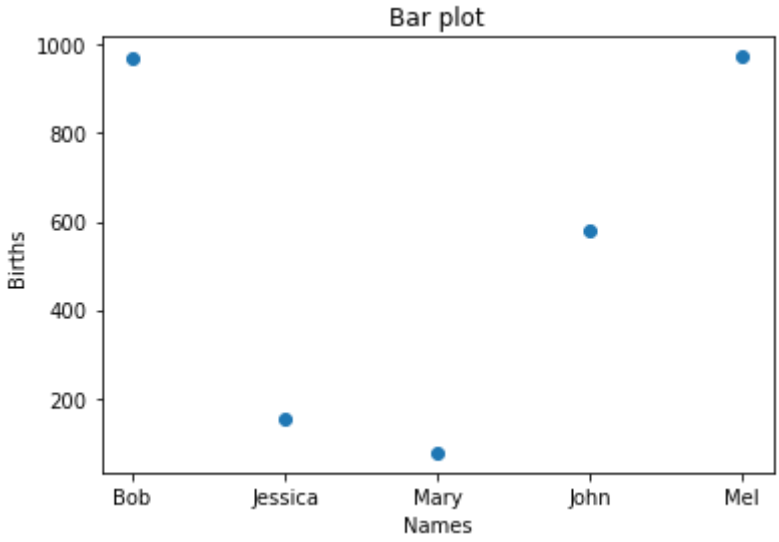
In [28]:
```
print(y)
```

```
0    968
1    155
2     77
3    578
4    973
Name: Births, dtype: int64
```

In [29]:
```
y=df['Births']
x=df['Names']

# bar plot을 출력
plt.bar(x, y) # 막대그래프 객체 생성
plt.xlabel('Names') # x축 제목
plt.ylabel('Births') # y축 제목
plt.title('Bar plot') # 그래프 제목
plt.show() # 그래프 출력
```

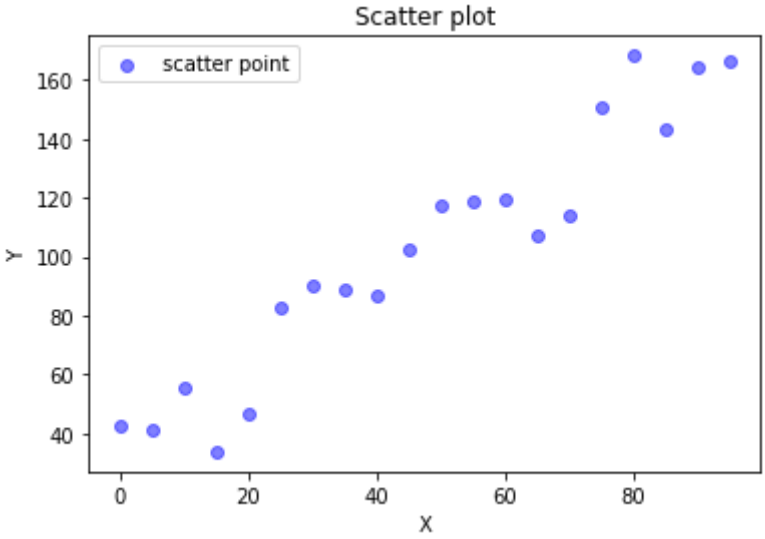


In [30]:
```
y=df['Births']
x=df['Names']

# bar plot을 출력
plt.scatter(x, y) # 점 그래프 객체 생성
plt.xlabel('Names') # x축 제목
plt.ylabel('Births') # y축 제목
plt.title('Bar plot') # 그래프 제목
plt.show() # 그래프 출력
```



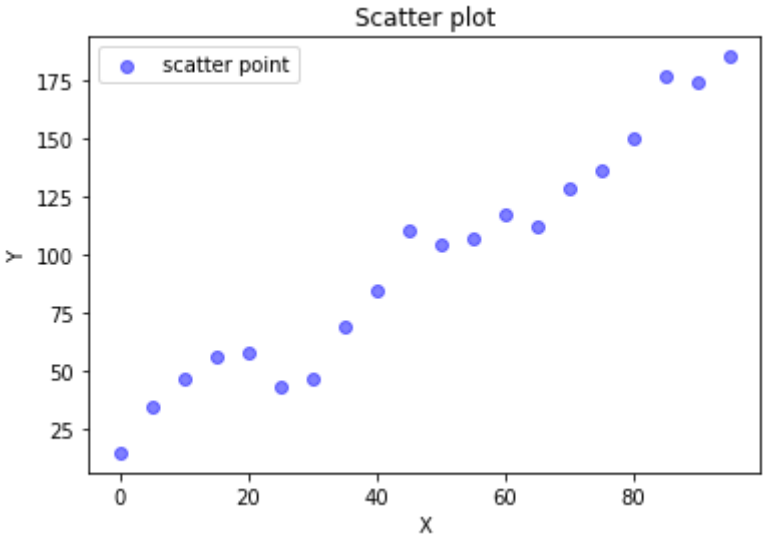
In [31]:

```
# scatter plot (numpy를 이용한)데이터 생성
x = np.arange(0.0, 100.0, 5.0)
y = (x * 1.5) + np.random.rand(20) * 50

# scatter plot 출력 (alpha : 점 크기)
plt.scatter(x, y, c='b', alpha=0.5, label='scatter point')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend(loc='upper left')
plt.title('Scatter plot')
plt.show()
```



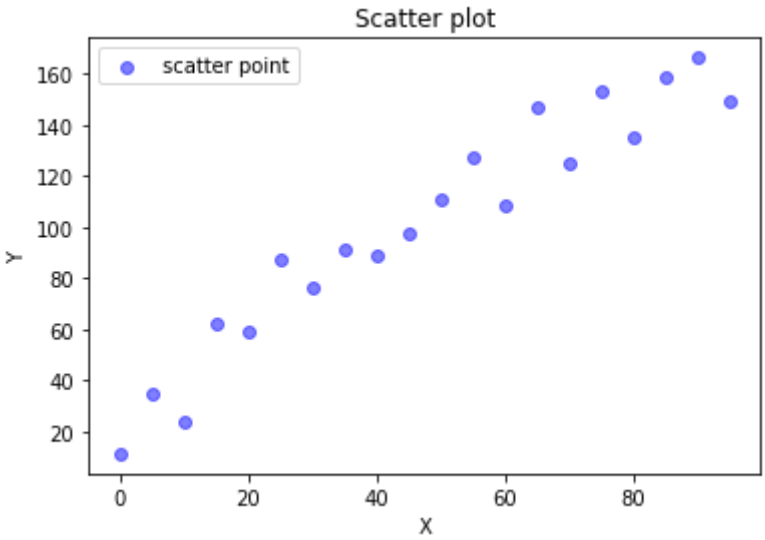
In [32]:

```
# scatter plot (numpy를 이용한)데이터 생성
x = np.arange(0.0, 100.0, 5.0)
y = (x * 1.5) + np.random.rand(20) * 50

# scatter plot 출력 (alpha : 점 크기)
plt.scatter(x, y, c='b', alpha=0.5, label='scatter point')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend(loc='upper left')
plt.title('Scatter plot')
plt.show()

# random 추출이라서 실행때마다 다른 결과가 나온다.
```



In [33]:

```
# random 추출 시드 고정
np.random.seed(202107)

# scatter plot (numpy를 이용한)데이터 생성
x = np.arange(0.0, 100.0, 5.0)
y = (x * 1.5) + np.random.rand(20) * 50

# scatter plot 출력 (alpha : 점 크기)
plt.scatter(x, y, c='b', alpha=0.5, label='scatter point')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend(loc='upper left')
plt.title('Scatter plot')
plt.show()
```



In [ ]: