Как я использовала бутстрэп.

Подгружаем библиотеки и датафрейм diamonds:

```
In [85]: import numpy as np
         import pandas as pd
         import seaborn as sns

         #!pip install arch
```

```
In [86]: from arch.bootstrap import IIDBootstrap
         from arch.bootstrap import IndependentSamplesBootstrap
```

```
In [34]: df = pd.read_csv('C:/Users/Victoria/Downloads/diamonds.csv')
         df
```

Out[34]:

	size	color	clarity	cut	symmetry	polish	depth_percent	table_percent	meas_length	meas_width	meas_depth	total_sales_price
0	0.50	K	SI2	Excellent	Excellent	Excellent	61.4	55.0	5.10	5.12	3.14	990
1	0.50	E	VVS2	Excellent	Excellent	Very Good	61.9	60.0	5.06	5.09	3.14	3384
2	0.35	G	VS2	Excellent	Excellent	Excellent	63.0	55.0	4.47	4.51	2.83	1154
3	0.30	E	SI2	Excellent	Excellent	Excellent	63.2	57.0	4.24	4.27	2.69	886
4	0.30	F	VS2	Very Good	Very Good	Excellent	63.4	61.0	4.24	4.26	2.69	864
...	...	...	...	...	...	...	...	...	...	...	...	...
67593	0.30	D	SI2	Very Good	Very Good	Excellent	64.4	55.0	4.19	4.21	2.71	640
67594	0.60	H	VS2	Excellent	Excellent	Excellent	62.4	59.0	5.40	5.42	3.38	2932
67595	0.36	L	VVS2	Excellent	Excellent	Excellent	62.3	55.0	4.55	4.59	2.85	788
67596	0.41	J	SI1	Excellent	Excellent	Excellent	62.7	57.0	4.74	4.78	2.98	1074
67597	0.23	H	VVS1	Excellent	Excellent	Excellent	61.3	58.0	3.96	3.97	2.43	646

67598 rows × 12 columns

Нас интересует поле size – размеры бриллиантов. Сделаем из него массив.

```
In [77]: list1 = (df['size']).to_numpy()
         list1
```

```
Out[77]: array([0.5 , 0.5 , 0.35, ..., 0.36, 0.41, 0.23])
```

Мы хотим получить оценку для математического ожидания. Пусть E(Xi)=0, Var(Xi)=1, Xi - независимы. Получили среднее -0.00189185

```
In [91]: x=np.random.normal(0, 1, list1.shape)
         x
```

```
Out[91]: array([ 1.56376156, -0.70891361, -0.91189081, ..., -0.74927825,
                 1.15638622,  0.65349988])
```

```
In [92]: np.mean(x)
```

```
Out[92]: -0.0018918559886195675
```

Представим, что мы не знаем E(Xi). Точечная оценка=-0.00189185
“Размножим” ее, чтобы получить интервальную оценку=-0.001999

```
In [93]: p_e=np.mean(x)
         p_e
```

```
Out[93]: -0.0018918559886195675
```

```
In [107]: x_star = random.choices(x, k=len(x))
          x_star
```

```
Out[107]: [0.30284621649829635,
           -0.4213327307895218,
           0.22357500656536222,
           0.24721551598647348,
           -1.3026411167631822,
           0.9231188443327781,
           -0.28818069157758763,
           0.2816870805186203,
           -2.325209133249263,
           0.1125254787263098,
           -0.0995252689114555,
           0.06665180334238047,
           0.6019929569111866,
           -0.27461725315087054,
           -2.455716416324023,
           -1.7368289400367611,
           0.4712029110094115,
           -1.8644758274429094,
           0.3360862987005734,
```

```
In [108]: np.mean(x_star)
```

```
Out[108]: -0.0019991927707888185
```

С помощью наивного бутстрэпа построим доверительный интервал для μ - интервал покрывает значение p_e=-0.00189185

In [21]:
```
n_boot = 1000
mu_hat_star = [np.mean(random.choices(x, k=len(x))) for i in range(n_boot)]
```

In [22]:
```
[np.quantile(mu_hat_star, 0.025), np.quantile(mu_hat_star, 0.975)]
```

Out[22]:
```
[-0.009035970411242593, 0.006144639009483332]
```

5% интервал:

In [25]:
```
boot_x=IIDBootstrap(x)
boot_x.conf_int(np.mean, method='basic', reps=10000, size=0.95)
#интервал покрыл p_e
```

Out[25]:
```
array([[-0.00906919],
       [ 0.00610501]])
```

In [27]:
```
boot_xx = IndependentSamplesBootstrap(x)
boot_xx.conf_int(np.mean, reps=10000, size=0.95, method='basic')
```

Out[27]:
```
array([[-0.00922059],
       [ 0.00591375]])
```