Задание 1.
Исходные данные:
Рассмотрим данные со значениями площадей квартир в квадратных метрах (массив x1) и соответствующими им ценами на квартиры в тысячах долларов (массив y1), приведеннные для 12 наблюдений. По этим данным построим модель линейной регрессии.

Решение:

```
Python 3.8.10 (default, Jun  2 2021, 10:49:15)

[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> x1 = np.array([80, 90, 85, 115, 85, 85, 90, 80, 105, 110, 65, 100], dtype=np.float64)
>>> y1 = np.array([150, 160, 155, 175, 140, 150, 140, 155, 165, 190, 140, 165])
>>> b1 = np.cov(x1, y1, ddof=1)[0, 1] / np.var(x1, ddof=1)
>>> b0 = y1.mean() - b1 * x1.mean()
>>> print(b0, b1)
77.8996282527881 0.8717472118959109
>>>
>>> x1
array([ 80.,  90.,  85., 115.,  85.,  85.,  90.,  80., 105., 110.,  65.,
       100.])
>>>
>>> z1 = b0 + b1 * x1
>>> print(z1)
[147.6394052  156.35687732 151.99814126 178.15055762 151.99814126
 151.99814126 156.35687732 147.6394052  169.4330855  173.79182156
 134.56319703 165.07434944]
>>>
>>> e1 = y1 - z1
>>> print(e1)
[  2.3605948    3.64312268   3.00185874  -3.15055762 -11.99814126
  -1.99814126 -16.35687732   7.3605948   -4.4330855   16.20817844
   5.43680297  -0.07434944]
>>>
>>> e1.mean()
9.473903143468002e-15
>>>
```

Задание 2.
Исходные данные:
Рассмотрим данные о хоккеистах и попробуем построить модель регрессии, которая будет предсказывать возраст хоккеиста по его росту и весу.

Решение:

```
Python 3.8.10 (default, Jun  2 2021, 10:49:15)

[GCC 9.4.0] on linux

Type "help", "copyright", "credits" or "license" for more information.

>>> import numpy as np
```

```
>>> import pandas as pd
>>> df = pd.read_csv('hockey.csv')
>>> df.head()
   year country  no              name  ...                  club        age  cohort        bmi
0  2001     RUS  10     tverdovsky oleg  ...    anaheim mighty ducks   24.952772    1976  24.543462
1  2001     RUS   2   vichnevsky vitali  ...    anaheim mighty ducks   21.119781    1980  24.332277
2  2001     RUS  26  petrochinin evgeni  ...  severstal cherepovetal   25.229295    1976  28.680111
3  2001     RUS  28     zhdan alexander  ...           ak bars kazan   29.675565    1971  26.827421
4  2001     RUS  32     orekhovsky oleg  ...          dynamo moscow   23.490760    1977  28.734694

[5 rows x 13 columns]
>>>
>>> x2 = df[['height', 'weight']].values
>>> y2 = df['age'].values
>>> y2
array([24.95277207, 21.11978097, 25.229295  , ..., 26.82546201,
       26.01232033, 20.39698836])
>>>
>>> x2.shape
(6292, 2)
>>>
>>> ones = np.ones((x2.shape[0], 1))
>>> x2 = np.hstack((ones, x2))
>>> x2.shape
(6292, 3)
>>>
>>> x2
array([[  1., 185.,  84.],
       [  1., 188.,  86.],
       [  1., 182.,  95.],
       ...,
       [  1., 191.,  88.],
       [  1., 188.,  89.],
       [  1., 193.,  95.]])
>>> x2.T
array([[  1.,   1.,   1., ...,   1.,   1.,   1.],
```

```
       [185., 188., 182., ..., 191., 188., 193.],
       [ 84.,  86.,  95., ...,  88.,  89.,  95.]])
>>>
>>> xtx = x2.T.dot(x2)
>>> xtx_inv = np.linalg.inv(xtx)
>>> b = xtx_inv.dot(x2.T).dot(y2)
>>> print(b)
[49.70252952 -0.190563    0.1438651 ]
>>>
>>> z2 = x2.dot(b)
>>> e2 = y2 - z2
>>> e2
array([-1.5802704 , -5.1293027 , -3.45795261, ...,  0.86033715,
       -0.66835865, -6.19406621])
>>> e2.mean()
3.953738375051003e-12
>>>
>>> import sklearn
>>> from sklearn.linear_model import LinearRegression
>>> fit_intercept = False
>>> lr = LinearRegression(fit_intercept=False).fit(x2, y2)
>>> print(b, lr.coef_)
[49.70252952 -0.190563    0.1438651 ] [49.70252952 -0.190563    0.1438651 ]
>>>
```

Задание 3.

Исходные данные:

Посчитаем коэффициент детерминации для модели из предыдущих заданий. Для задания 1:

Решение:

```
Python 3.8.10 (default, Jun  2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> x1 = np.array([80, 90, 85, 115, 85, 85, 90, 80, 105, 110, 65, 100], dtype=np.float64)
>>> y1 = np.array([150, 160, 155, 175, 140, 150, 140, 155, 165, 190, 140, 165])
>>> b1 = np.cov(x1, y1, ddof=1)[0, 1] / np.var(x1, ddof=1)
```

```
>>> b0 = y1.mean() - b1 * x1.mean()
>>> print(b0, b1)
77.8996282527881 0.8717472118959109
>>>
>>> x1
array([ 80.,  90.,  85., 115.,  85.,  85.,  90.,  80., 105., 110.,  65.,
       100.])
>>>
>>> z1 = b0 + b1 * x1
>>> print(z1)
[147.6394052  156.35687732 151.99814126 178.15055762 151.99814126
 151.99814126 156.35687732 147.6394052  169.4330855  173.79182156
 134.56319703 165.07434944]
>>>
>>> e1 = y1 - z1
>>> print(e1)
[  2.3605948    3.64312268   3.00185874  -3.15055762 -11.99814126
  -1.99814126 -16.35687732   7.3605948   -4.4330855   16.20817844
   5.43680297  -0.07434944]
>>>
>>> e1.mean()
9.473903143468002e-15
>>>
>>> def sum_of_squares(samples: np.ndarray) -> float:
...     return ((samples - samples.mean()) ** 2).sum()
...
>>> r1 = 1 - sum_of_squares(e1) / sum_of_squares(y1)
>>> r1
0.6752261641274685
>>>
```

Также посчитаем коэффициент детерминации для задания 2:

```
Python 3.8.10 (default, Jun  2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
```

```
>>> import pandas as pd
>>> import sklearn
>>> df = pd.read_csv('hockey.csv')
>>> df.head()
   year country  no              name  ...                club        age  cohort        bmi
0  2001     RUS  10    tverdovsky oleg  ...  anaheim mighty ducks   24.952772    1976  24.543462
1  2001     RUS   2  vichnevsky vitali  ...  anaheim mighty ducks   21.119781    1980  24.332277
2  2001     RUS  26  petrochinin evgeni  ...  severstal cherepovetal   25.229295    1976  28.680111
3  2001     RUS  28    zhdan alexander  ...          ak bars kazan   29.675565    1971  26.827421
4  2001     RUS  32    orekhovsky oleg  ...          dynamo moscow   23.490760    1977  28.734694

[5 rows x 13 columns]
>>>
>>> x2 = df[['height', 'weight']].values
>>> y2 = df['age'].values
>>> y2
array([24.95277207, 21.11978097, 25.229295  , ..., 26.82546201,
       26.01232033, 20.39698836])
>>>
>>> x2.shape
(6292, 2)
>>>
>>> ones = np.ones((x2.shape[0], 1))
>>> x2 = np.hstack((ones, x2))
>>> x2.shape
(6292, 3)
>>>
>>> x2
array([[  1., 185.,  84.],
       [  1., 188.,  86.],
       [  1., 182.,  95.],
       ...,
       [  1., 191.,  88.],
       [  1., 188.,  89.],
       [  1., 193.,  95.]])
>>> x2.T
```

```
array([[  1.,   1.,   1., ...,   1.,   1.,   1.],
       [185., 188., 182., ..., 191., 188., 193.],
       [ 84.,  86.,  95., ...,  88.,  89.,  95.]])
>>>
>>> xtx = x2.T.dot(x2)
>>> xtx_inv = np.linalg.inv(xtx)
>>> b = xtx_inv.dot(x2.T).dot(y2)
>>> print(b)
[49.70252952 -0.190563    0.1438651 ]
>>>
>>> z2 = x2.dot(b)
>>> e2 = y2 - z2
>>> e2
array([-1.5802704 , -5.1293027 , -3.45795261, ...,  0.86033715,
       -0.66835865, -6.19406621])
>>> e2.mean()
3.953738375051003e-12
>>>
>>> from sklearn.linear_model import LinearRegression
>>> fit_intercept = False
>>> lr = LinearRegression(fit_intercept=False).fit(x2, y2)
>>> print(b, lr.coef_)
[49.70252952 -0.190563    0.1438651 ] [49.70252952 -0.190563    0.1438651 ]
>>>
>>> def sum_of_squares(samples: np.ndarray) -> float:
...     return ((samples - samples.mean()) ** 2).sum()
...
>>> r2 = 1 - sum_of_squares(e2) / sum_of_squares(y2)
>>> r2
0.03453384226670064
>>>
>>> print(z2.var(), y2.var())
0.6313664853957208 18.282543845515622
>>>
>>>
```

Задание 4.

Исходные данные:

Для данных из первого задания:

Решение:

```
Python 3.8.10 (default, Jun  2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> x1 = np.array([80, 90, 85, 115, 85, 85, 90, 80, 105, 110, 65, 100], dtype=np.float64)
>>> y1 = np.array([150, 160, 155, 175, 140, 150, 140, 155, 165, 190, 140, 165])
>>> b1 = np.cov(x1, y1, ddof=1)[0, 1] / np.var(x1, ddof=1)
>>> b0 = y1.mean() - b1 * x1.mean()
>>> print(b0, b1)
77.8996282527881 0.8717472118959109
>>>
>>> x1
array([ 80.,  90.,  85., 115.,  85.,  85.,  90.,  80., 105., 110.,  65.,
       100.])
>>>
>>> z1 = b0 + b1 * x1
>>> print(z1)
[147.6394052  156.35687732 151.99814126 178.15055762 151.99814126
 151.99814126 156.35687732 147.6394052  169.4330855  173.79182156
 134.56319703 165.07434944]
>>>
>>> e1 = y1 - z1
>>> print(e1)
[  2.3605948    3.64312268   3.00185874  -3.15055762 -11.99814126
  -1.99814126 -16.35687732   7.3605948   -4.4330855   16.20817844
   5.43680297  -0.07434944]
>>>
>>> e1.mean()
9.473903143468002e-15
>>>
>>> np.corrcoef(x1, y1) ** 2
array([[1.        , 0.67522616],
```

```
       [0.67522616, 1.        ]])
>>>
>>> np.corrcoef(y1, z1) ** 2
array([[1.        , 0.67522616],
       [0.67522616, 1.        ]])
>>>
>>>
```

Для второго задания:

```
Python 3.8.10 (default, Jun  2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> import pandas as pd
>>> import sklearn
>>> df = pd.read_csv('hockey.csv')
>>> df.head()
   year country  no                name  ...                  club        age  cohort        bmi
0  2001     RUS  10     tverdovsky oleg  ...    anaheim mighty ducks   24.952772    1976  24.543462
1  2001     RUS   2   vichnevsky vitali  ...    anaheim mighty ducks   21.119781    1980  24.332277
2  2001     RUS  26  petrochinin evgeni  ...  severstal cherepovetal   25.229295    1976  28.680111
3  2001     RUS  28     zhdan alexander  ...           ak bars kazan   29.675565    1971  26.827421
4  2001     RUS  32     orekhovsky oleg  ...           dynamo moscow   23.490760    1977  28.734694

[5 rows x 13 columns]
>>>
>>> x2 = df[['height', 'weight']].values
>>> y2 = df['age'].values
>>> y2
array([24.95277207, 21.11978097, 25.229295  , ..., 26.82546201,
       26.01232033, 20.39698836])
>>>
>>> x2.shape
(6292, 2)
>>>
>>> ones = np.ones((x2.shape[0], 1))
```

```
>>> x2 = np.hstack((ones, x2))
>>> x2.shape
(6292, 3)
>>>
>>> x2
array([[  1., 185.,  84.],
       [  1., 188.,  86.],
       [  1., 182.,  95.],
       ...,
       [  1., 191.,  88.],
       [  1., 188.,  89.],
       [  1., 193.,  95.]])
>>>
>>> x2.T
array([[  1.,   1.,   1., ...,   1.,   1.,   1.],
       [185., 188., 182., ..., 191., 188., 193.],
       [ 84.,  86.,  95., ...,  88.,  89.,  95.]])
>>>
>>> xtx = x2.T.dot(x2)
>>> xtx_inv = np.linalg.inv(xtx)
>>> b = xtx_inv.dot(x2.T).dot(y2)
>>> print(b)
[49.70252952 -0.190563    0.1438651 ]
>>>
>>> z2 = x2.dot(b)
>>> e2 = y2 - z2
>>> e2
array([-1.5802704 , -5.1293027 , -3.45795261, ...,  0.86033715,
       -0.66835865, -6.19406621])
>>>
>>> e2.mean()
3.953738375051003e-12
>>>
>>> from sklearn.linear_model import LinearRegression
>>> fit_intercept = False
>>> lr = LinearRegression(fit_intercept=False).fit(x2, y2)
```

```
>>> print(b, lr.coef_)
[49.70252952 -0.190563    0.1438651 ] [49.70252952 -0.190563    0.1438651 ]
>>>
>>> np.corrcoef(y2, z2) ** 2
array([[1.        , 0.03453384],
       [0.03453384, 1.        ]])
>>>
>>>
```

Задание 5.

Исходные данные:

Проверим значимость уравнений регрессии, построенных в первом и втором заданиях. В первом задании:

Решение:

```
Python 3.8.10 (default, Jun  2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
vik@vik-Z580:~$ command python3
Python 3.8.10 (default, Jun  2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> x1 = np.array([80, 90, 85, 115, 85, 85, 90, 80, 105, 110, 65, 100], dtype=np.float64)
>>> y1 = np.array([150, 160, 155, 175, 140, 150, 140, 155, 165, 190, 140, 165])
>>> b1 = np.cov(x1, y1, ddof=1)[0, 1] / np.var(x1, ddof=1)
>>> b0 = y1.mean() - b1 * x1.mean()
>>> print(b0, b1)
77.8996282527881 0.8717472118959109
>>>
>>> x1
array([ 80.,  90.,  85., 115.,  85.,  85.,  90.,  80., 105., 110.,  65.,
       100.])
>>>
>>> z1 = b0 + b1 * x1
>>> print(z1)
```

```
[147.6394052  156.35687732 151.99814126 178.15055762 151.99814126
 151.99814126 156.35687732 147.6394052  169.4330855  173.79182156
 134.56319703 165.07434944]
>>>
>>> e1 = y1 - z1
>>> print(e1)
[  2.3605948    3.64312268   3.00185874  -3.15055762 -11.99814126
  -1.99814126 -16.35687732   7.3605948   -4.4330855   16.20817844
   5.43680297  -0.07434944]
>>>
>>> e1.mean()
9.473903143468002e-15
>>>
>>> n = x1.shape[0]
>>> m = 1
>>> k1 = m
>>> k2 = n - m - 1
>>> print(k1, k2)
1 10
>>> alpha = 0.05
>>> import scipy
>>> from scipy import stats
>>> t = stats.f.ppf(1 - alpha, k1, k2)
>>> print(t)
4.9646027437307145
>>>
>>> def sum_of_squares(samples: np.ndarray) -> float:
...     return ((samples - samples.mean()) ** 2).sum()
...
>>> r1 = sum_of_squares(e1) / sum_of_squares(y1)
>>> r1
0.3247738358725315
>>>
>>> f = (r1 / k1) / ((1 - r1) / k2)
>>> print(f)
4.809852655698055
```

```
>>>
```

Теперь проводим аналогичные расчёты для второго задания.

```
Python 3.8.10 (default, Jun  2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> import pandas as pd
>>> import sklearn
>>> df = pd.read_csv('hockey.csv')
>>> df.head()
   year country  no              name  ...                  club        age  cohort        bmi
0  2001     RUS  10    tverdovsky oleg  ...   anaheim mighty ducks   24.952772    1976  24.543462
1  2001     RUS   2  vichnevsky vitali  ...   anaheim mighty ducks   21.119781    1980  24.332277
2  2001     RUS  26  petrochinin evgeni  ...  severstal cherepovetal   25.229295    1976  28.680111
3  2001     RUS  28    zhdan alexander  ...          ak bars kazan   29.675565    1971  26.827421
4  2001     RUS  32    orekhovsky oleg  ...          dynamo moscow   23.490760    1977  28.734694

[5 rows x 13 columns]
>>> x2 = df[['height', 'weight']].values
>>> y2 = df['age'].values
>>> y2
array([24.95277207, 21.11978097, 25.229295  , ..., 26.82546201,
       26.01232033, 20.39698836])
>>> x2.shape
(6292, 2)
>>>
>>> ones = np.ones((x2.shape[0], 1))
>>> x2 = np.hstack((ones, x2))
>>> x2.shape
(6292, 3)
>>>
>>> x2
array([[  1., 185.,  84.],
       [  1., 188.,  86.],
       [  1., 182.,  95.],
       ...,
```

```
       [  1., 191.,  88.],
       [  1., 188.,  89.],
       [  1., 193.,  95.]])
>>> x2.T
array([[  1.,   1.,   1., ...,   1.,   1.,   1.],
       [185., 188., 182., ..., 191., 188., 193.],
       [ 84.,  86.,  95., ...,  88.,  89.,  95.]])
>>>
>>> xtx = x2.T.dot(x2)
>>> xtx_inv = np.linalg.inv(xtx)
>>> b = xtx_inv.dot(x2.T).dot(y2)
>>> print(b)
[49.70252952 -0.190563    0.1438651 ]
>>>
>>> z2 = x2.dot(b)
>>> e2 = y2 - z2
>>> e2
array([-1.5802704 , -5.1293027 , -3.45795261, ...,  0.86033715,
       -0.66835865, -6.19406621])
>>>
>>> e2.mean()
3.953738375051003e-12
>>>
>>> from sklearn.linear_model import LinearRegression
>>> fit_intercept = False
>>> lr = LinearRegression(fit_intercept=False).fit(x2, y2)
>>> print(b, lr.coef_)
[49.70252952 -0.190563    0.1438651 ] [49.70252952 -0.190563    0.1438651 ]
>>>
>>> def sum_of_squares(samples: np.ndarray) -> float:
...     return ((samples - samples.mean()) ** 2).sum()
...
>>> r2 = 1 - sum_of_squares(e2) / sum_of_squares(y2)
>>> r2
0.03453384226670064
>>>
```

```
>>> print(z2.var(), y2.var())
0.6313664853957208 18.282543845515622
>>>
>>> n = x2.shape[0]
>>> m = x2.shape[1]
>>> m = x2.shape[1] - 1
>>> k1 = m
>>> k2 = n - m -1
>>> print(k1, k2)
2 6289
>>>
>>> alpha = 0.05
>>> import scipy
>>> from scipy import stats
>>> t = stats.f.ppf(1 - alpha, k1, k2)
>>> t
2.9971597282399225
>>>
>>> f = (r2 / k1) / ((1 - r2) / k2)
>>> print(f)
112.475891710787
>>>
```

Задание 6.

Исходные данные:

В задании 1 мы получили модель парной регрессии с коэффициентами:

Решение:

```
Python 3.8.10 (default, Jun  2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> x1 = np.array([80, 90, 85, 115, 85, 85, 90, 80, 105, 110, 65, 100], dtype=np.float64)
>>> y1 = np.array([150, 160, 155, 175, 140, 150, 140, 155, 165, 190, 140, 165])
>>> b1 = np.cov(x1, y1, ddof=1)[0, 1] / np.var(x1, ddof=1)
>>> b0 = y1.mean() - b1 * x1.mean()
>>> print(b0, b1)
```

```
77.8996282527881 0.8717472118959109
>>>
>>> x1
array([ 80.,  90.,  85., 115.,  85.,  85.,  90.,  80., 105., 110.,  65.,
       100.])
>>>
>>> z1 = b0 + b1 * x1
>>> print(z1)
[147.6394052  156.35687732 151.99814126 178.15055762 151.99814126
 151.99814126 156.35687732 147.6394052  169.4330855  173.79182156
 134.56319703 165.07434944]
>>>
>>> e1 = y1 - z1
>>> print(e1)
[  2.3605948    3.64312268   3.00185874  -3.15055762 -11.99814126
  -1.99814126 -16.35687732   7.3605948   -4.4330855   16.20817844
   5.43680297  -0.07434944]
>>>
>>> e1.mean()
9.473903143468002e-15
>>>
>>> def standard_error_slope(
...         x: np.ndarray,
...         y: np.ndarray,
...         z: np.ndarray,
... ) -> float:
...     n = x.shape[0]
...     upper = ((y - z) ** 2).sum() / (n - 2)
...     lower = ((x - x.mean()) ** 2).sum()
...     return np.sqrt(upper / lower)
...
>>> s_slope = standard_error_slope(x1, y1, z1)
>>> s_slope
0.19118616125822915
>>>
>>> alpha = 0.05
```

```
>>> n = x1.shape[0]
>>> import scipy
>>> from scipy import stats
>>> t1 = stats.t.ppf(alpha / 2, df=n - 2)
>>> t2 = stats.t.ppf(1 - alpha / 2, df=n - 2)
>>> print(t1, t2)
-2.2281388519649385 2.2281388519649385
>>>
>>> b1_lower, b1_upper = (b1 + t1 * s_slope, b1 + t2 * s_slope)
>>> b1_lower, b1_upper
(0.44575789803841653, 1.2977365257534053)
>>>
>>>
```

Задание 7.

Исходные данные:

Независимым образом получены две выборки из роста людей:

Решение:

```
Python 3.8.10 (default, Jun  2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> x1 = np.array([169.6, 178.6, 175.3, 171.8, 169.8, 164.1, 179.0, 162.9, 179.5, 169.1, 173.7, 168.7, 182.9, 176.3, 156.9, 174.2, 187.2, 178.5])
>>> x2 = np.array([180.3, 179.4, 178.3, 168.8, 151.4, 168.1, 169.1, 150.0, 156.3, 176.3, 163.5, 169.8, 177.5, 168.0, 162.4, 167.3, 176.4, 166.1, 164.9, 163.4, 163.2, 169.6, 160.9, 170.8])
>>> n1 = x1.size
>>> n2 = x2.size
>>> s1 = x1.std(ddof=1)
>>> s2 = x2.std(ddof=1)
>>> s_delta = np.sqrt(s1 ** 2 / n1 + s2 ** 2 / n2)
>>> s_delta
2.4248215967971274
>>>
>>> t = (x1.mean() - x2.mean()) / s_delta
>>> t
2.331213886103755
```

```
>>>
>>> df = (s1 ** 2 / n1 + s2 ** 2 / n2) ** 2 / ((s1 ** 2 / n1) ** 2 / (n1 - 1) + (s2 ** 2 / n2) ** 2 / (n2 - 1))
>>> df
38.264950672414635
>>>
>>> alpha = 0.05
>>> import scipy
>>> from scipy import stats
>>> t1 = stats.t.ppf(alpha / 2, df=df)
>>> t2 = stats.t.ppf(1 - alpha / 2, df=df)
>>> print(t1, t2)
-2.0239339487009755 2.023933948700975
>>>
>>> import scipy
>>> from scipy.stats import ttest_ind
>>> equal_var = False
>>> stats.ttest_ind(x1, x2, equal_var=False)
Ttest_indResult(statistic=2.331213886103755, pvalue=0.025107534360731)
>>>
>>>
```