In [3]:
```
import numpy as np
```

Task 2

Измерены значения IQ выборки студентов, обучающихся в местных технических вузах: 131, 125, 115, 122, 131, 115, 107, 99, 125, 111. Известно, что в генеральной совокупности IQ распределен нормально. Найдите доверительный интервал для математического ожидания с надежностью 0.95.

In [11]:
```
values = [131, 125, 115, 122, 131, 115, 107, 99, 125, 111]
```

In [12]:
```
M = np.mean(values)
print(M)
```

```
118.1
```

In [13]:
```
S = np.sqrt(sum([(x - M)**2 for x in values])/(len(values) - 1))
print(S)
```

```
10.54566788359614
```

из таблицы берем значение t-критерия, тк не знаем дисперсию генеральной совокупности

для alpha = 0.05 alpha/2 = 0.025 и d = n-1 = 9 (http://sixsigmaonline.ru/baza-znanij/37-1-0-210 (http://sixsigmaonline.ru/baza-znanij/37-1-0-210))

In [14]:
```
tcrit = 2.262
a = tcrit * S / np.sqrt(len(values))
```

In [15]:
```
print(f'доверительный интервал: [{M - a}, {M + a}]')
```

```
доверительный интервал: [110.55660776308164, 125.64339223691834]
```

Task 3

Известно, что рост футболистов в сборной распределен нормально с дисперсией генеральной совокупности, равной 25 кв.см. Объем выборки равен 27, среднее выборочное составляет 174.2. Найдите доверительный интервал для математического ожидания с надежностью 0.95.

In [16]:
```
D = 25
n = 27
M = 174.2
alpha = 1 - 0.95
```

используем Z критерий, тк дана дисперсия генеральной совокупности

из таблицы берем значение Z-критерия, тк знаем ср отклонение генеральной совокупности

для p = 0.975, тк нам нужен двусторонний случай p = 1 - alpha/2 (http://sixsigmaonline.ru/baza-znanij/37-1-0-36 (http://sixsigmaonline.ru/baza-znanij/37-1-0-36))

In [22]:
```
Z = 1.96
```

In [23]:
```
a = Z * np.sqrt(D) / np.sqrt(n)
```

In [24]:
```
print(f'доверительный интервал: [{M - a}, {M + a}]')
```

```
доверительный интервал: [172.31398912064722, 176.08601087935276]
```

In [ ]:
```

```

Task 1

In [33]:
```
zp = np.array([35, 45, 190, 200, 40, 70, 54, 150, 120, 110])
ks = np.array([401, 574, 874, 919, 459, 739, 653, 902, 746, 832])
```

вариант расчета 1

In [35]:
```
def otkl(x):
    return np.array([xi - np.mean(x) for xi in x])
```

In [37]:
```
def my_cov(x, y):
    return np.mean(otkl(ks) * otkl(zp))
```

In [39]:
```
my_cov(zp, ks)
```

Out[39]:
```
9157.84
```

вариант расчета 2

In [42]:
```
np.mean(zp * ks) - np.mean(zp)*np.mean(ks)
```

Out[42]:
```
9157.839999999997
```

In [47]:
```
np.cov(zp, ks, ddof=0)  # смещенная величина, при расчете матожиданий в знаменателе n-ddof, тк мы берем CC = 0 -> получаем смещенную величину
```

Out[47]:
```
array([[ 3494.64,  9157.84],
       [ 9157.84, 30468.89]])
```

In [50]:
```
print(f'{np.cov(zp, ks, ddof=0)[0][1]} - смещенная ковариация')
print(f'{np.cov(zp, ks, ddof=1)[0][1]} - несмещенная ковариация')
```

```
9157.84 - смещенная ковариация
10175.377777777776 - несмещенная ковариация
```

Коэффициент корреляции Пирсона

In [56]:
```
# смещенное стандартное отклонение
def get_s(values):
    return np.sqrt(sum([(x - np.mean(values))**2 for x in values])/(len(values)))
```

In [55]:
```
def my_corrcoeff(x, y):
    return my_cov(x, y)/(get_s(x) * get_s(y))
```

In [62]:
```
print(f'Коэффициент корреляции, полученный самописной функцией: {my_corr
coeff(zp, ks)}')
print(f'Коэффициент корреляции, полученный самописной функцией: {np.corr
coef(zp, ks)[0][1]}')
```

```
Коэффициент корреляции, полученный самописной функцией: 0.887490092073916
2
Коэффициент корреляции, полученный самописной функцией: 0.887490092073916
2
```

In [ ]:

In [ ]: