In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from collections import Counter
from math import fabs
```

In [2]:
```
X = (69, 74, 68, 70, 72, 67, 66, 70, 76, 68, 72, 79, 74, 67, 66, 71, 74, 75, 75,

Y = (153, 175, 155, 135, 172, 150, 115, 137, 200, 130, 140, 265, 185, 112, 140,
```

Задание 1

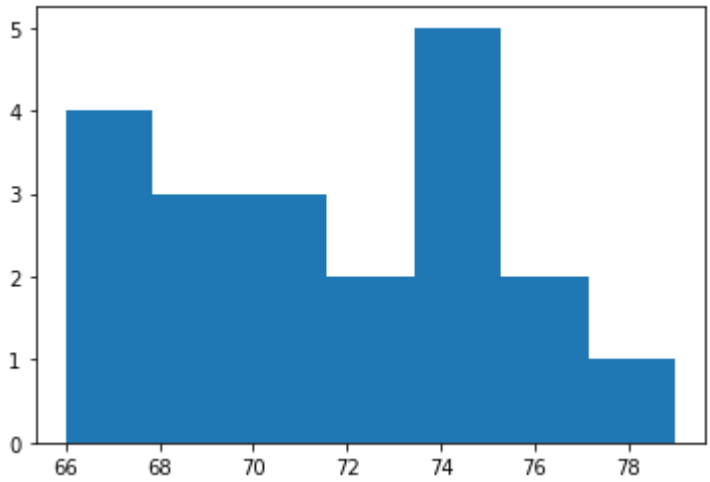
In [3]:
```
# A. Найти среднее, медиану и моду величины X

print(f'Средняя по X - {np.mean(X)}')
print(f'Медиана по X - {np.median(X)}')
print(f'Мода по X - {Counter(X).most_common(1)[0][0]}')
```

```
Средняя по X - 71.45
Медиана по X - 71.5
Мода по X - 74
```

In [4]:
```
# B. Найти дисперсию Y

print(f'Дисперсия по Y - {np.var(Y)}')

# other approach

y_mean = np.mean(Y)
D = sum([(y-y_mean)**2 for y in Y]) / len(Y)

print(f'Дисперсия по Y своими руками - {D}')
```

```
Дисперсия по Y - 1369.2099999999998
Дисперсия по Y своими руками - 1369.2099999999998
```

In [5]:
```
# C. Построить график нормального распределения для X

plt.hist(X, bins=7, density=False)
plt.show()

print(f'Распределение ненормальное')
```




```
Распределение ненормальное
```

In [6]:
```
# D. Найти вероятность того, что возраст больше 80

print(f"Вер-ть того, что возраст больше 80 - {Counter(X)[80] / len(X)}")
```

```
Вер-ть того, что возраст больше 80 - 0.0
```

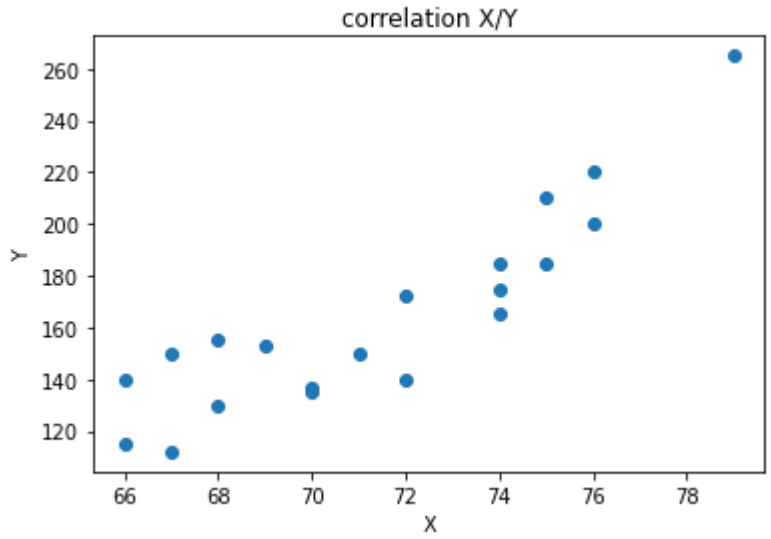
In [7]:
```
# E. Найти двумерное мат. ожидания и ковариационную матрицу
# для этих двух величин

print(f'Мат ожидание {np.mean([X, Y], axis=1)}')
print(f'Матрица ковариации: {np.cov(X, Y)}')
```

```
Мат ожидание [ 71.45 164.7 ]
Матрица ковариации: [[  14.57631579  128.87894737]
 [ 128.87894737 1441.27368421]]
```

In [8]:
```
# F. Определять корреляцию между X и Y

from math import sqrt


corr = np.corrcoef(X, Y)[1,0]
print(f'Корреляция между X и Y - {corr}')

# other approach

x_mean = np.mean(X)
y_mean = np.mean(Y)

my_corr = (
    sum([(X[i]-x_mean)*(Y[i]-y_mean) for i in range(len(X))])
    /
    sqrt(sum([(x-x_mean)**2 for x in X])*sum([(y-y_mean)**2 for y in Y]))
       )

print(f'Корреляция между X и Y подсчитанная руками - {my_corr}')
```

```
Корреляция между X и Y - 0.8891701351748049
Корреляция между X и Y подсчитанная руками - 0.8891701351748048
```

In [9]:
```
# G. Построить диаграмму рассеяния,
# отображающая зависимость между возрастом и весом

_, ax = plt.subplots()

ax.scatter(X, Y)
ax.set_xlabel("X")
ax.set_ylabel("Y")

ax.set_title('correlation X/Y')

plt.show()
```



Задание 2

In [10]:
```
indexes = ['a', 'b', 'c']
df = pd.DataFrame(
    {
      "x1": [17, 11, 11],
      "x2": [17, 9, 8],
      "x3": [12, 13, 19]
    },
      index=indexes
)
df
```

Out[10]:

	x1	x2	x3
a	17	17	12
b	11	9	13
c	11	8	19

In [11]:
```
print("Ковариационная матрица")
df.cov()
```

```
Ковариационная матрица
```

Out[11]:

	x1	x2	x3
x1	12.0	17.000000	-8.000000
x2	17.0	24.333333	-12.833333
x3	-8.0	-12.833333	14.333333

In [12]:
```
print("Обобщенная дисперсия (определитель ковариационной матрицы)")
np.linalg.det(df.cov())
```

```
Обобщенная дисперсия (определитель ковариационной матрицы)
```

Out[12]:
```
-1.565414464721469e-14
```

Задание 3

In [13]:
```
# Даны два одномерных нормальных распределения Na и Nb
# с мат. ожиданиями 4, 8 и СКО 1, 2 соответственно.
```

In [14]:
```
amount = 10000

na_mean, nb_mean = 4, 8
a_std, b_std = 1, 2

Na = np.random.normal(na_mean, a_std, amount)
Nb = np.random.normal(nb_mean, b_std, amount)
```

In [15]:
```
# A. Для каждого из значения {5,6,7} определите
# какое из распределений сгенерировало значение с большей вероятностью.

def check_(val):
    return ("Na" if fabs((val-na_mean)/a_std) < fabs((val-nb_mean)/b_std)
            else "Nb")

for val in [5, 6, 7]:
    print(f'Распределение {check_(val)} сгенирировало число {val} с большей вероя
```

```
Распределение Na сгенирировало число 5 с большей вероятностью
Распределение Nb сгенирировало число 6 с большей вероятностью
Распределение Nb сгенирировало число 7 с большей вероятностью
```

In [16]:
```
# B. Найди значение, которой могло быть сгенерировано
# обеими распределениями с равной вероятностью

print("Можно приравнять функции плотности распределений, их пересечение - решение

for indx in range(len(Na)):
    if fabs(Na[indx] - Nb[indx]) <= 0.001:
        print(f"Пересечение произошло в {Na[indx]}")
```

```
Можно приравнять функции плотности распределений, их пересечение - решение
Пересечение произошло в 5.327285305438631
```