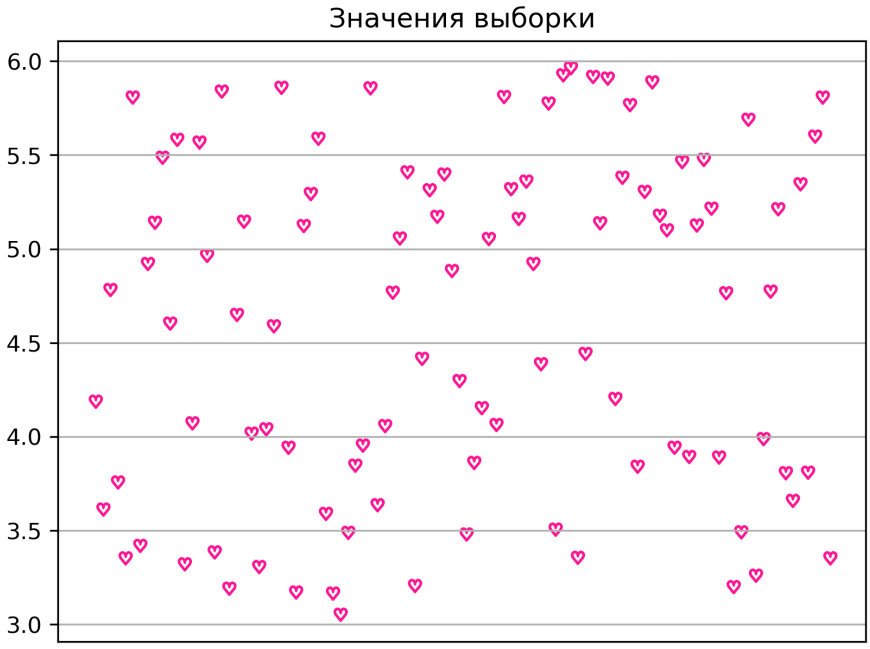
```
import numpy as np
import scipy.stats as sps
from scipy.special import erfinv
import matplotlib.pyplot as plt

from matplotlib import ticker
```

Генерация выборки

Параметр $\theta = 3$

```
theta = 3


class UniformGenerator(sps.rv_continuous):
        def _pdf(self, x):
            return 1 / theta

uniform = UniformGenerator(a=theta, b=2*theta, name='uniform')


sample = uniform.rvs(size=100)

X = np.arange(1, 101)
Y = sample

fig, ax = plt.subplots(dpi=300)

ax.yaxis.set_major_locator(ticker.MultipleLocator(0.5))
ax.set_xticks([])

ax.grid(True)

ax.scatter(X, Y, marker='$\heartsuit$', color='deeppink')
ax.set_title('Значения выборки')

plt.style.use("default")
plt.show()
```


Значения выборки
6.0
5.5
5.0
4.5
4.0
3.5
3.0

Точный доверительный интервал

```
x_max = np.max(sample)
n = len(sample)

betta_probability = 0.95

t1 = ((1 - betta_probability) / 2) ** (1 / n) + 1
t2 = ((1 + betta_probability) / 2) ** (1 / n) + 1

h1 = x_max / t2
h2 = x_max / t1

print('Точный доверительный интервал: [', h1, h2, ']. Длина: ', h2 - h1)
```

```
Точный доверительный интервал: [ 2.9969066980476993 3.0517902451905994 ]. Длина:  0.05488354714290011
```

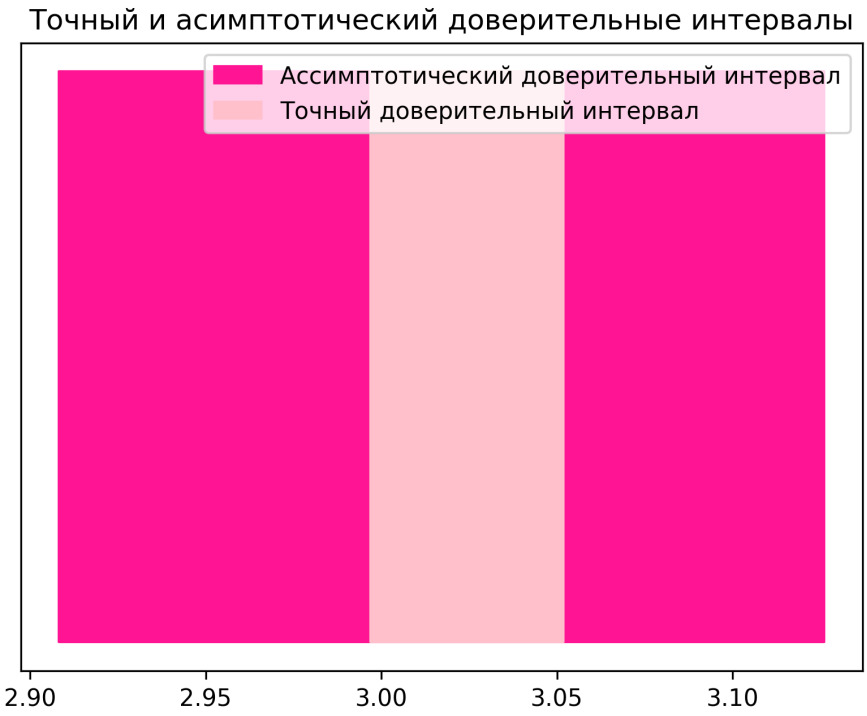
Асимптотический доверительный интервал

```
x_mean = np.mean(sample)
x_square_mean = np.mean(sample ** 2)

equal_part = np.sqrt(2) * erfinv(-betta_probability) * (2 / 3) * np.sqrt(x_square_mean - x_mean ** 2) / np.sqrt(n)
as_h2 = -equal_part + (2 / 3) * x_mean
as_h1 = equal_part + (2 / 3) * x_mean

print('Асимптотический доверительный интервал: [', as_h1, as_h2, ']. Длина: ', as_h2 - as_h1)
```

```
Асимптотический доверительный интервал: [ 2.9082288017477564 3.126115092420405 ]. Длина:  0.21788629067264864
```

Сравнение первых двух интервалов

```
fig1, ax1 = plt.subplots(dpi=300)

ax1.fill_betweenx([0, 1], [as_h1], [as_h2], color='deeppink', label='Ассимптотический доверительный интервал')
ax1.fill_betweenx([0, 1], [h1], [h2], color='pink', label='Точный доверительный интервал')
ax1.set_yticks([])

ax1.set_title("Точный и асимптотический доверительные интервалы")
ax1.legend()

plt.show()
```


Точный и асимптотический доверительные интервалы
Ассимптотический доверительный интервал
Точный доверительный интервал
2.90
2.95
3.00
3.05
3.10

Bootstrap

Параметрический bootstrap

```
parametric_theta = np.max(sample) / 2

class ParametricUniformGenerator(sps.rv_continuous):
    def _pdf(self, x):
        return 1 / parametric_theta

parametric_uniform = ParametricUniformGenerator(a=parametric_theta, b=2*parametric_theta, name='parametric_uniform')
N = 50000
parametric_samples = np.array([parametric_uniform.rvs(size=len(sample)) for _ in range(N)])

deltas = []
for parametric_sample in parametric_samples:
    deltas.append(np.mean(parametric_sample) * (2 / 3))
deltas = np.sort(deltas)
k1 = int((1 - betta_probability) * N / 2) - 1
k2 = int((1 + betta_probability) * N / 2) - 1

p_h1 = deltas[k1]
p_h2 = deltas[k2]

print('Доверительный интервал: [', p_h1, p_h2, '[ Длина:', p_h2 - p_h1)
```

```
Доверительный интервал: [ 2.8835865961320835 3.110086160470157 [ Длина: 0.2264995643380736
```

Непараметрический bootstrap

```
np_samples = np.array([np.random.choice(sample, 100) for _ in range(1000)])

theta_ = (2 / 3) * np.mean(sample)
np_thetas = []

for np_sample in np_samples:
    np_thetas.append((2 / 3) * np.mean(np_sample) - theta_)

np_thetas.sort()
np_h1 = -np_thetas[974] + theta_
np_h2 = -np_thetas[24] + theta_
print('Доверительный интервал: [', np_h1, np_h2, '[ Длина:', np_h2 - np_h1)
```

```
Доверительный интервал: [ 2.9069958112218526 3.12800613740365 [ Длина: 0.22101032618179728
```

Интервалы

```
fig2, ax2 = plt.subplots(dpi=200)

ax2.plot([as_h1, as_h2], [1, 1], color='pink', label='Ассимптотический', alpha=0.9)
ax2.plot([p_h1, p_h2], [2, 2], color='deeppink', label='ОММ + параметрический bootstrap', alpha=0.9)
ax2.plot([np_h1, np_h2], [3, 3], color='fuchsia',  label='ОММ + непараметрический bootstrap', alpha=0.9)
ax2.plot([h1, h2], [4, 4], color='peachpuff', label='Точный')

ax2.set_yticks([])

ax2.set_title("Точный и асимптотический доверительные интервалы")
ax2.legend(loc='lower left', bbox_to_anchor=(1, 0.75))

plt.show()
```