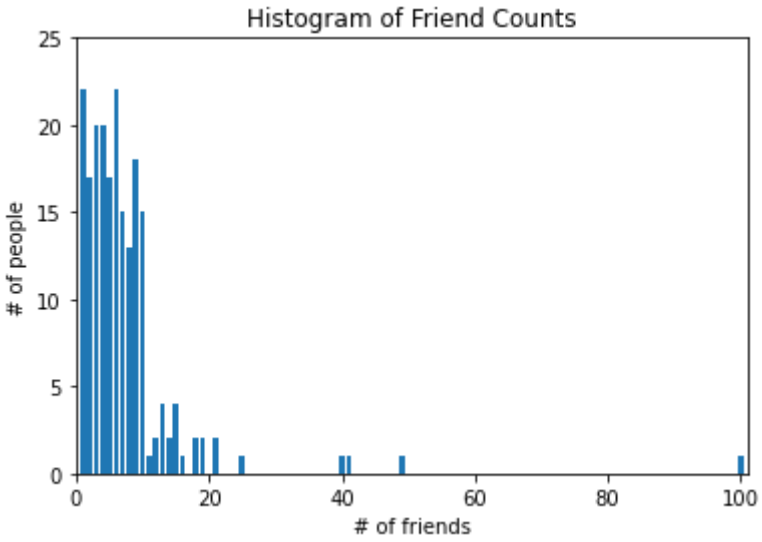
201600282 엄기산

In [46]:
```
def dot(v, w):
    return sum(v_i * w_i for v_i, w_i in zip(v, w))

def sum_of_squares(v):
    return dot(v, v)

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

from collections import Counter
import math
import numpy as np
```

In [47]:
```
num_friends = [100,49,41,40,25,21,21,19,19,18,18,16,15,15,15,15,14,14,13,13,13,13,12,

def make_friend_counts_histogram(plt):
    friend_counts = Counter(num_friends)
    xs = range(101)
    ys = [friend_counts[x] for x in xs]
    plt.bar(xs, ys)
    plt.axis([0,101,0,25])
    plt.title("Histogram of Friend Counts")
    plt.xlabel("# of friends")
    plt.ylabel("# of people")
    plt.show()

import matplotlib as plt
%pylab inline

make_friend_counts_histogram(plt)
```

```
Populating the interactive namespace from numpy and matplotlib
C:\Users\yongh\anaconda3\lib\site-packages\IPython\core\magics\pylab.py:159: UserWarni
ng: pylab import has clobbered these variables: ['dot', 'plt']
`%matplotlib` prevents importing * from pylab and numpy
  warn("pylab import has clobbered these variables: %s"  % clobbered +
```



In [48]:
```
num_points = len(num_friends)                    # 204
largest_value = max(num_friends)                 # 100
smallest_value = min(num_friends)                # 1
sorted_values = sorted(num_friends)
smallest_value = sorted_values[0]                # 1
```

```
second_smallest_value = sorted_values[1]    # 1
second_largest_value = sorted_values[-2]    # 49

print(num_points)
print(largest_value)
print(smallest_value)
print(sorted_values)
print(smallest_value)
print(second_smallest_value)
print(second_largest_value)
```

```
204
100
1
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5,
5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
8, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 10, 10, 10, 10, 10, 10, 1
0, 10, 10, 10, 10, 10, 10, 10, 10, 11, 12, 12, 13, 13, 13, 13, 14, 14, 15, 15, 15, 15,
16, 18, 18, 19, 19, 21, 21, 25, 40, 41, 49, 100]
1
1
49
```

In [49]:
```
def mean(x):
    return sum(x) / len(x)

mean(num_friends)
```

Out[49]: 7.333333333333333

In [50]:
```
np.mean(num_friends)
```

Out[50]: 7.333333333333333

In [51]:
```
def median(v):
    n = len(v)
    sorted_v = sorted(v)
    midpoint = n // 2

    if n % 2 == 1:
        # if odd, return the middle value
        return sorted_v[midpoint]
    else:
        lo = midpoint - 1
        hi = midpoint
        return (sorted_v[lo] + sorted_v[hi]) / 2

median(num_friends)
```

Out[51]: 6.0

In [52]:
```
np.median(num_friends)
```

Out[52]: 6.0

In [53]:
```
def quantile(x, p):
    p_index = int(p * len(x))
    return sorted(x)[p_index]

for i in range(0, 100, 25):
    print("%.2f Percentage value" % (i*0.01) , quantile(num_friends, i * 0.01))
```

```
0.00 Percentage value 1
0.25 Percentage value 3
0.50 Percentage value 6
0.75 Percentage value 9
```

In [54]:
```
np.percentile(num_friends, [i for i in range(0,100,25)])
```

Out[54]:
```
array([1., 3., 6., 9.])
```

In [55]:
```
def mode(x):
    counts = Counter(x)
    max_count = max(counts.values())
    return [x_i for x_i, count in counts.items()
            if count == max_count]

mode(num_friends)
```

Out[55]:
```
[6, 1]
```

In [56]:
```
def data_range(x):
    return max(x) - min(x)

data_range(num_friends)
```

Out[56]:
```
99
```

In [57]:
```
np.max(num_friends) - np.min(num_friends)
```

Out[57]:
```
99
```

In [58]:
```
def de_mean(x):
    x_bar = mean(x)
    return [x_i - x_bar for x_i in x]

def variance(x):
    n = len(x)
    deviations = de_mean(x)
    return sum_of_squares(deviations) / (n - 1)

variance(num_friends)
```

Out[58]:
```
81.54351395730707
```

In [59]:
```
np.var(num_friends)
```

Out[59]:
```
81.14379084967321
```

In [60]:
```
def standard_deviation(x):
    return math.sqrt(variance(x))

standard_deviation(num_friends)

np.std(num_friends, dtype=np.float64)

def interquartile_range(x):
    return quantile(x, 0.75) - quantile(x, 0.25)

interquartile_range(num_friends)
```

Out[60]:
```
9.030144736232474
```

Out[60]:
```
9.007984838446012
```

Out[60]: 6

In [61]:
```
daily_minutes = [1,68.77,51.25,52.08,38.36,44.54,57.13,51.4,41.42,31.22,34.76,54.01,38

def covariance(x, y):
    n = len(x)
    return dot(de_mean(x), de_mean(y)) / (n - 1)

covariance(num_friends, daily_minutes)
```

Out[61]: 22.425435139573075

In [62]:
```
np.cov(num_friends,daily_minutes)
```

Out[62]:
```
array([[ 81.54351396,  22.42543514],
       [ 22.42543514, 100.78589895]])
```

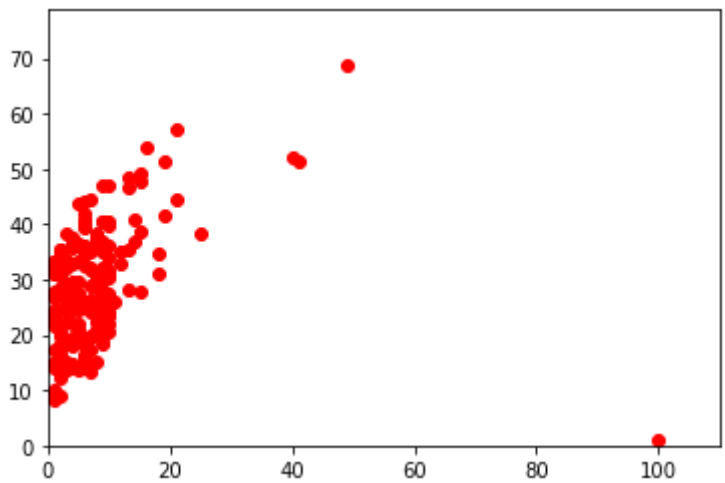
In [63]:
```
def correlation(x, y):
    stdev_x = standard_deviation(x)
    stdev_y = standard_deviation(y)
    if stdev_x > 0 and stdev_y > 0:
        return covariance(x, y) / stdev_x / stdev_y
    else:
        return 0 # if no variation, correlation is zero

correlation(num_friends, daily_minutes)

np.corrcoef(num_friends, daily_minutes)

plt.plot(num_friends, daily_minutes, 'ro')
plt.axis([0,max(num_friends)+10,0,max(daily_minutes) +10 ])
plt.show()
```

Out[63]: 0.24736957366478235

Out[63]:
```
array([[1.        , 0.24736957],
       [0.24736957, 1.        ]])
```

Out[63]: [<matplotlib.lines.Line2D at 0x271e5d448e0>]

Out[63]: (0.0, 110.0, 0.0, 78.77)

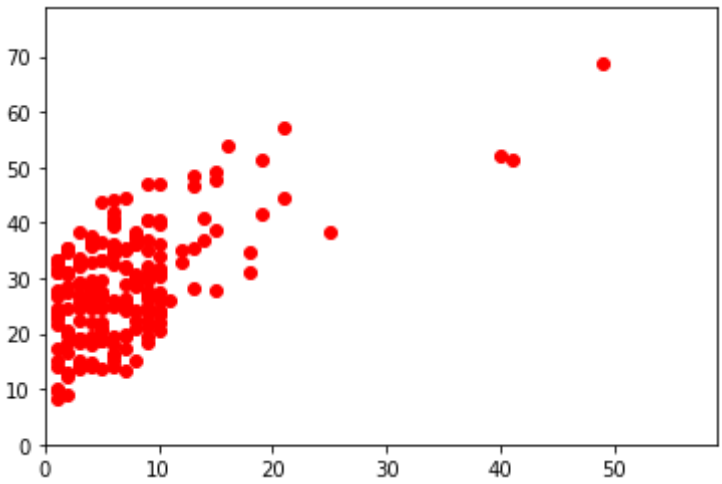


In [64]:
```
outlier = num_friends.index(100) # index of outlier

num_friends_good = [x
                    for i, x in enumerate(num_friends)
                    if i != outlier]
```

```
daily_minutes_good = [x
                      for i, x in enumerate(daily_minutes)
                      if i != outlier]

plt.plot(num_friends_good, daily_minutes_good, 'ro')
plt.axis([0,max(num_friends_good)+10,0,max(daily_minutes_good) +10 ])
plt.show()
```

Out[64]: [<matplotlib.lines.Line2D at 0x271e62151f0>]

Out[64]: (0.0, 59.0, 0.0, 78.77)



In [66]:
```
#LAB 6
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

import pandas as pd
from matplotlib import pyplot as plt

data = pd.read_csv('height-weight.csv')
Male = data[data['Gender']=="Male"]
Female = data[data['Gender']=="Female"]

Mh = Male.Height
Mw = Male.Weight
Fh = Female.Height
Fw = Female.Weight

plt.scatter(Mh,Mw, color='blue',label='Male')
plt.scatter(Fh,Fw, color='red',label='Female')
plt.xlabel("Height in inches")
plt.ylabel("Weight in Lbs")
plt.legend(loc=2)
plt.title("Height-Weigth Plotting by Eom Gi San")
plt.show()
```

Out[66]: <matplotlib.collections.PathCollection at 0x271e62c5370>

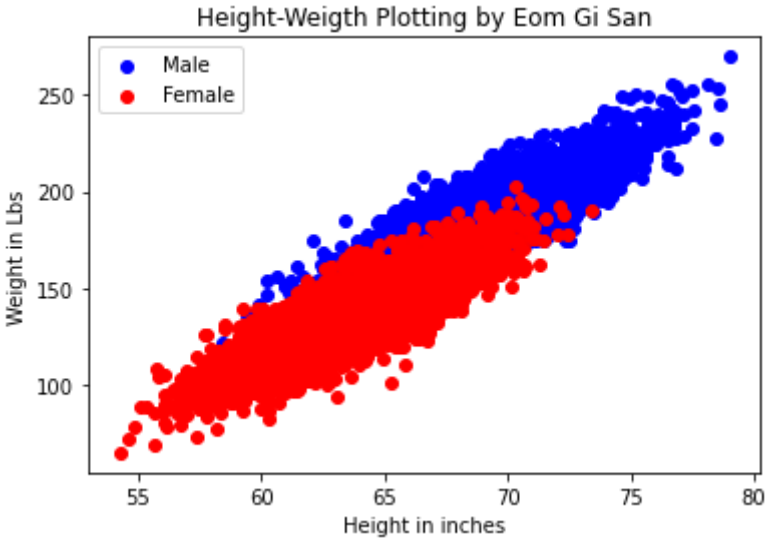
Out[66]: <matplotlib.collections.PathCollection at 0x271e62c5790>

Out[66]: Text(0.5, 0, 'Height in inches')

Out[66]: Text(0, 0.5, 'Weight in Lbs')

Out[66]: <matplotlib.legend.Legend at 0x271e62c5b50>

Out[66]: Text(0.5, 1.0, 'Height-Weigth Plotting by Eom Gi San')



In [108]:
```
#평균값
mean(Mh)
mean(Mw)
mean(Fh)
mean(Fw)
```

Out[108]: 69.02634590621741

Out[108]: 187.0206206581932

Out[108]: 63.70877360342507

Out[108]: 135.86009300746835

In [109]:
```
#중앙값
median(Mh)
median(Mw)
median(Fh)
median(Fw)
```

Out[109]: 69.02770850939555

Out[109]: 187.033546088862

Out[109]: 63.7309238591475

Out[109]: 136.11758297008498

In [110]:
```
#분위
np.percentile(Mh,[i for i in range(0,100,25)])
np.percentile(Mw,[i for i in range(0,100,25)])
np.percentile(Fh,[i for i in range(0,100,25)])
np.percentile(Fw,[i for i in range(0,100,25)])
```

Out[110]: array([58.40690493, 67.17467907, 69.02770851, 70.98874363])

Out[110]: array([112.90293945, 173.88776733, 187.03354609, 200.3578018])

Out[110]: array([54.26313333, 61.89444149, 63.73092386, 65.56356518])

Out[110]: array([64.70012671, 122.93409617, 136.11758297, 148.81092626])

In [111]:
```
#최빈값
mode(Mh.astype(int))
mode(Mw.astype(int))
```

```
mode(Fh.astype(int))
mode(Fw.astype(int))
```

Out[111]: [69]

Out[111]: [192]

Out[111]: [63]

Out[111]: [137]

In [112]:
```
#범위
data_range(Mh)
data_range(Mw)
data_range(Fh)
data_range(Fw)
```

Out[112]: 20.59183741463979

Out[112]: 157.086759057288

Out[112]: 19.126452540972608

Out[112]: 137.53708702680598

In [113]:
```
#분산
variance(Mh)
variance(Mw)
variance(Fh)
variance(Fw)
```

Out[113]: 8.198843252520467

Out[113]: 391.29407401608495

Out[113]: 7.269947493670126

Out[113]: 361.85428140439893

In [114]:
```
#표준편차
standard_deviation(Mh)
standard_deviation(Mw)
standard_deviation(Fh)
standard_deviation(Fw)
```

Out[114]: 2.863362228660647

Out[114]: 19.781154516763802

Out[114]: 2.696284015765054

Out[114]: 19.022467805319028

In [115]:
```
#공분산
covariance(Mh,Mw)
covariance(Mh,Fh)
covariance(Mh,Fw)

covariance(Mw,Fh)
covariance(Mw,Fw)

covariance(Fh,Fw)
```

Out[115]: 48.87964899179647

Out[115]: -0.2512910696943295

Out[115]: -1.097985039155029

Out[115]: -1.6854586795698363

Out[115]: -7.371512872924114

Out[115]: 43.57640416460329

In [116]:
```
#상관관계
correlation(Mh,Mw)
correlation(Mh,Fh)
correlation(Mh,Fw)

correlation(Mw,Fh)
correlation(Mw,Fw)

correlation(Fh,Fw)
```

Out[116]: 0.8629788486163176

Out[116]: -0.03254881082238815

Out[116]: -0.02015827070583475

Out[116]: -0.03160100083756695

Out[116]: -0.01959016686149972

Out[116]: 0.849608591418601

201600282 엄기산