



# Машинное обучение

---

НИЯУ МИФИ, КАФЕДРА ФИНАНСОВОГО МОНИТОРИНГА

КУРС ЛЕКЦИЙ

В.Ю. РАДЫГИН. ЛЕКЦИЯ 3

# Библиотеки

---

В данной лекции будут рассмотрены примеры с использованием следующих библиотек:

- NumPy – <https://numpy.org/>
- SciPy – <https://scipy.org/>
- scikit-learn – <https://scikit-learn.org>
- Statsmodels – <https://www.statsmodels.org/stable/index.html>
- Matplotlib – <https://matplotlib.org/>
- Pandas – <https://pandas.pydata.org/>

Также будет неявно использоваться

- Xlrd – <https://pypi.org/project/xlrd/>

# Часть 1

---

ИСПОЛЬЗОВАНИЕ МАТЕМАТИЧЕСКИХ БИБЛИОТЕК ЯЗЫКА PYTHON ДЛЯ ИЗУЧЕНИЯ СТЕПЕНИ БЛИЗОСТИ (ТЕСНОТЫ ВЗАИМОСВЯЗИ) ВЫБОРОК

# Анализ тесноты взаимосвязи

---

Одним из важнейших факторов понимания того или иного процесса является понимание тесноты взаимосвязи между параметрами, характеризующими данный процесс.

Параметры могут быть независимыми, или связанными. Причём связь может быть либо функциональной, либо стохастической.

При стохастической зависимости каждому значению одного параметра соответствует множество допустимых значений другого параметра.

Виды взаимосвязи и способы их анализа сильно отличаются в зависимости от типов шкал, в которых представлены исследуемые параметры.

# Типы измерительных шкал

---

Различают следующие типы измерительных шкал:

- 1) шкала отношений (ratio scale);
- 2) интервальная шкала (interval scale);
- 3) порядковая шкала (ordinal scale);
- 4) номинальная шкала (nominal scale).

# Шкала отношений

---

Для параметров, шкала измерений которых является шкалой отношений, имеет смысл процентного соотношения измерений. Например, возьмём цену доллара на по данным MOEX:

07.09.2018 1USD = 69,4925RUR;

04.04.2018 1USD = 57,4450RUR.

Для данных наблюдений имеет смысл отношение  $69,4925/57,4450$ , то есть фраза «7 сентября доллар в 1,21 раза или на 21% дороже» имеет смысл.

# Интервальная шкала

---

Для параметров, шкала измерений которых является интервальной, имеет смысл разница между двумя наблюдениями, но не имеет смысл отношение двух наблюдений. Например, возьмём температуру в Москве и Вашингтоне на 19:00 07.09.2018:

Москва – 20°C (68°F);

Вашигнтон – 30°C (86°F).

Для данных наблюдений имеет смысл фраза «В Вашингтоне на 10°C теплее». Но фраза «В Вашингтоне на 50% теплее, чем в Москве» бессмысленна.

# Порядковая шкала

---

Для параметров, шкала измерений которых является порядковой, разница между двумя наблюдениями не имеет смысла. Тем не менее, наблюдения можно ранжировать по порядку. Например, средний балл оценки студентами преподавателя:

В.Ю. Радыгин – 2,73;

Другой преподаватель – 2,81.

Для данных наблюдений имеет смысл фраза «Оценка другого преподавателя больше оценки В.Ю. Радыгина», но бессмысленна фраза «Другой преподаватель на 0,08 лучше В.Ю. Радыгина».



# Номинальная шкала

---

Для параметров, шкала измерений которых является номинальной, никакие сравнения с помощью логических или арифметических операций не имеют смысла.

Пример 1: пол – женский и мужской.

Пример 2: марка любимого автомобиля.

Пример 3: номер телефона.

# Типы измерительных шкал

---

Шкала	Имеют ли смысл мат. ожид. и станд. откл.	Другое название
Отношений	Да	Количественная
Интервальная	Да	Количественная
Порядковая	Нет	Качественная
Номинальная	Нет	Качественная

# Корреляционная зависимость

---

Частным случаем стохастической зависимости является корреляционная зависимость — зависимость между случайными величинами, при которой наблюдается функциональная зависимость между значениями величины одного параметра и средними значениями другого параметра.

Очевидно, что для номинальных шкал говорить о наличии корреляционной зависимости нельзя.

Основными свойствами корреляционной зависимости являются теснота и количество признаков (параметров).

Для парной корреляционной связи различают линейную и нелинейную связи.

# Оценка тесноты зависимости

---

Наиболее простыми и популярными средствами для оценки тесноты (силы) линейной связи служат коэффициент выборочной ковариации и коэффициент корреляции Пирсона.

$$\overline{cov}(X, Y) = \frac{1}{n-1} \sum_i (x_i - \bar{x})(y_i - \bar{y})$$

$$r = \frac{\overline{cov}(X, Y)}{s_x s_y}$$

$s_x s_y$  – стандартные отклонения

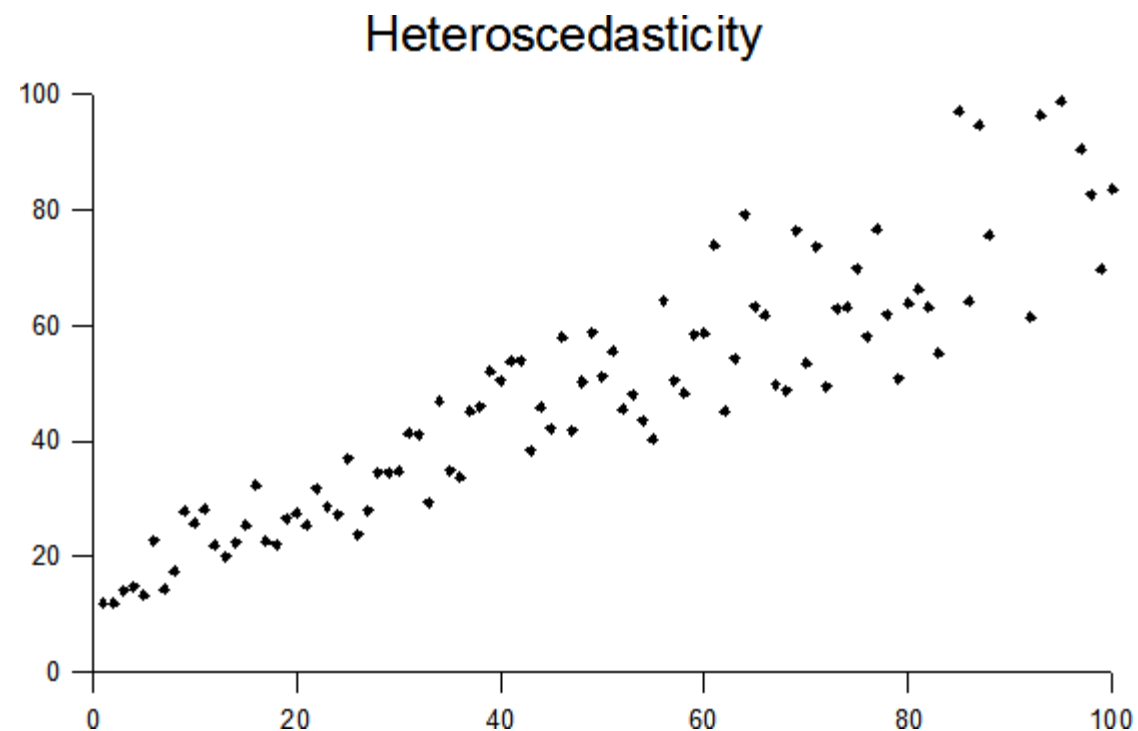
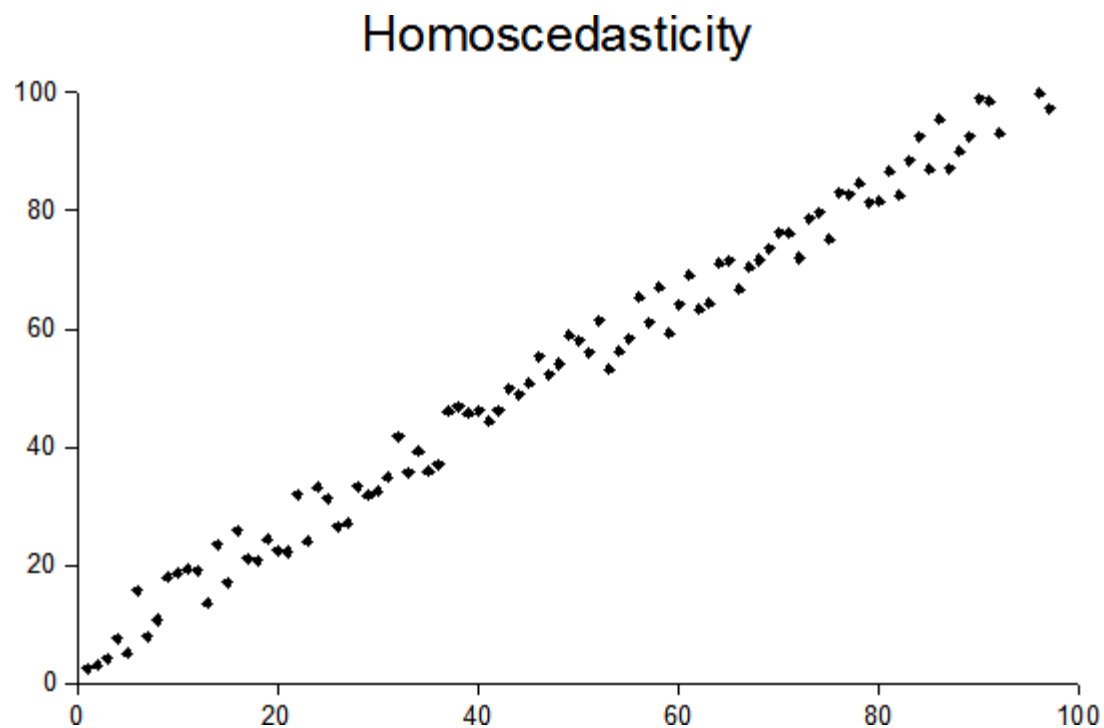
# Ограничения метода Пирсона

---

Использование коэффициента корреляции Пирсона имеет ряд строгих ограничений, в том числе:

- 1) оба параметра должны быть количественными и непрерывными;
- 2) как минимум один из параметров (а лучше оба) должен иметь нормальное распределение;
- 3) зависимость между параметрами должны носить линейный характер;
- 4) гомоскедастичность параметров (вариабельность одной переменной не зависит от значений другой переменной);
- 5) число наблюдений не менее 25.

# Гомоскедастичность



Изображения взяты с Википедии:

<https://ru.wikipedia.org/wiki/%D0%93%D0%BE%D0%BC%D0%BE%D1%81%D0%BA%D0%B5%D0%B4%D0%B0%D1%81%D1%82%D0%B8%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%8C>

# Пример

---

Для демонстрации примеров данной лекции используем информацию о стоимости валют, взятую с сайта Центробанка РФ ([https://cbr.ru/currency\\_base/dynamics](https://cbr.ru/currency_base/dynamics)), и информацию о цене нефти марки Brent, взятую с сайта Финам (<https://www.finam.ru/profile/tovary/brent/export>). Данные сложим в один Excel-файл (название файла, для однозначности, пусть будет exp.xlsx).

```
*Untitled*
File Edit Format Run Options Window Help
import numpy as np
import pandas as pd

pd.set_option('display.max_columns', 2000)
pd.set_option('display.width', 2000)

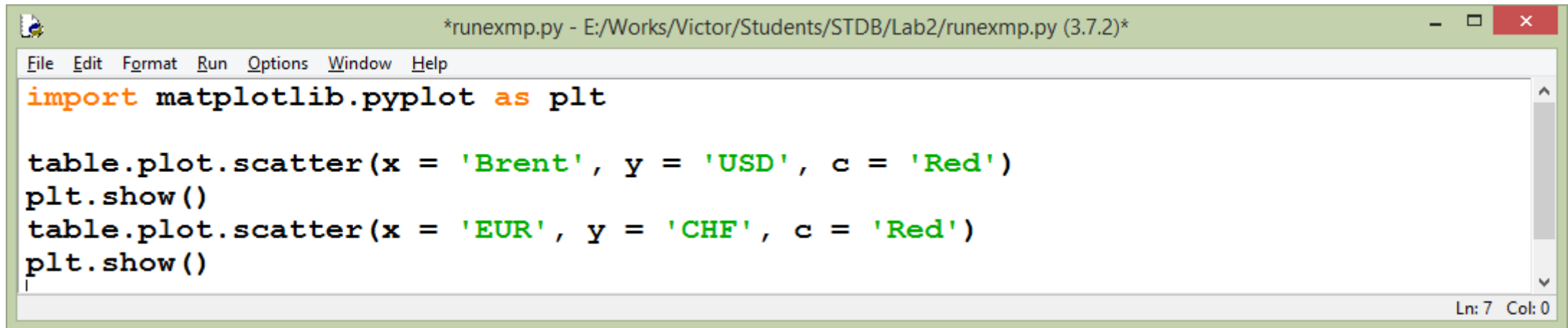
table = pd.read_excel("exp.xlsx")
print(table)|
```

```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:/Works/Victor/Students/STDB/Lab2/runexmp.py =====
      EUR      USD      CHF  Brent
0    76.0540  65.7508  66.3881  81.56
1    75.9236  65.5305  66.3063  80.36
2    75.6512  65.4026  65.9766  79.35
3    75.5692  65.7238  66.1471  80.02
4    75.3702  65.3065  65.5293  76.24
5    74.8584  65.3101  65.5526  75.68
6    75.0806  65.6299  65.8472  76.70
7    74.9851  65.7476  65.8596  77.82
8    75.0399  65.8129  65.8590  76.18
9    74.7918  65.7742  65.5644  74.59
10   76.0926  67.5238  66.8420  65.35
11   76.0737  67.6812  66.9514  65.86
12   76.7556  67.9975  67.4779  66.63
13   75.5358  66.6159  66.3241  67.10
14   75.3218  66.0081  66.0543  62.44
15   75.1825  65.5871  66.1604  62.46
```



# Графический анализ

График рассеяния (Scatter Plot) – это простейший метод визуального анализа степени и характера взаимосвязи между двумя параметрами.

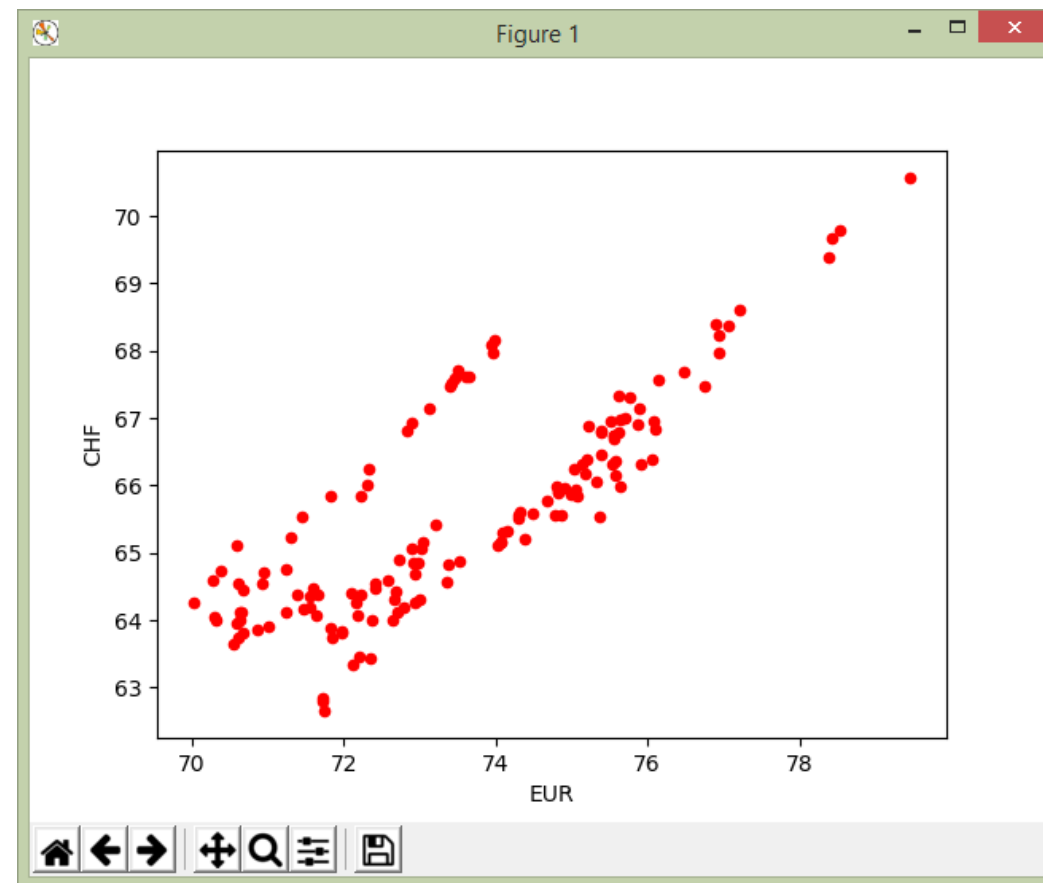
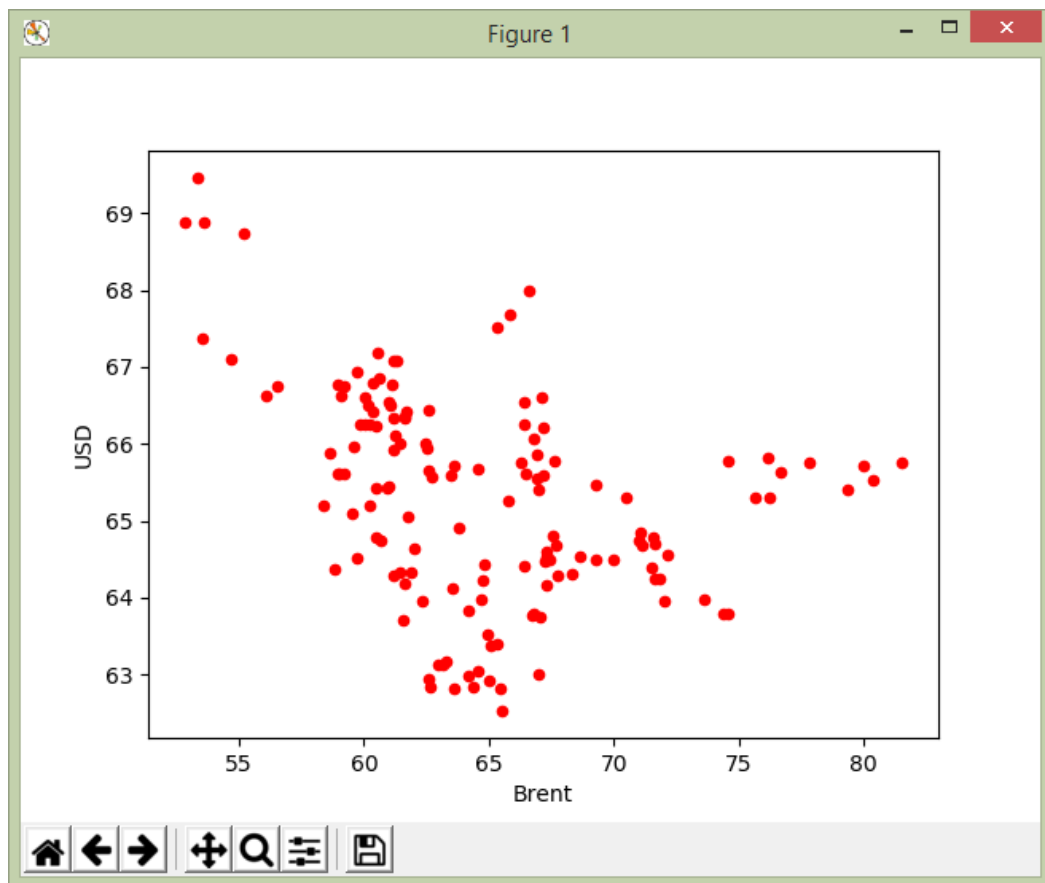
A screenshot of a Python IDE window titled '\*runexmp.py - E:/Works/Victor/Students/STDB/Lab2/runexmp.py (3.7.2)\*'. The window has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code editor contains the following Python code:

```
import matplotlib.pyplot as plt

table.plot.scatter(x = 'Brent', y = 'USD', c = 'Red')
plt.show()
table.plot.scatter(x = 'EUR', y = 'CHF', c = 'Red')
plt.show()
```

The status bar at the bottom right indicates 'Ln: 7 Col: 0'.

# Результат



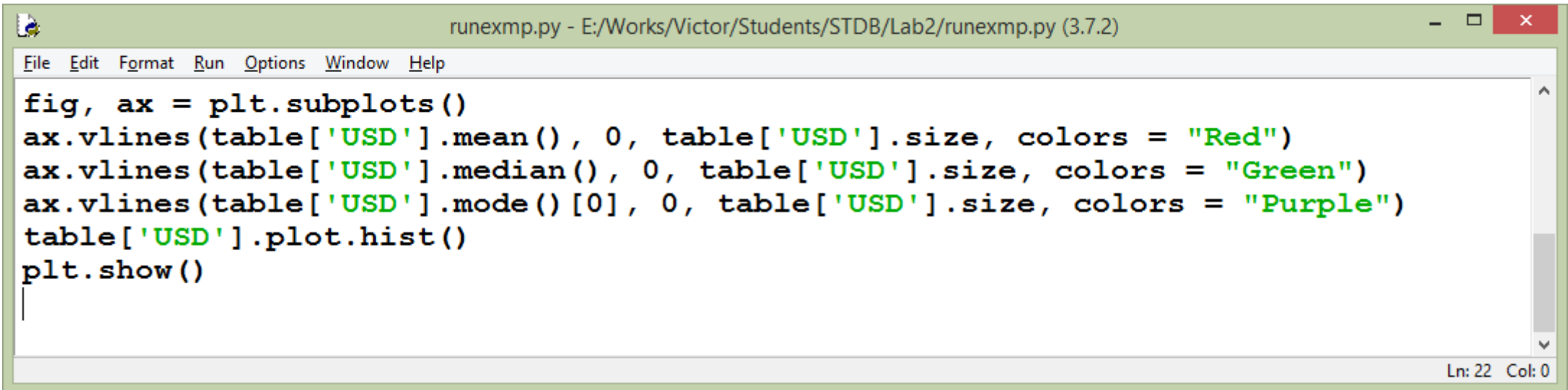
# Нормально распределённые совокупности

---

Не для всех методов выявления взаимосвязи между параметрами подходят произвольные совокупности наблюдений. Некоторые требуют, что совокупность была нормально распределённой. Для проверки данного факта требуется выполнение:

- максимальной близости или равенства значений средней арифметической, моды и медианы;
- соблюдения правила «трёх сигм» (в интервале  $M \pm 1\sigma$  находятся не менее 68,3% значений, в интервале  $M \pm 2\sigma$  – не менее 95,5% значений, в интервале  $M \pm 3\sigma$  находятся не менее 99,7% значений);
- требования измерения параметров в количественной шкале;
- положительных результатов проверки на нормальность распределения при помощи специальных критериев: Жака-Бера, Колмогорова-Смирнова, Шапиро-Уилка и Андерсона-Дарлинга.

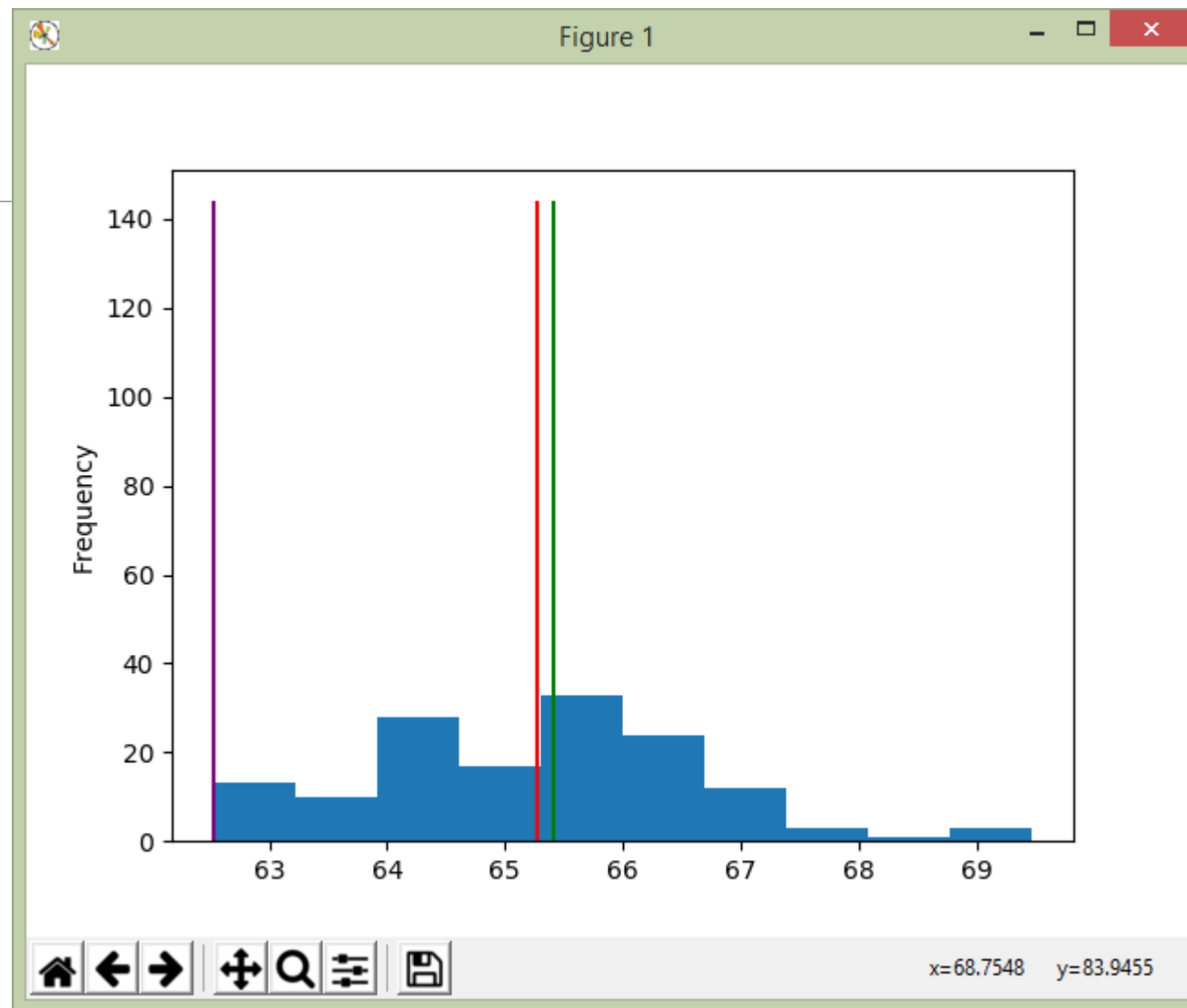
# Математическое ожидание, мода, медиана



```
runexmp.py - E:/Works/Victor/Students/STDB/Lab2/runexmp.py (3.7.2)
File Edit Format Run Options Window Help
fig, ax = plt.subplots()
ax.vlines(table['USD'].mean(), 0, table['USD'].size, colors = "Red")
ax.vlines(table['USD'].median(), 0, table['USD'].size, colors = "Green")
ax.vlines(table['USD'].mode()[0], 0, table['USD'].size, colors = "Purple")
table['USD'].plot.hist()
plt.show()
Ln: 22 Col: 0
```

# Результат

Что не так с модой (фиолетовый цвет)?



# Графический анализ

---

Для графического анализа выборки на соответствие нормальному распределению обычно используют вероятностные графики (Probability Plot).

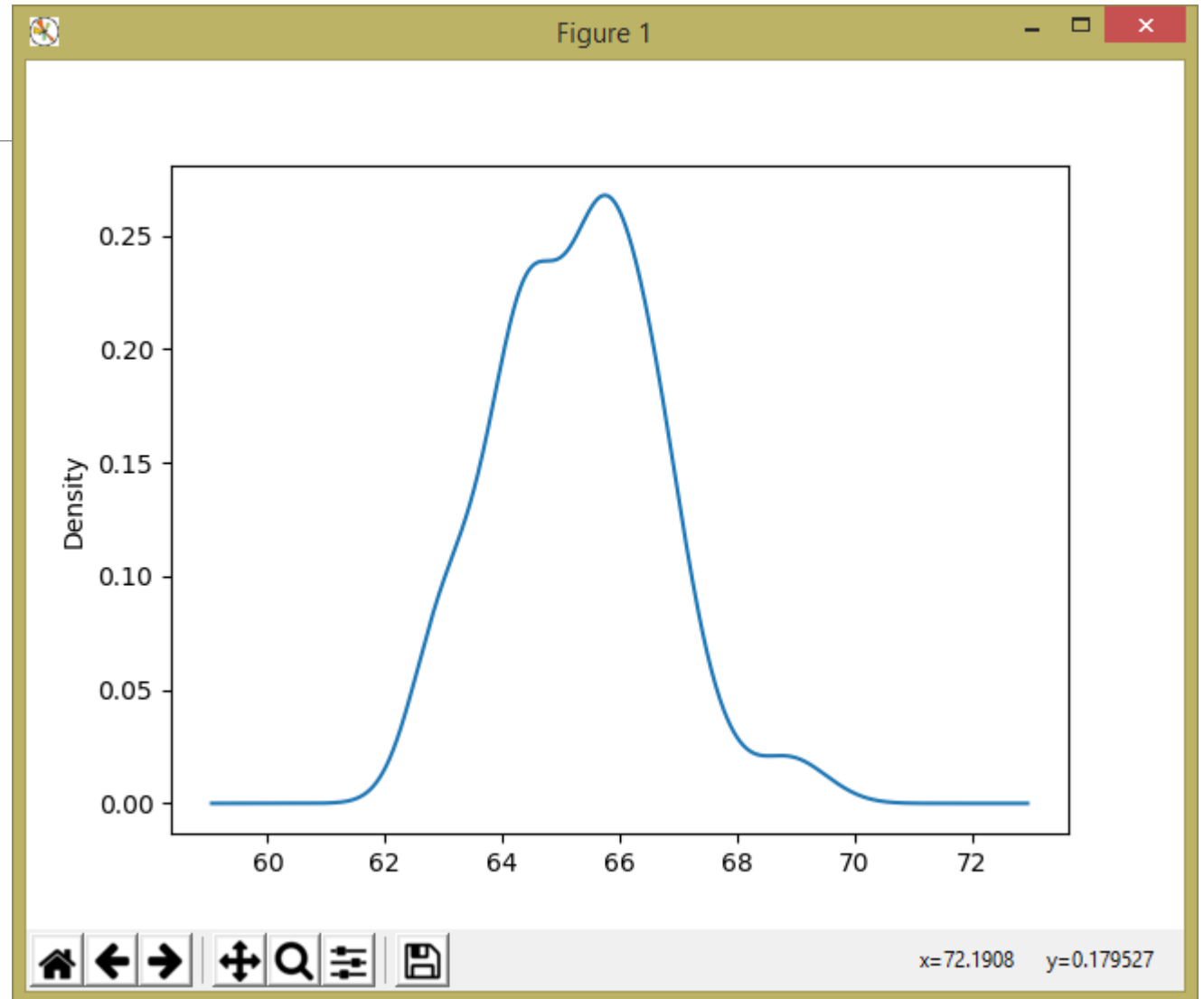
Наиболее известны графики P-P (вероятность-вероятность) и Q-Q (квантиль-квантиль).

График P-P строится на основе двух графиков интегральных функций распределения (CDF) параметров. Точки получаются путем сопоставления координат  $x$  (в процентах) двух разных CDF при одинаковых величинах  $Y$ .

График Q-Q строится на основе двух гистограмм параметров. Анализируется не «непрерывное» множество точек, а квантили каждого параметра. Точки получаются путем сопоставления координат квантилей одинаковых номеров двух разных параметров.

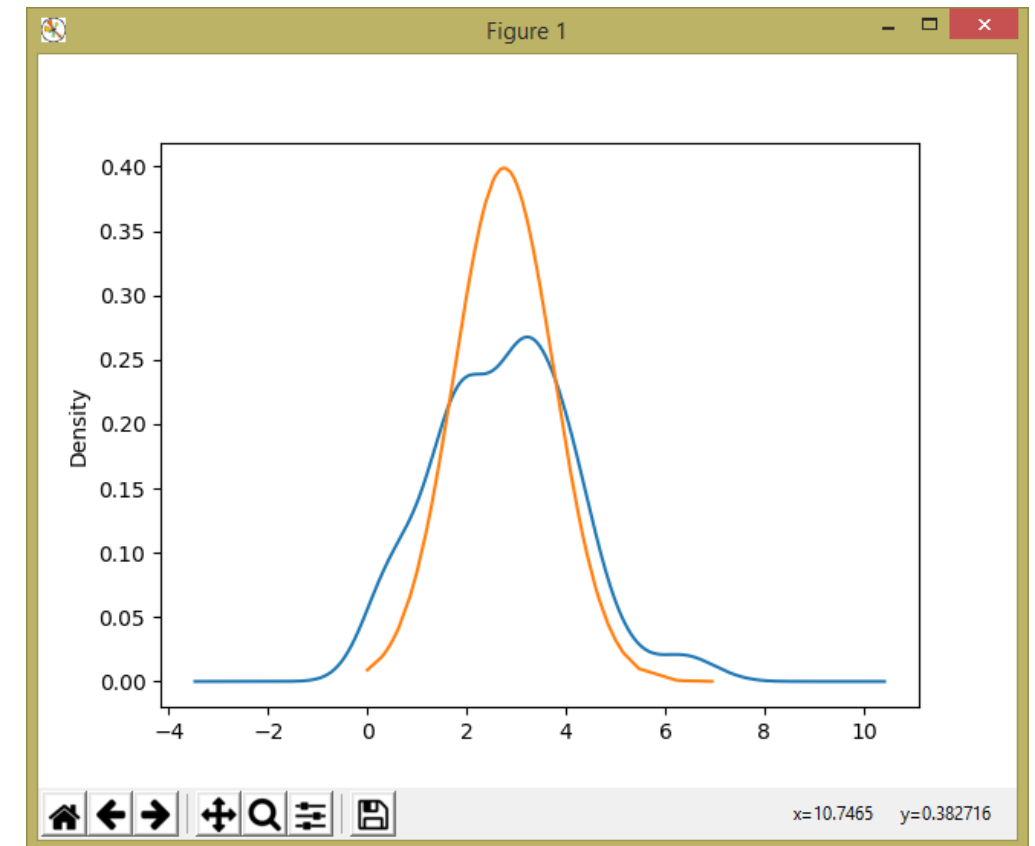
# PDF – функция плотности вероятности

```
example1.py - E:\Works\Victor\Students\S...  
File Edit Format Run Options Window Help  
table['USD'].plot.kde()  
plt.show()  
Ln: 23 Col: 0
```



# Сравним с нормальным распределением

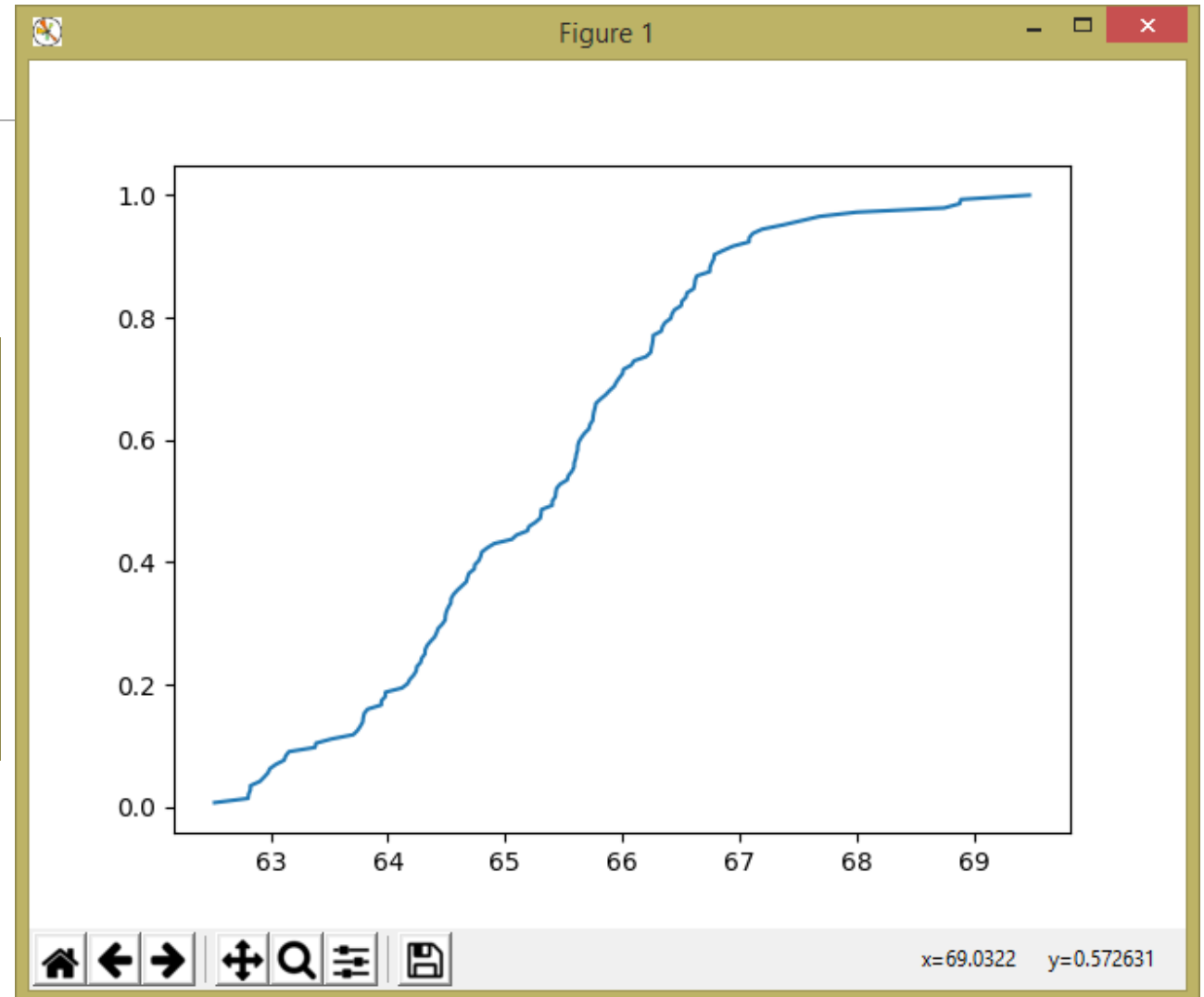
```
example1.py - E:\Works\Victor\Students\STDB\Lab2\example1.py (3.7.2)
File Edit Format Run Options Window Help
(table['USD'] - table['USD'].min()).plot.kde()
x = np.sort(table['USD'] - table['USD'].min())
plt.plot(x, stats.norm.pdf(x, x.mean(), 1))
plt.show()
Ln: 23 Col: 0
```





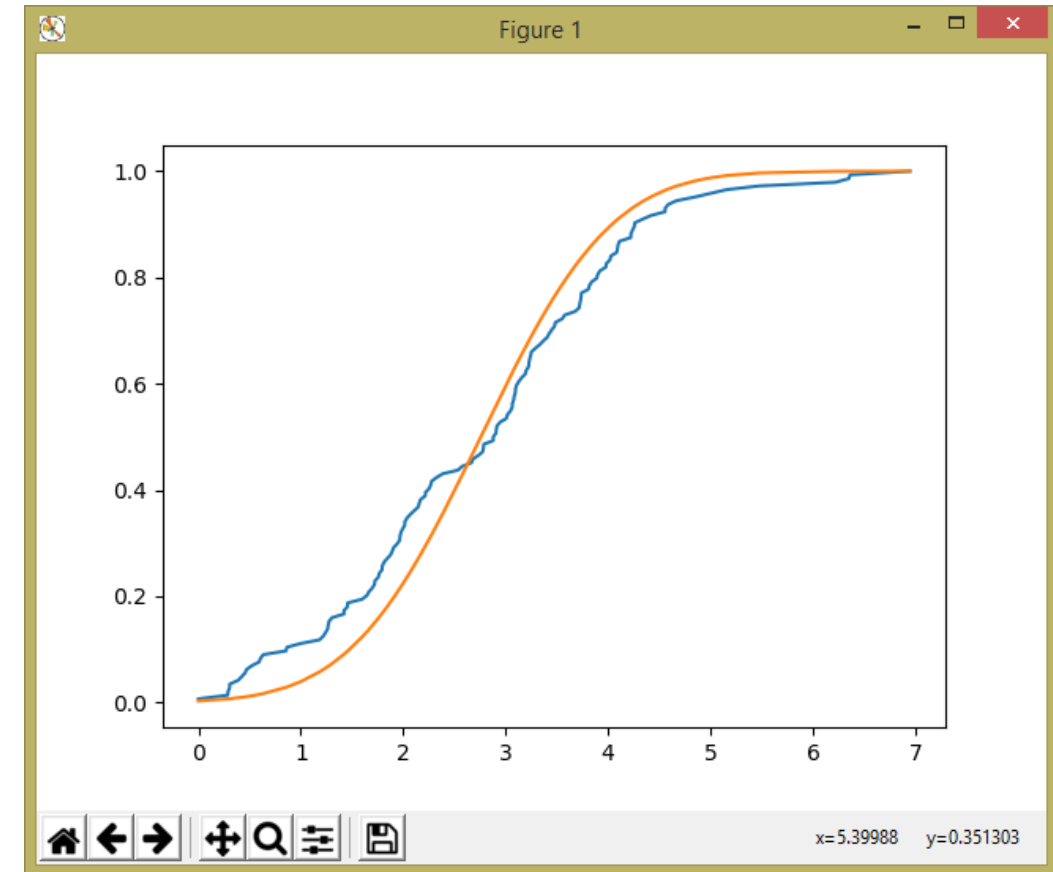
# CDF – интегральная функция распределения

```
*example1.py - E:\Works\Victor\Students\STDB\Lab2\example1.p...  
File Edit Format Run Options Window Help  
x = np.sort(table['USD'])  
y = np.arange(1, len(x) + 1) / len(x)  
plt.plot(x, y)  
plt.show()  
Ln: 34 Col: 0
```



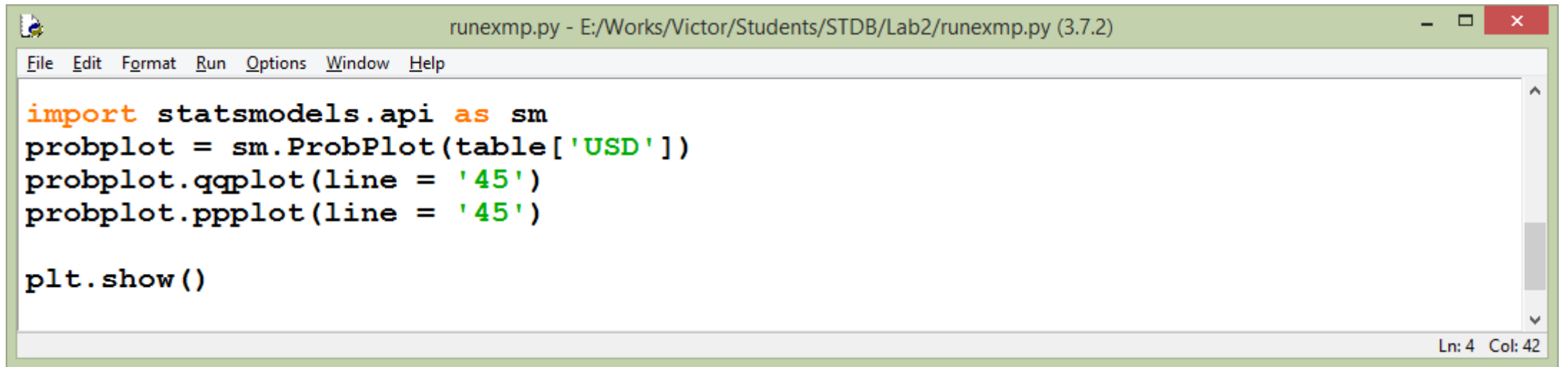
# Сравним с нормальным распределением

```
example1.py - E:\Works\Victor\Students\STDB\Lab2\example1.py (3.7.2) - □ ×
File Edit Format Run Options Window Help
x = np.sort(table['USD'] - table['USD'].min())
y = np.arange(1, len(x) + 1) / len(x)
plt.plot(x, y)
plt.plot(x, stats.norm.cdf(x, x.mean(), 1))
plt.show()
Ln: 34 Col: 0
```



# Попробуем построить P-P и Q-Q

---



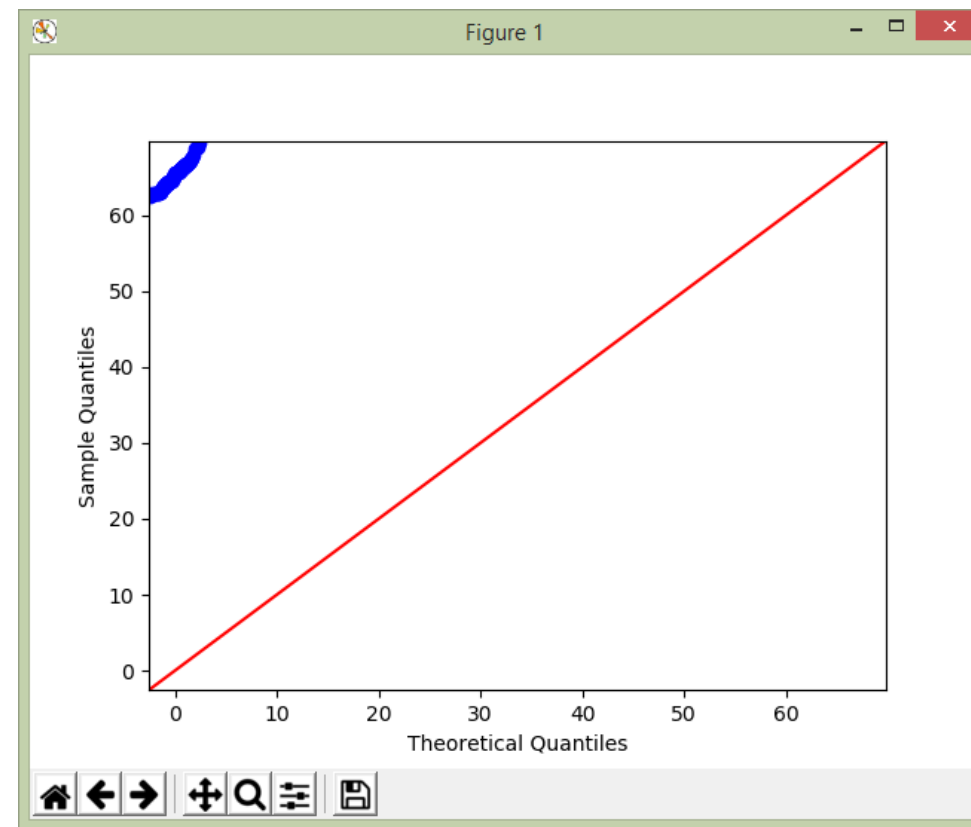
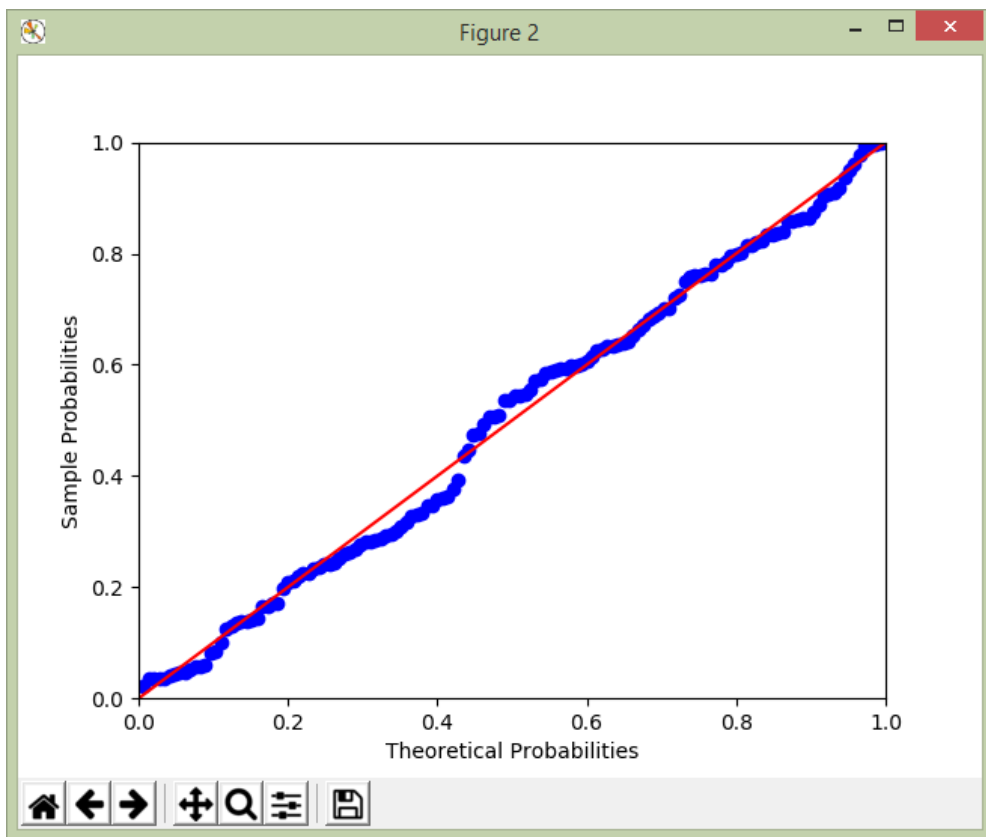
The screenshot shows a Python IDE window titled "runexmp.py - E:/Works/Victor/Students/STDB/Lab2/runexmp.py (3.7.2)". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The code editor contains the following Python code:

```
import statsmodels.api as sm
probplot = sm.ProbPlot(table['USD'])
probplot.qqplot(line = '45')
probplot.ppplot(line = '45')

plt.show()
```

The status bar at the bottom right indicates "Ln: 4 Col: 42".

# Попробуем построить



# Что не так?

---

Почему Q-Q график получился неудачным? Всё дело в том, что в отличие от графика P-P он чувствителен к абсолютным значениям сравниваемых величин. Чтобы избежать данной проблемы есть два варианта решения:

- выполнить стандартизацию;
- заменить линию 45 градусов на регрессионную прямую.

# Нормализация и стандартизация

---

Нормализация и стандартизация имеют множество назначений в численной математике и машинном обучении и других задачах, в том числе:

- 1) устранение дискриминации одних параметров численного метода по отношению к другим, возникающей за счёт разницы в их абсолютных числовых значениях (например, при подготовке данных к кластеризации по методу к-средних);
- 2) устранение ошибки численного метода, возникающей за счёт ограниченности представления чисел на ЭВМ (например, ошибка при сложении значительного числа маленьких и больших чисел);
- 3) визуализации результатов многомерных исследований (для совмещения реальных данных и доступного на экране пространства);
- 4) другие задачи.

# Нормализация данных

---

Нормализация данных (линейная нормализация) – это проекция данных на отрезок от 0 до 1 для неотрицательных данных и от -1 до 1 для данных, содержащих отрицательные числа.

Для отрезка от 0 до 1:

$$x' = \frac{x - x_{min}}{(x_{max} - x_{min})}$$

Для отрезка от -1 до 1:

$$x' = 2 \frac{x - x_{min}}{(x_{max} - x_{min})} - 1$$

# Преимущества и недостатки

---

1. Нормализация данных плохо применима при наличии значительно выделяющихся экстремальных выбросов.
2. Нормализация данных применима как к нормально распределённым данным, так и к данным с другим распределением.
3. Нормализация данных часто применяется в численных методах решения систем уравнений, дифференциальных уравнений, матричных численных методах и т.д.



# Стандартизация данных

---

Стандартизация данных – это такое биективное отображение данных из пространства действительных чисел в пространство действительных чисел, при котором данные оказываются распределёнными вокруг 0 со стандартным отклонением 1:

$$x' = \frac{x - M_x}{\sigma_x},$$

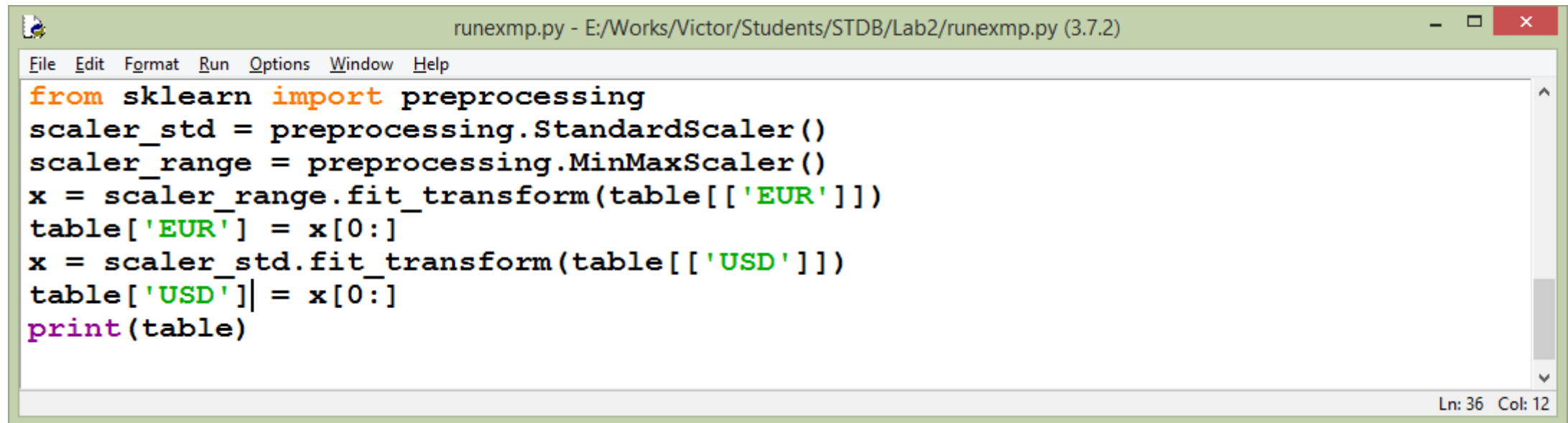
где  $M_x$  – математическое ожидание (среднее арифметическое) величины  $x$ , а  $\sigma_x$  – стандартное отклонение величины  $x$ .

# Преимущества и недостатки

---

1. Стандартизация данных применима даже при наличии значительно выделяющихся экстремальных выбросов.
2. Стандартизация данных хорошо применима только к нормально распределённым данным.
3. Стандартизация наиболее часто применяется для обеспечения работы таких методов машинного обучения, как  $k$ -средних ( $k$ -means), метод опорных векторов (SVM) и т.д.

# Используем



The image shows a screenshot of a Python IDE window titled "runexmp.py - E:/Works/Victor/Students/STDB/Lab2/runexmp.py (3.7.2)". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The main text area contains the following Python code:

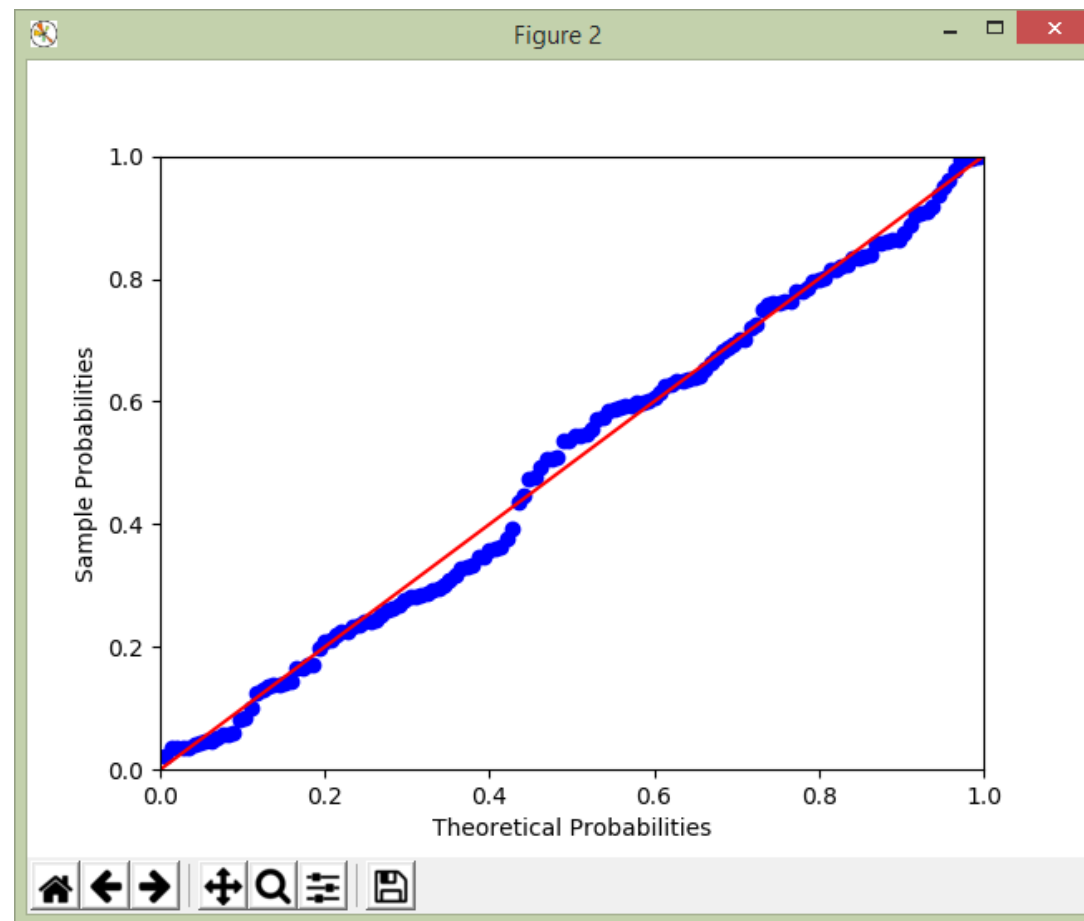
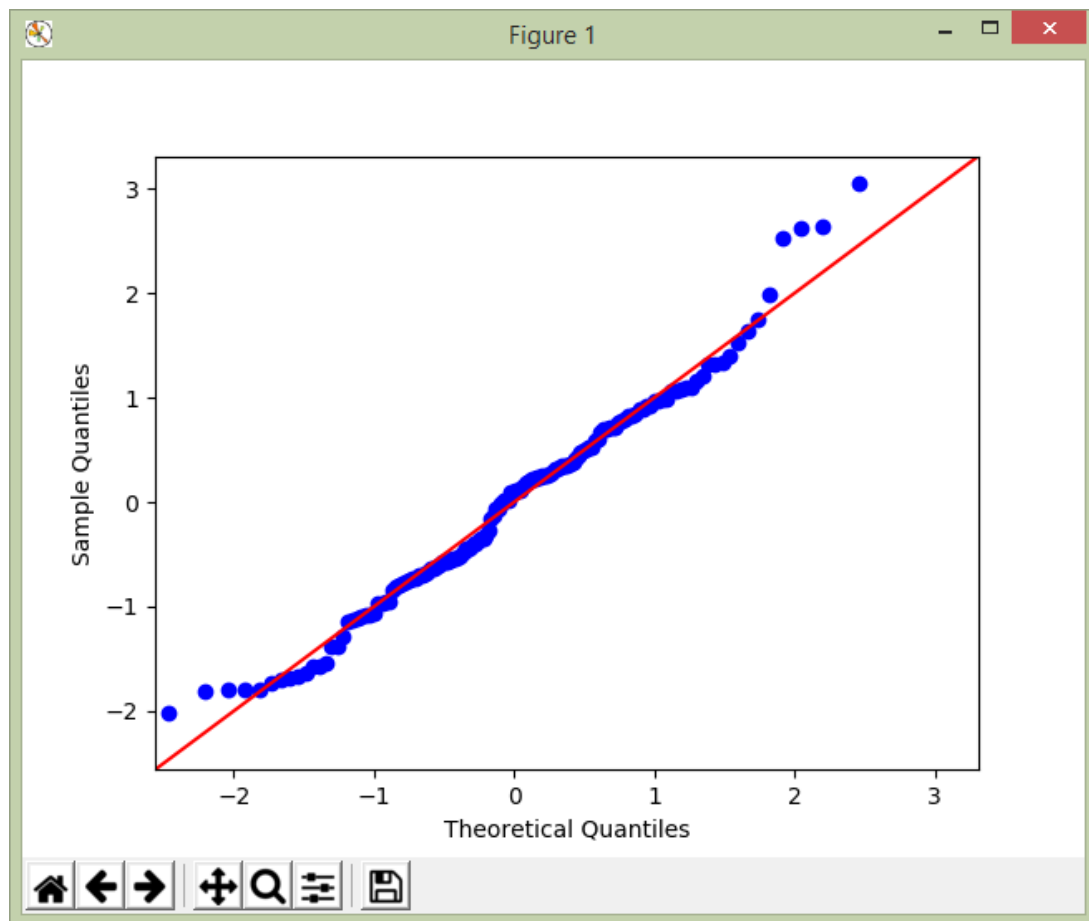
```
from sklearn import preprocessing
scaler_std = preprocessing.StandardScaler()
scaler_range = preprocessing.MinMaxScaler()
x = scaler_range.fit_transform(table[['EUR']])
table['EUR'] = x[0:]
x = scaler_std.fit_transform(table[['USD']])
table['USD'] = x[0:]
print(table)
```

The status bar at the bottom right of the window indicates "Ln: 36 Col: 12".

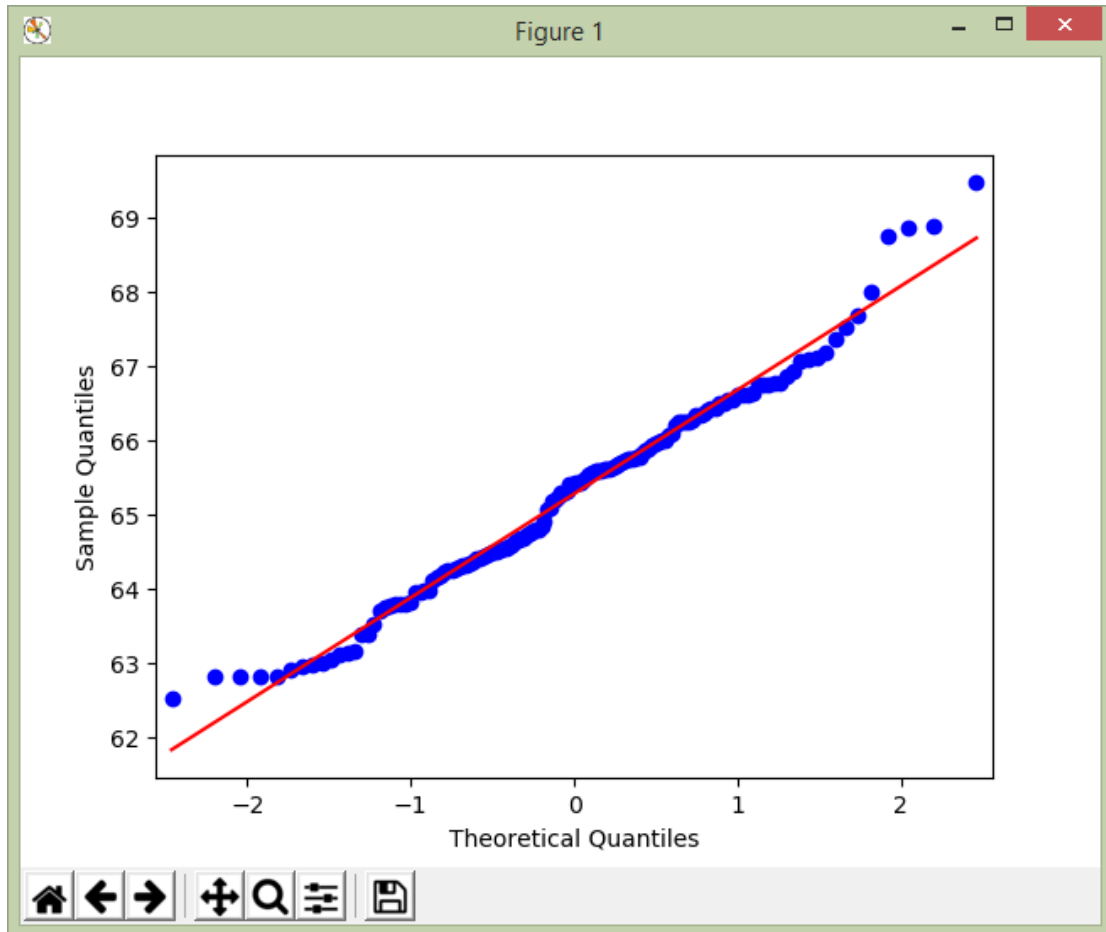
# Используем

```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:/Works/Victor/Students/STDB/Lab2/runexmp.py =====
      EUR      USD      CHF  Brent
0    0.639188  0.343711  66.3881  81.56
1    0.625376  0.182951  66.3063  80.36
2    0.596524  0.089618  65.9766  79.35
3    0.587838  0.324008  66.1471  80.02
4    0.566761  0.019491  65.5293  76.24
5    0.512551  0.022118  65.5526  75.68
6    0.536087  0.255486  65.8472  76.70
7    0.525971  0.341376  65.8596  77.82
8    0.531776  0.389027  65.8590  76.18
9    0.505497  0.360787  65.5644  74.59
10   0.643276  1.637526  66.8420  65.35
11   0.641274  1.752386  66.9514  65.86
12   0.713500  1.983200  67.4779  66.63
13   0.584301  0.975002  66.3241  67.10
14   0.561634  0.531471  66.0543  62.44
```

# Гораздо лучше



# Альтернатива

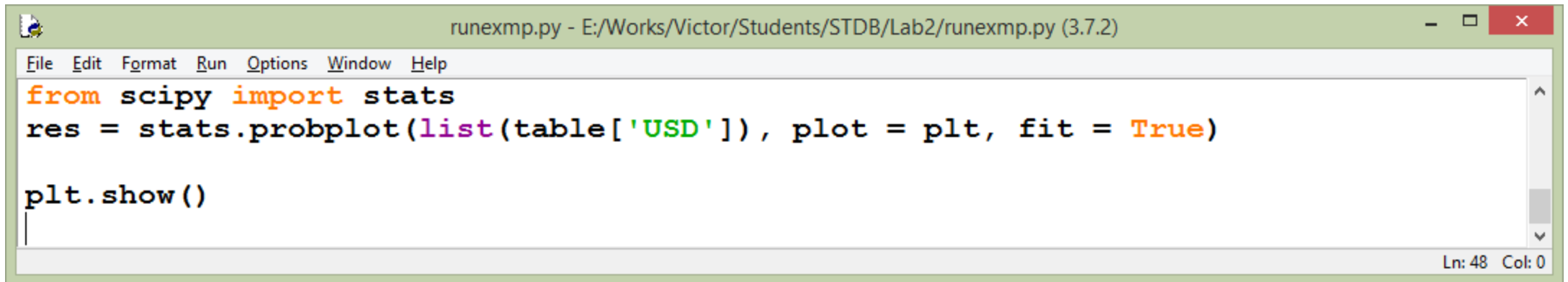


```
runexmp.py - E:/Works/Victor/Students/STDB/Lab2/runexmp.py (3.7.2)
File Edit Format Run Options Window Help
import statsmodels.api as sm
probplot = sm.ProbPlot(table['USD'])
probplot.qqplot(line = 'r')
probplot.ppplot(line = '45')

plt.show()
Ln: 25 Col: 27
```

# Такой вариант можно и проще

---



The screenshot shows a Python IDE window titled "runexmp.py - E:/Works/Victor/Students/STDB/Lab2/runexmp.py (3.7.2)". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The code editor contains the following Python code:

```
from scipy import stats
res = stats.probplot(list(table['USD']), plot = plt, fit = True)

plt.show()
```

The status bar at the bottom right indicates "Ln: 48 Col: 0".

# Что и для чего?

---

P-P график лучше подходит для анализа центра распределения (участка с высокой плотностью).

Q-Q график лучше подходит для анализа хвостов распределения (участки с низкой плотностью распределения).



# Аналитические критерии

---

Выбор критерия нормальности делается в зависимости от выборки:

- 7-2000 наблюдений – критерий Шапиро-Уилка (основан на отношении оптимальной линейной несмещённой оценке дисперсии к её обычной оценке методом максимального правдоподобия);
- > 2000 наблюдений – подходят методы на основе эмпирической функции распределения (EDF):
  - для большого числа отклонений в хвостах – критерий Андерсона-Дарлинга;
  - для большого числа отклонений в середине – критерий Колмогорова-Смирнова.
- Для очень больших выборок и низкой вычислительной сложности подходит тест Жака-Бера – один из самых простых тестов. Как его использовать в Python мы рассмотрим в более поздних лекциях.

В общем случае для выборок с большим числом наблюдений чаще всего лучший результат даёт критерий Андерсона-Дарлинга (если хватает вычислительной мощности).

# Формулы

---

Жака-Бера:  $JB = n / 6 * [S^2 + (K - 3)^2/24]$ , где

$n$  – размер выборки;

$S$  – коэффициент асимметрии (мера сдвига распределения влево или вправо);

$K$  – эксцесс мера выпуклости или вогнутости (ниже – выше).

Шапиро-Уилка

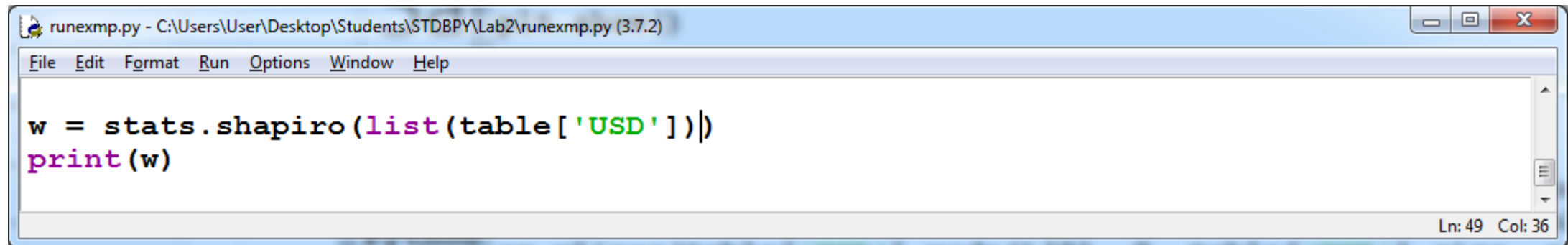
$$W = \frac{(\sum_{i=1}^n a_i x_{(i)})^2}{\sum_{i=1}^n (x_i - \bar{x})^2}, \text{ где}$$

$x_{(i)}$  -  $i$ -е наименьшее число в выборке;

$a_i$  – коэффициенты (формула здесь приводиться не будет).

# Критерий Шапиро-Уилка

---

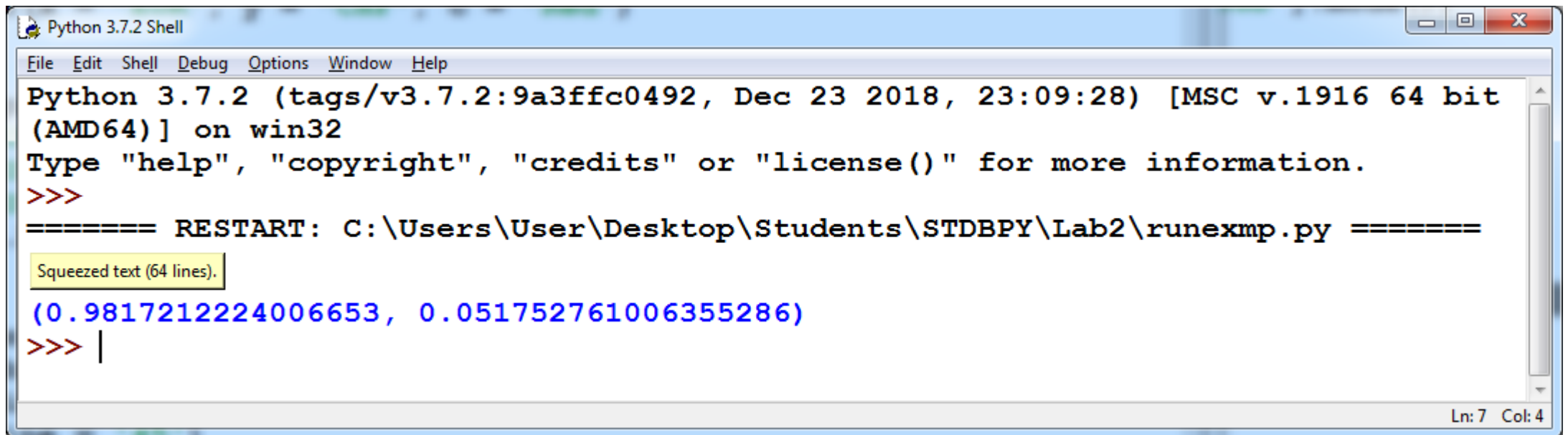


The image shows a screenshot of a Python IDE window titled "runexmp.py - C:\Users\User\Desktop\Students\STDBPY\Lab2\runexmp.py (3.7.2)". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The main text area contains the following Python code:

```
w = stats.shapiro(list(table['USD']))  
print(w)
```

The status bar at the bottom right of the window indicates "Ln: 49 Col: 36".

# Результат



The screenshot shows a Python 3.7.2 Shell window with a menu bar (File, Edit, Shell, Debug, Options, Window, Help). The output text is as follows:

```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\User\Desktop\Students\STDBPY\Lab2\runexmp.py =====
Squeezed text (64 lines).
(0.9817212224006653, 0.051752761006355286)
>>> |
```

The status bar at the bottom right indicates "Ln: 7 Col: 4".

# Интерпретация результата

---

Метод shapiro вернул нам два значения. Первое значение – это, так называемая, W-статистика. Сравнивая данное значение с таблицей критических значений (W должно быть больше критического значения) можно вычислить уровень значимости гипотезы  $H_0$ , проверяемой критерием.

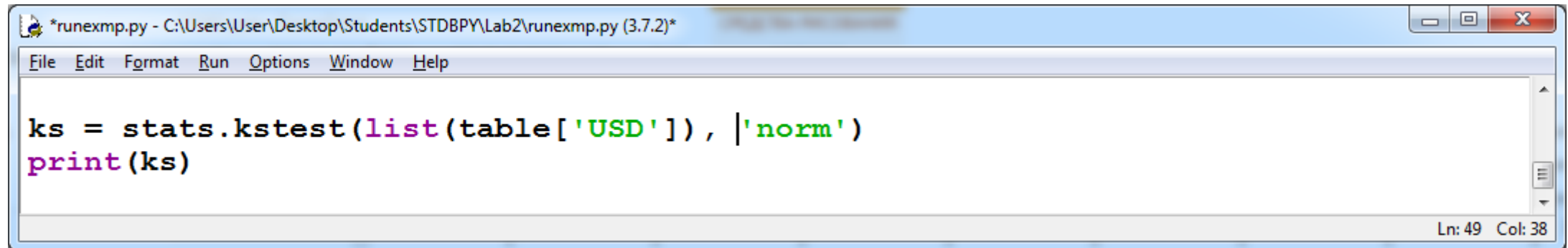
В нашем случае гипотеза  $H_0$  звучит следующим образом: «Распределение является нормальным».

Второе значение – это, так называемое, p-значение (p-value). Фактически, это уровень значимости нашей гипотезы. Или, говоря проще, вероятность допустить ошибку, отбросив гипотезу  $H_0$ .

Результаты теста показывают уровень значимости в районе 5% (0.05). Требуемый уровень значимости выбирается заранее. Если p-value больше заданного уровня значимости, то гипотеза  $H_0$  не отвергается. Если p-value меньше заданного уровня значимости, гипотеза отвергается.

# Критерий Колмогорова-Смирнова

---

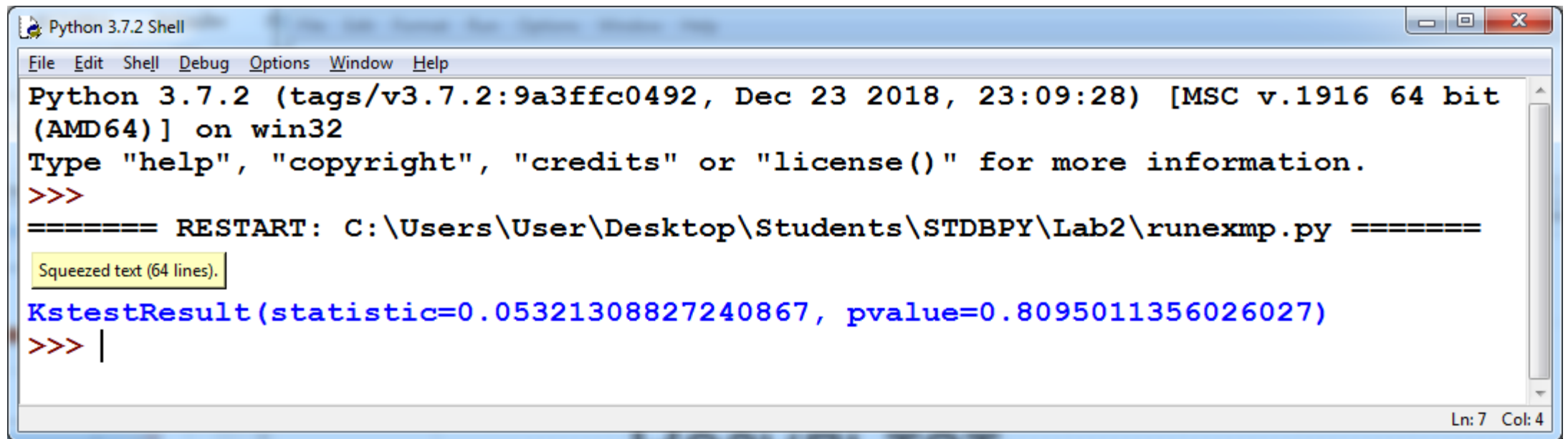


A screenshot of a Python IDE window titled "\*runexmp.py - C:\Users\User\Desktop\Students\STDBPY\Lab2\runexmp.py (3.7.2)\*". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The main text area contains the following Python code:

```
ks = stats.kstest(list(table['USD']), |'norm')
print(ks)
```

The status bar at the bottom right indicates "Ln: 49 Col: 38".

# Результат



The screenshot shows a Python 3.7.2 Shell window with a menu bar (File, Edit, Shell, Debug, Options, Window, Help). The main text area displays the following output:

```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\User\Desktop\Students\STDBPY\Lab2\runexmp.py =====
Squeezed text (64 lines).
KstestResult(statistic=0.05321308827240867, pvalue=0.8095011356026027)
>>> |
```

The status bar at the bottom right indicates "Ln: 7 Col: 4".

# Интерпретация результата

---

Метод `kstest` вернул нам два значения. Первое значение – это, так называемая, D-статистика. Сравнивая данное значение с таблицей критических значений (D должно быть меньше критического значения) можно вычислить уровень значимости гипотезы  $H_0$ , проверяемой критерием.

В нашем случае гипотеза  $H_0$  звучит следующим образом: «Распределение является нормальным».

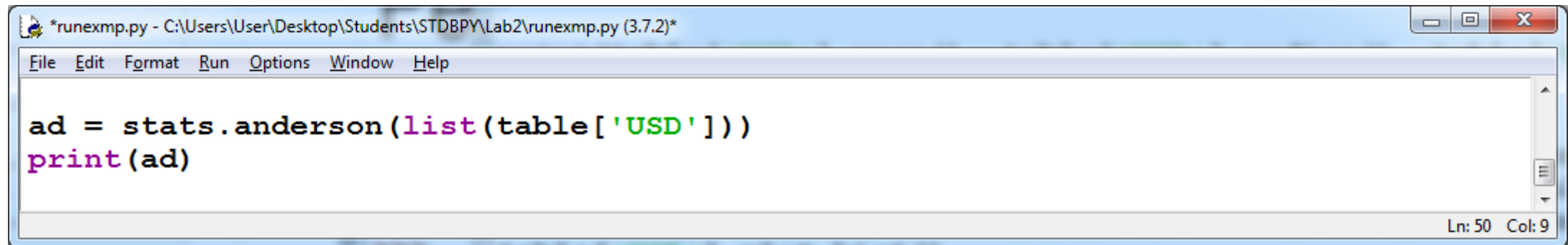
Второе значение – это, так называемое, p-значение (p-value). Фактически, это уровень значимости нашей гипотезы. Или, говоря проще, вероятность допустить ошибку, отбросив гипотезу  $H_0$ .

Результаты теста показывают уровень значимости в районе 80% (0.8). Требуемый уровень значимости выбирается заранее. Если p-value больше заданного уровня значимости, то гипотеза  $H_0$  не отвергается. Если p-value меньше заданного уровня значимости, гипотеза отвергается.



# Критерий Андерсона-Дарлинга

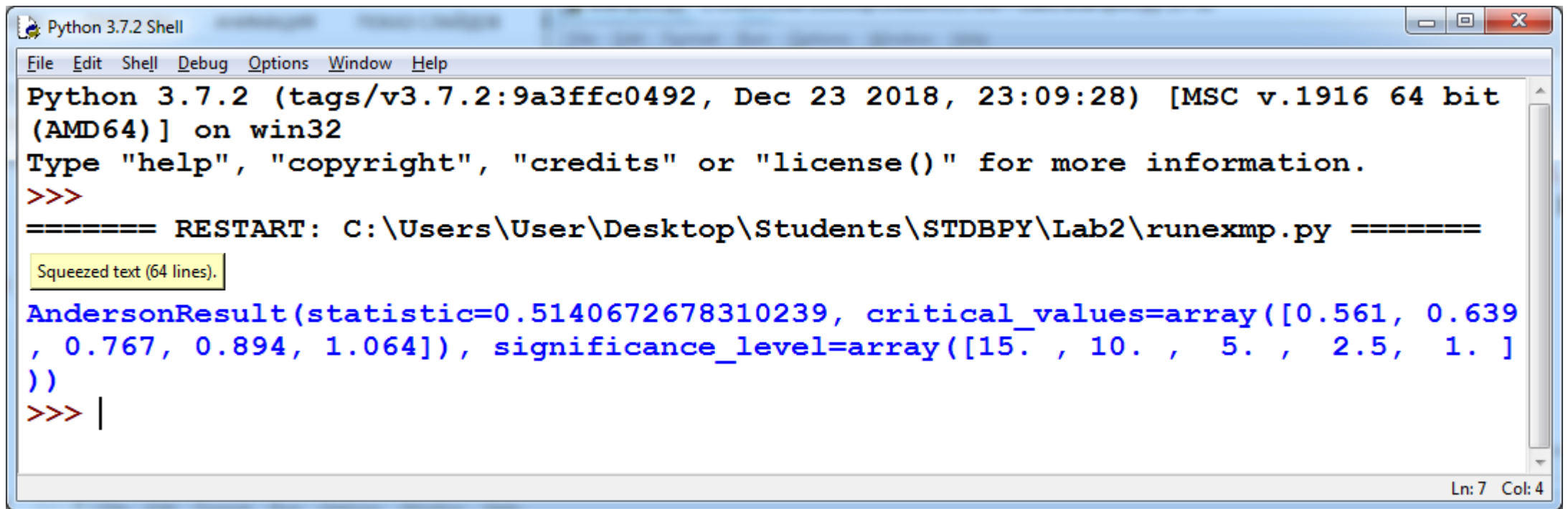
---



The image shows a screenshot of a Python IDE window. The title bar reads '\*runexmp.py - C:\Users\User\Desktop\Students\STDBPY\Lab2\runexmp.py (3.7.2)\*'. The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code editor contains two lines of Python code: `ad = stats.anderson(list(table['USD']))` and `print(ad)`. The status bar at the bottom right indicates 'Ln: 50 Col: 9'.

```
*runexmp.py - C:\Users\User\Desktop\Students\STDBPY\Lab2\runexmp.py (3.7.2)*  
File Edit Format Run Options Window Help  
  
ad = stats.anderson(list(table['USD']))  
print(ad)  
  
Ln: 50 Col: 9
```

# Результат



```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\User\Desktop\Students\STDBPY\Lab2\runexmp.py =====
Squeezed text (64 lines).
AndersonResult(statistic=0.5140672678310239, critical_values=array([0.561, 0.639
, 0.767, 0.894, 1.064]), significance_level=array([15. , 10. , 5. , 2.5, 1. ]
))
>>> |
```

Ln: 7 Col: 4

# Интерпретация результата

---

Метод anderson вернул нам одно число и два массива значений. Первое число – это, так называемая, А-статистика. Сравнивая данное значение с таблицей критических значений (А должно быть меньше критического значения) можно вычислить уровень значимости гипотезы  $H_0$ , проверяемой критерием.

В нашем случае гипотеза  $H_0$  звучит следующим образом: «Распределение является нормальным».

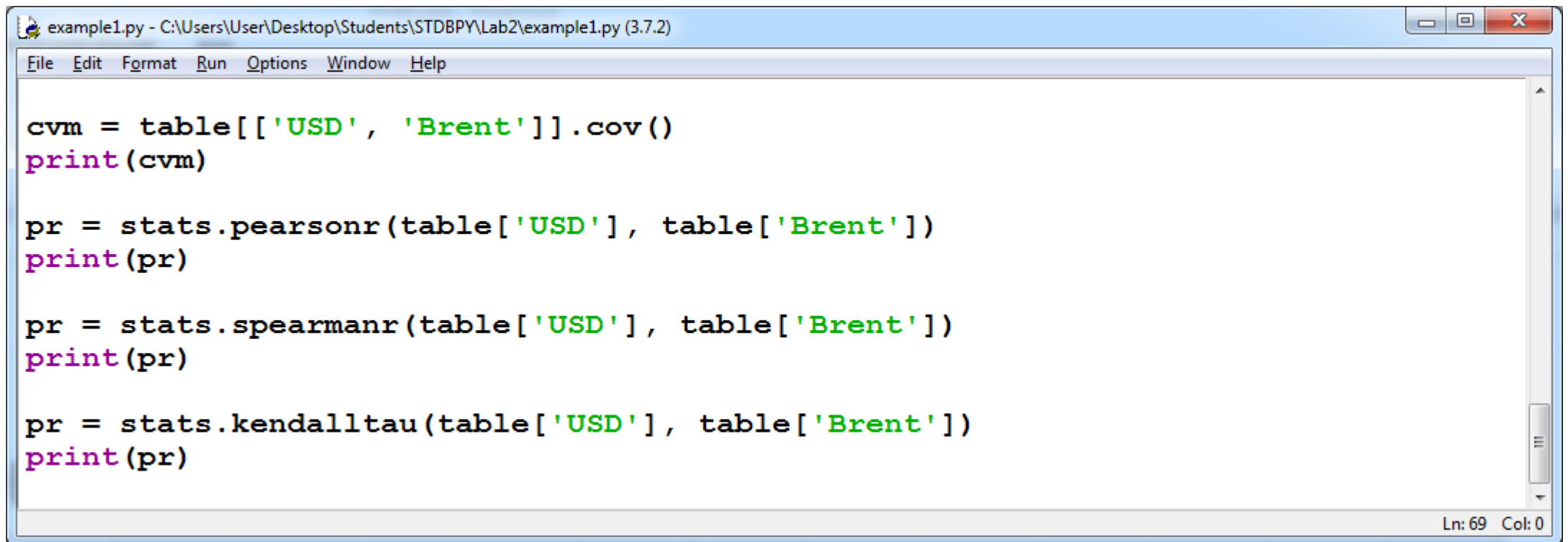
Два массива значений – это и есть таблица критических значений и уровней значимости.

Так как полученная статистика меньше всех приведённых критических значений, то можно говорить об уровне значимости 15% (0.15).

# Выбор критерия корреляции

Метод	Шкала измерения показателей	Количество сравниваемых совокупностей	Распределение данных	Замечание
Коэффициент корреляции Пирсона	количественная	2 ряда измерений	нормальное	Для исследования линейных зависимостей
Коэффициент ранговой корреляции Спирмена	количественная, ранговая	2 ряда измерений	любое	Если не подходит метод Пирсона
Коэффициент корреляции Кендалла	количественная, ранговая	2 ряда измерений	любое	Более предпочтителен, чем метод Спирмена, но вычислительно на порядок медленнее

# Ковариация и корреляция



The image shows a screenshot of a Python IDE window titled "example1.py - C:\Users\User\Desktop\Students\STDBPY\Lab2\example1.py (3.7.2)". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The main text area contains the following Python code:

```
cvm = table[['USD', 'Brent']].cov()
print(cvm)

pr = stats.pearsonr(table['USD'], table['Brent'])
print(pr)

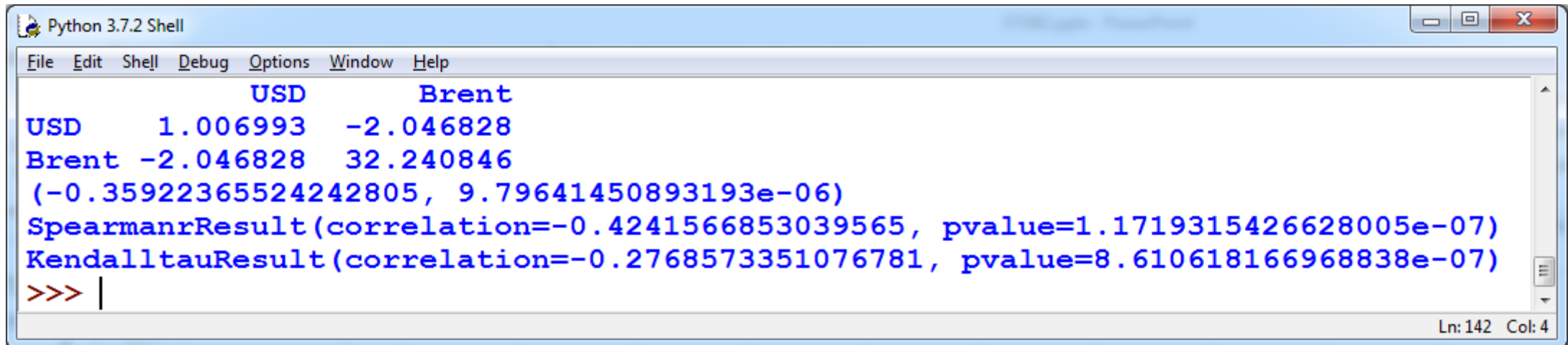
pr = stats.spearmanr(table['USD'], table['Brent'])
print(pr)

pr = stats.kendalltau(table['USD'], table['Brent'])
print(pr)
```

The status bar at the bottom right indicates "Ln: 69 Col: 0".

# Результаты

---



A screenshot of a Python 3.7.2 Shell window. The window has a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area displays the following output in blue text:

```
          USD      Brent
USD      1.006993  -2.046828
Brent -2.046828   32.240846
(-0.35922365524242805, 9.79641450893193e-06)
SpearmanrResult(correlation=-0.4241566853039565, pvalue=1.1719315426628005e-07)
KendalltauResult(correlation=-0.2768573351076781, pvalue=8.610618166968838e-07)
>>> |
```

The status bar at the bottom right indicates 'Ln: 142 Col: 4'.

# Как интерпретировать?

---

Все три критерия корреляции позволяют дать оценку нулевой гипотезе ( $H_0$ ) об отсутствии корреляции между двумя переменными. Значения критериев Пирсона, Спирмена, Кендалла находятся в диапазоне от -1 до 1 (0 – нет корреляции, 1 – полная корреляция, -1 – полная обратная корреляция).

Помимо самого коэффициента Python выводит также p-value, определяющее уровень значимости гипотезы  $H_0$ . Если он очень маленький (как в нашем случае), то гипотеза об отсутствии корреляции должна быть отброшена.

В нашем примере видно наличие обратной корреляции.

# Уровни корреляции по критерию Пирсона

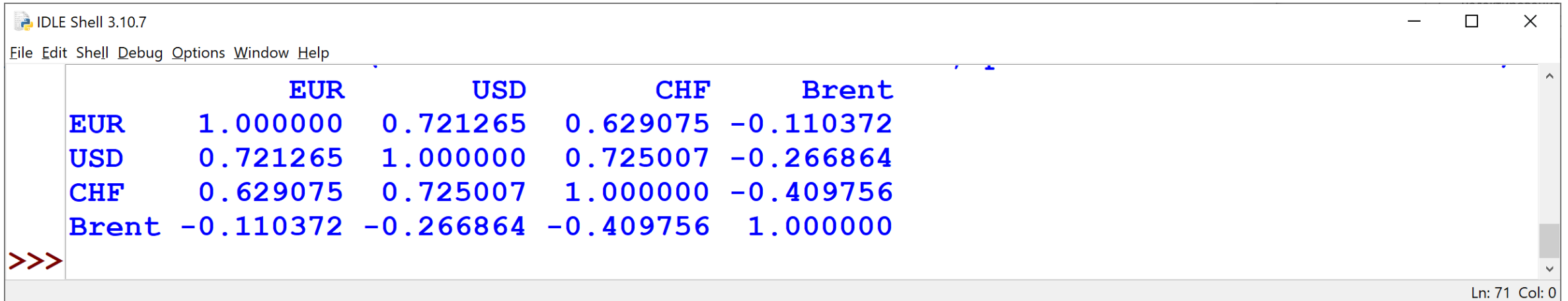
---

Абсолютное значение $r$	Теснота (сила) корреляционной связи
менее 0.3	слабая
от 0.3 до 0.5	умеренная
от 0.5 до 0.7	заметная
от 0.7 до 0.9	высокая
более 0.9	весьма высокая



# Матрица корреляции

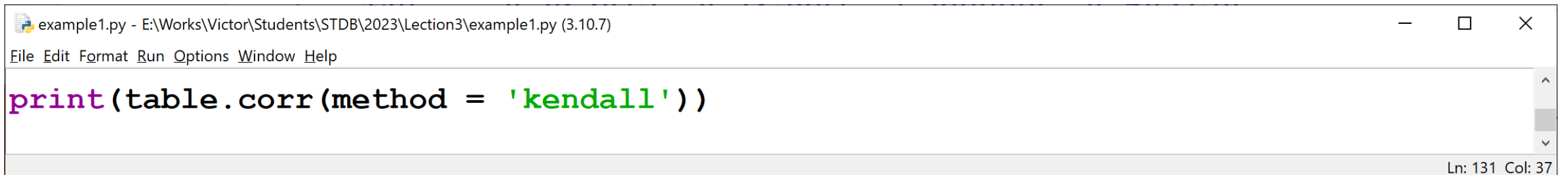
Матрица корреляции показывает попарные коэффициенты корреляции для каждой из возможных пар атрибутов.



```
IDLE Shell 3.10.7
File Edit Shell Debug Options Window Help

      EUR      USD      CHF      Brent
EUR      1.000000  0.721265  0.629075 -0.110372
USD      0.721265  1.000000  0.725007 -0.266864
CHF      0.629075  0.725007  1.000000 -0.409756
Brent -0.110372 -0.266864 -0.409756  1.000000
>>>
```

Ln: 71 Col: 0



```
example1.py - E:\Works\Victor\Students\STDB\2023\Lecture3\example1.py (3.10.7)
File Edit Format Run Options Window Help

print(table.corr(method = 'kendall'))
```

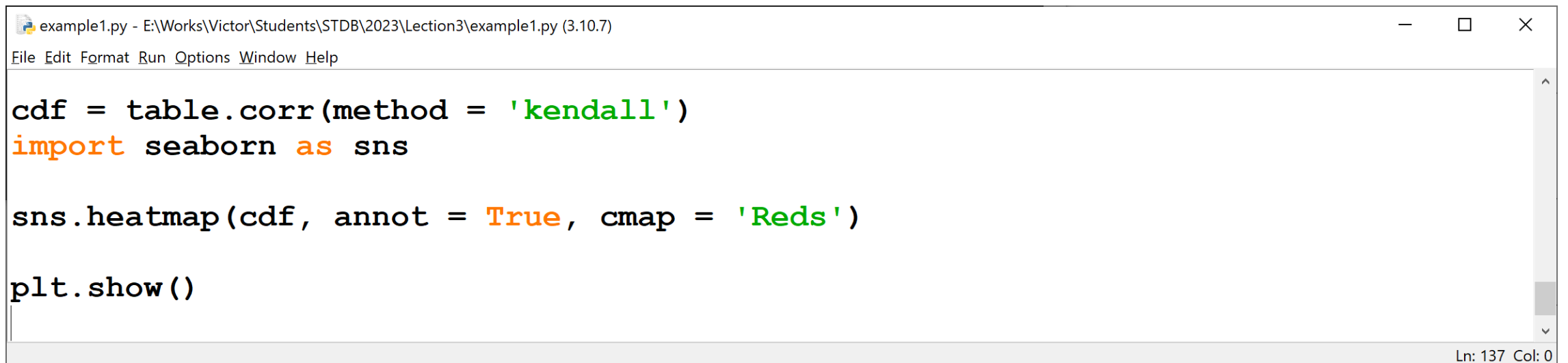
Ln: 131 Col: 37

# Тепловая карта

---

Для отображения тепловой карты с помощью pip установим пакет seaborn:

```
python.exe -m pip install seaborn
```

A screenshot of a Python IDE window titled 'example1.py - E:\Works\Victor\Students\STDB\2023\Lecture3\example1.py (3.10.7)'. The window has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code editor contains the following Python code:

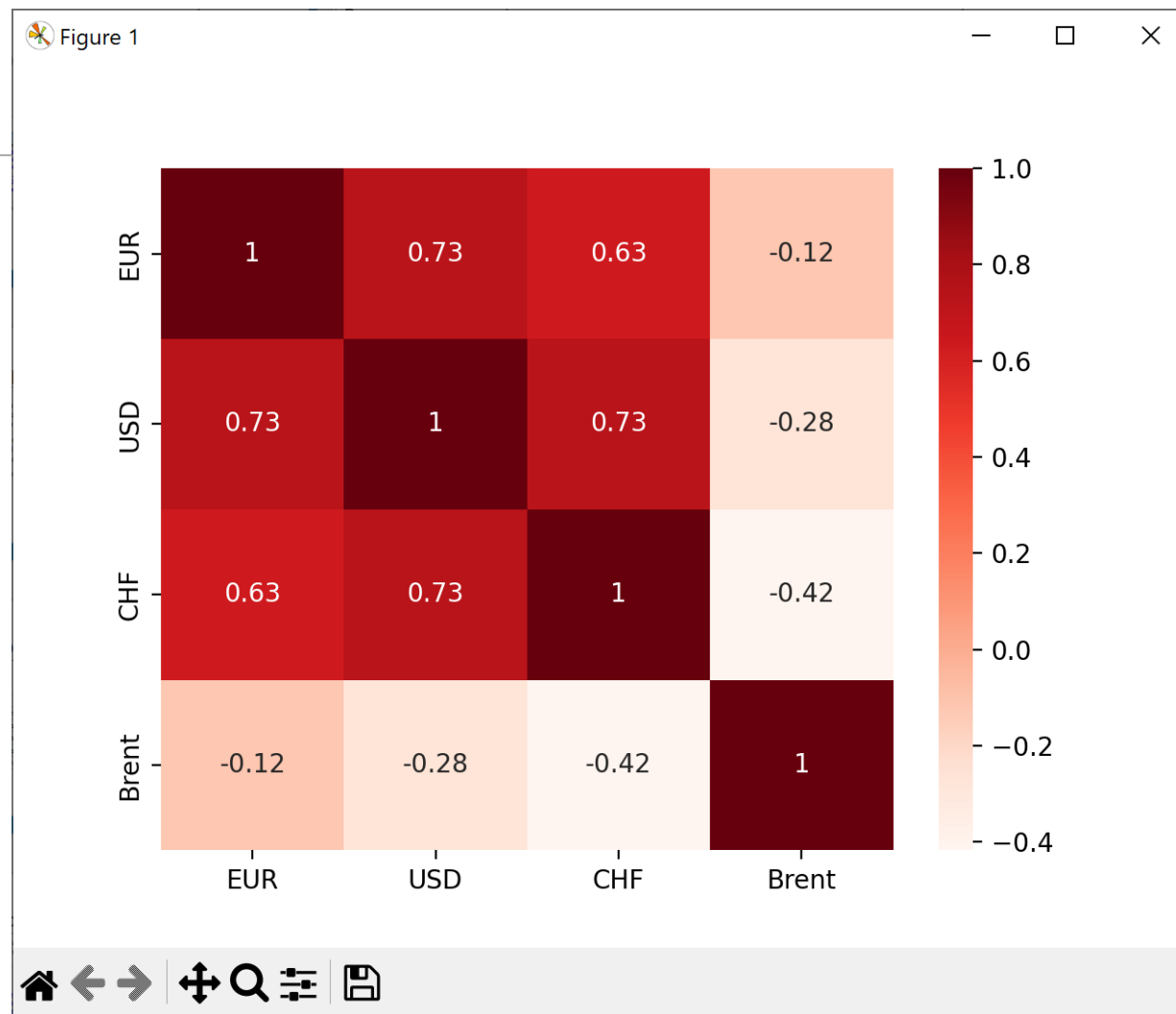
```
cdf = table.corr(method = 'kendall')
import seaborn as sns

sns.heatmap(cdf, annot = True, cmap = 'Reds')

plt.show()
```

The status bar at the bottom right indicates 'Ln: 137 Col: 0'.

# Тепловая карта



# Интернет ресурсы и литература

---

1. <https://towardsdatascience.com/explaining-probability-plots-9e5c5d304703>
2. <https://desktop.arcgis.com/ru/arcmap/10.4/extensions/geostatistical-analyst/normal-qq-plot-and-general-qq-plot.htm>
3. <http://window.edu.ru/resource/562/65562/files/m08-196.pdf>
4. [http://edu.tltsu.ru/sites/sites\\_content/site216/html/media96435/lec\\_11-1.pdf](http://edu.tltsu.ru/sites/sites_content/site216/html/media96435/lec_11-1.pdf)