**Лабораторная работа №4 «Использование библиотек Python для работы со случайными числами»**

Порядок выполнения практической работы:

1. Ознакомиться с возможностями модуля special библиотеки SciPy для генерации случайных чисел и их последовательностей.

*Рекомендуемые источники:*

- приложение1;

- https://skillbox.ru/media/code/gotovimsya\_k\_sobesedovaniyu\_sluchaynye\_chisla\_v\_python/

- https://python-scripts.com/random-data

- <https://ps.readthedocs.io/ru/latest/random.html>

1. Изучить примеры генерации случайных чисел средствами библиотеки [random](https://python-scripts.com/random" \t "_blank) и модуля [numpy.random](https://pythonworld.ru/numpy/3.html" \t "_blank) из библиотеки numpy. Реализовать приведенные примеры (см. рис. 1- 4) на языке программирования Python, изменив параметры вызываемых операторов (например, диапазоны случайных чисел).
2. Оставить и реализовать на Python задачу случайного выбора объекта из множества (аналогично задаче о шарах или монетах, представленные на рис. 5,6).
3. Сформировать отчет, включающий распечатку программного кода на языке Python с распечаткой результатов и соответствующими комментариями (относительно применяемых параметров процедур и функций, а также решаемой задачи) .

**Приложение 1[[1]](#footnote-1).**

Случайные числа (в контексте программирования) — это искусственно полученная последовательность чисел из определённого диапазона, которая подчиняется одному из законов распределения случайной величины.

Случайные числа и их последовательности используются в математических моделях различных сфер науки и техники: в экономике (страхование, логистика), физике (движение частиц), микроэлектронике, криптографии и др.

Существует два основных способа получения псевдослучайных чисел на Python: с помощью «родной» библиотеки [random](https://python-scripts.com/random" \t "_blank) и с помощью модуля [numpy.random](https://pythonworld.ru/numpy/3.html" \t "_blank) из библиотеки numpy (см. рис. 1, 2)

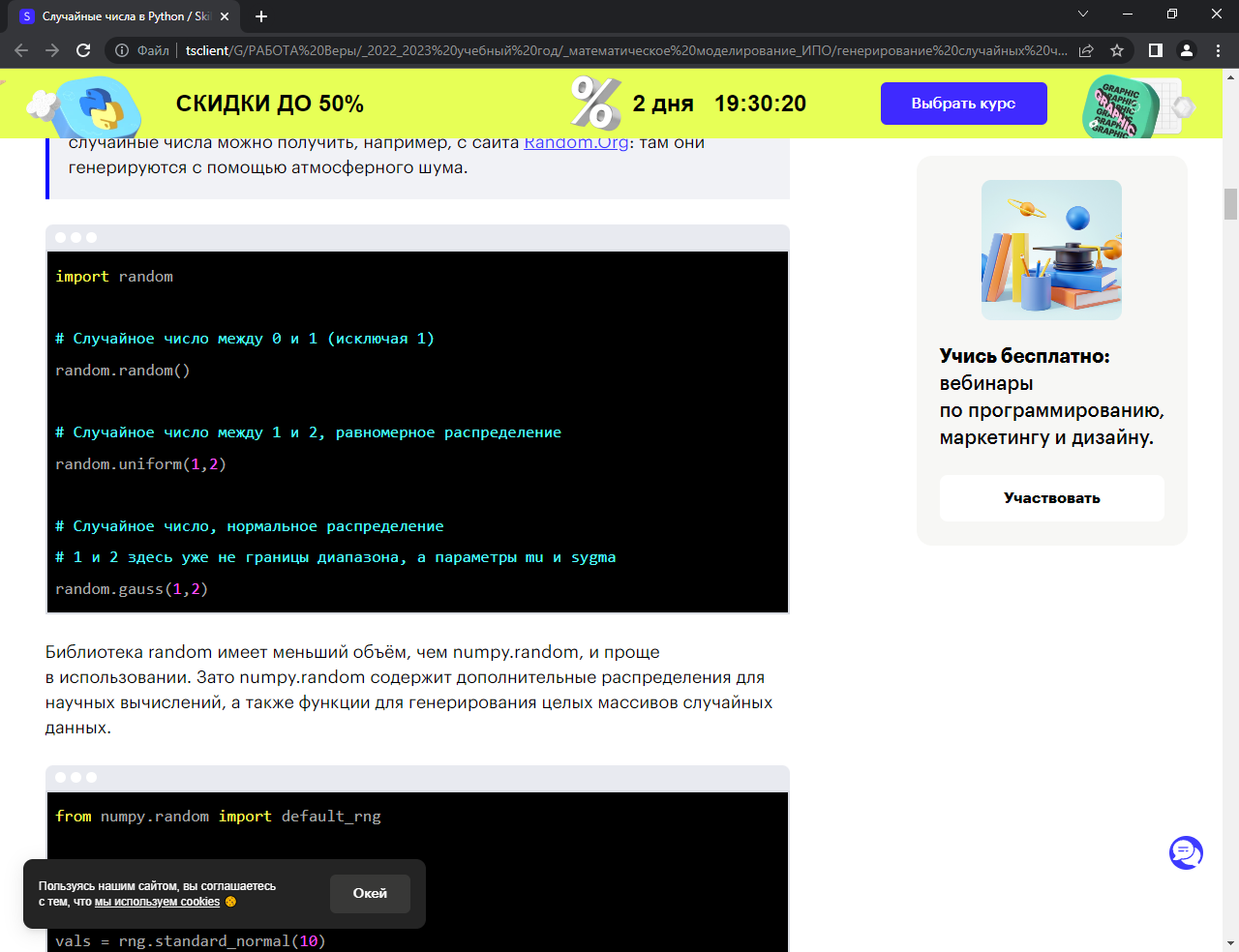


Рис1.

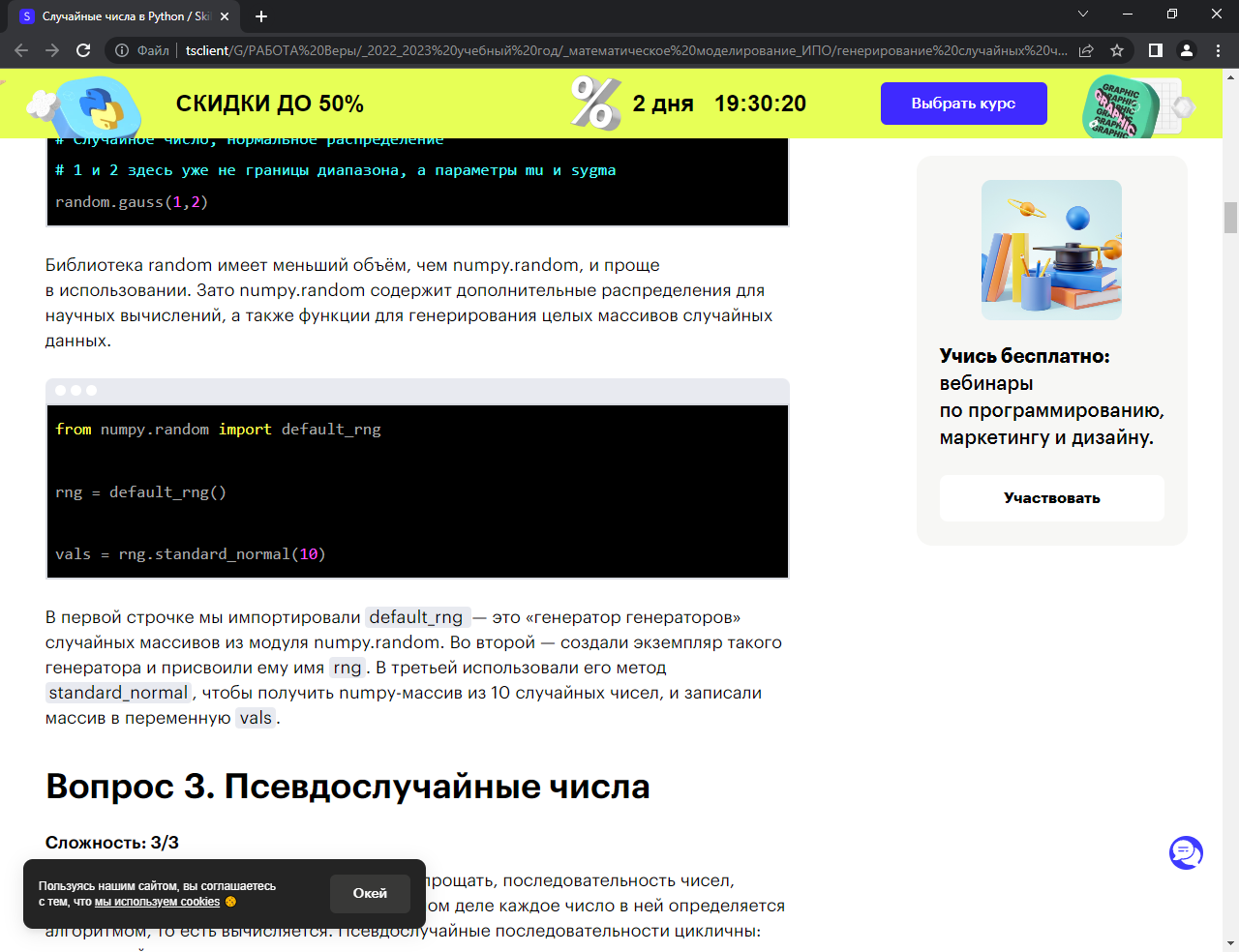


Рис. 2.

Истинно случайную последовательность повторить невозможно. Но для повторения псевдослучайных чисел в обеих основных библиотеках — random и numpy.random есть функция seed (), которая отвечает за инициализацию («посев») последовательности (см. рис. 3):.

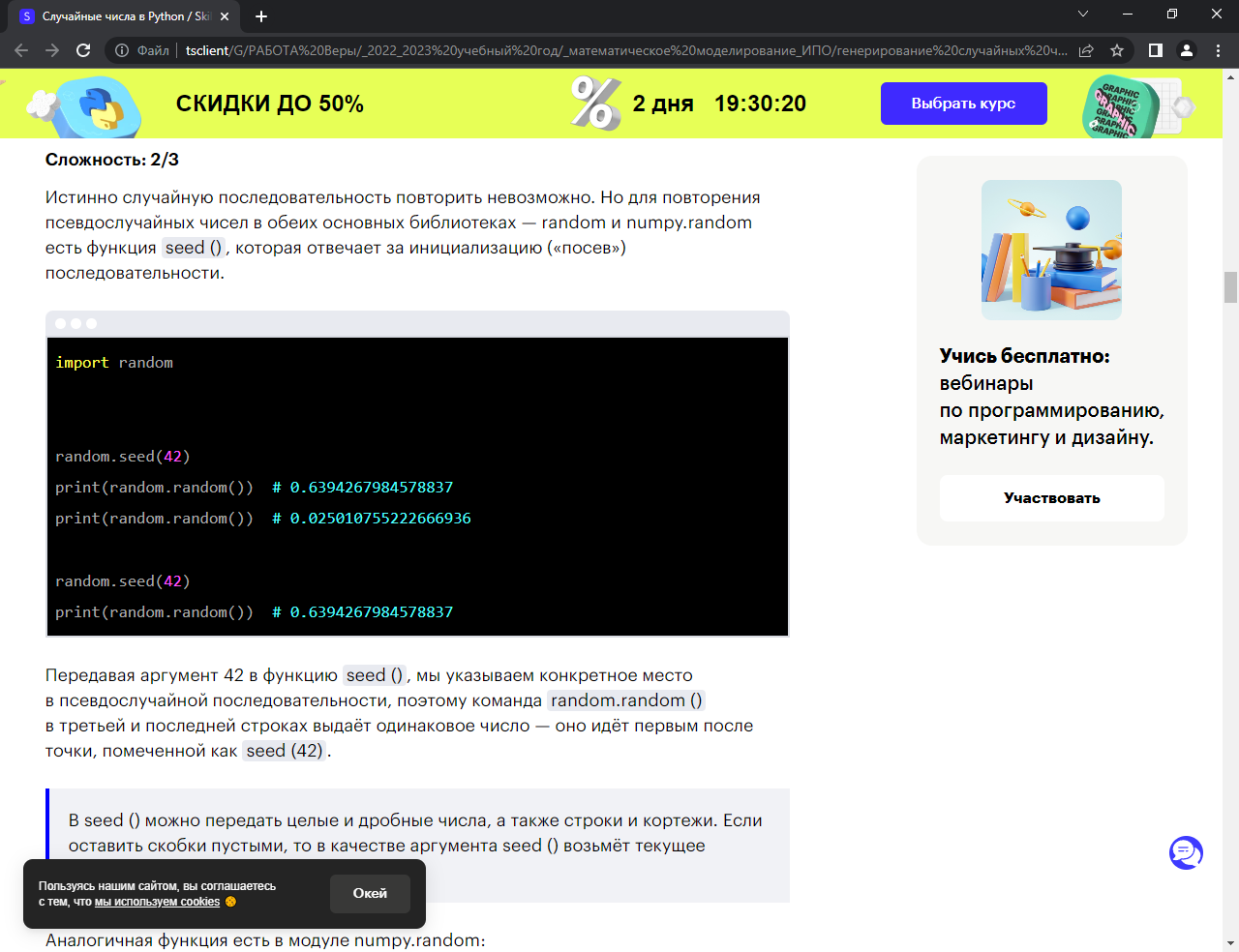


Рис. 3.

Передавая аргумент 42 в функцию seed (), мы указываем конкретное место в псевдослучайной последовательности, поэтому команда random.random () в третьей и последней строках выдаёт одинаковое число — оно идёт первым после точки, помеченной как seed (42).

В seed () можно передать целые и дробные числа, а также строки и кортежи. Если оставить скобки пустыми, то в качестве аргумента seed () возьмёт текущее системное время.

Аналогичная функция есть в модуле numpy.random (см. рис. 4):

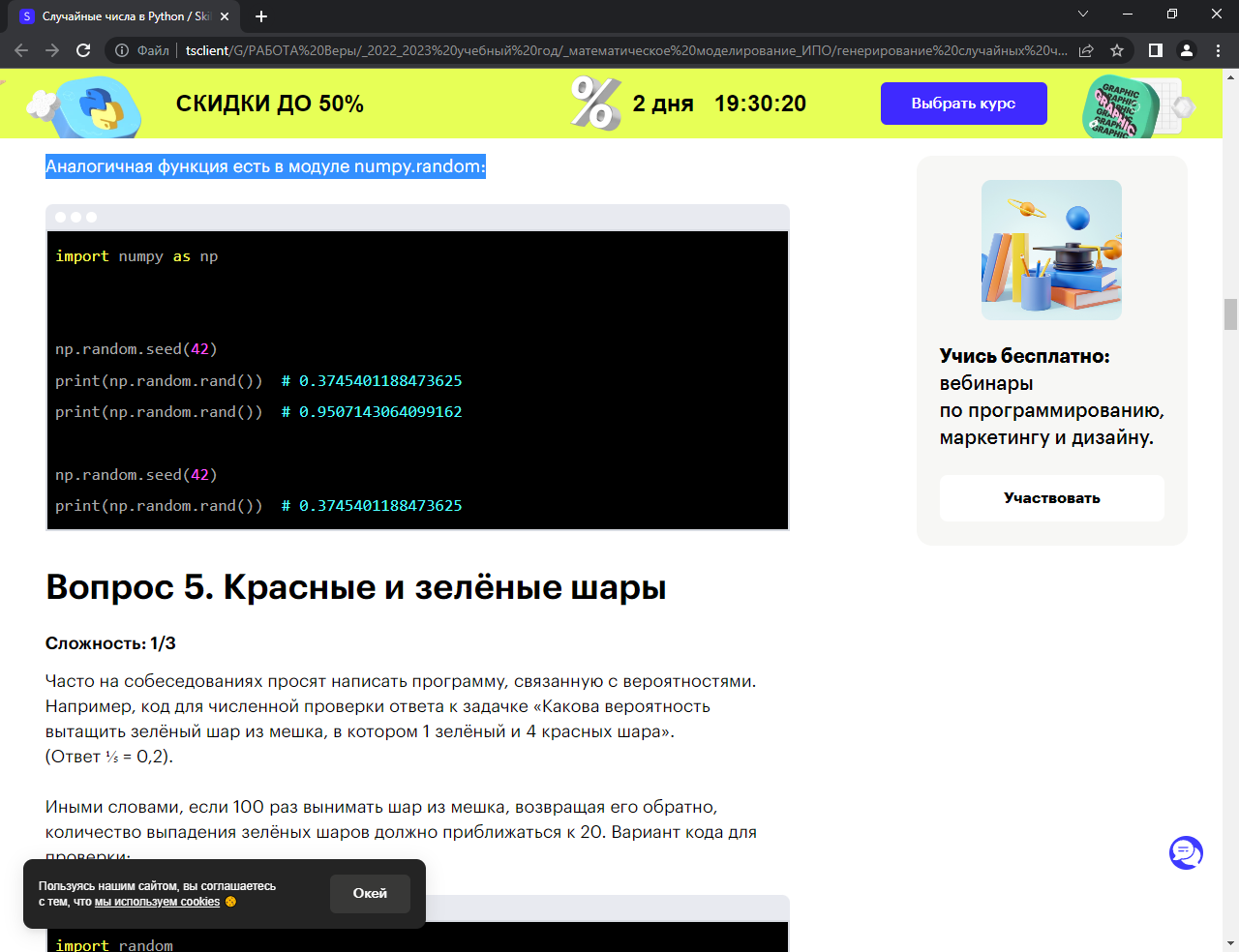


Рис. 4.

Например, код для численной проверки ответа к задачке «Какова вероятность вытащить зелёный шар из мешка, в котором 1 зелёный и 4 красных шара» представлен на рис. 5. (Ответ ⅕ = 0,2).

Иными словами, если 100 раз вынимать шар из мешка, возвращая его обратно, количество выпадения зелёных шаров должно приближаться к 20. Вариант кода для проверки:

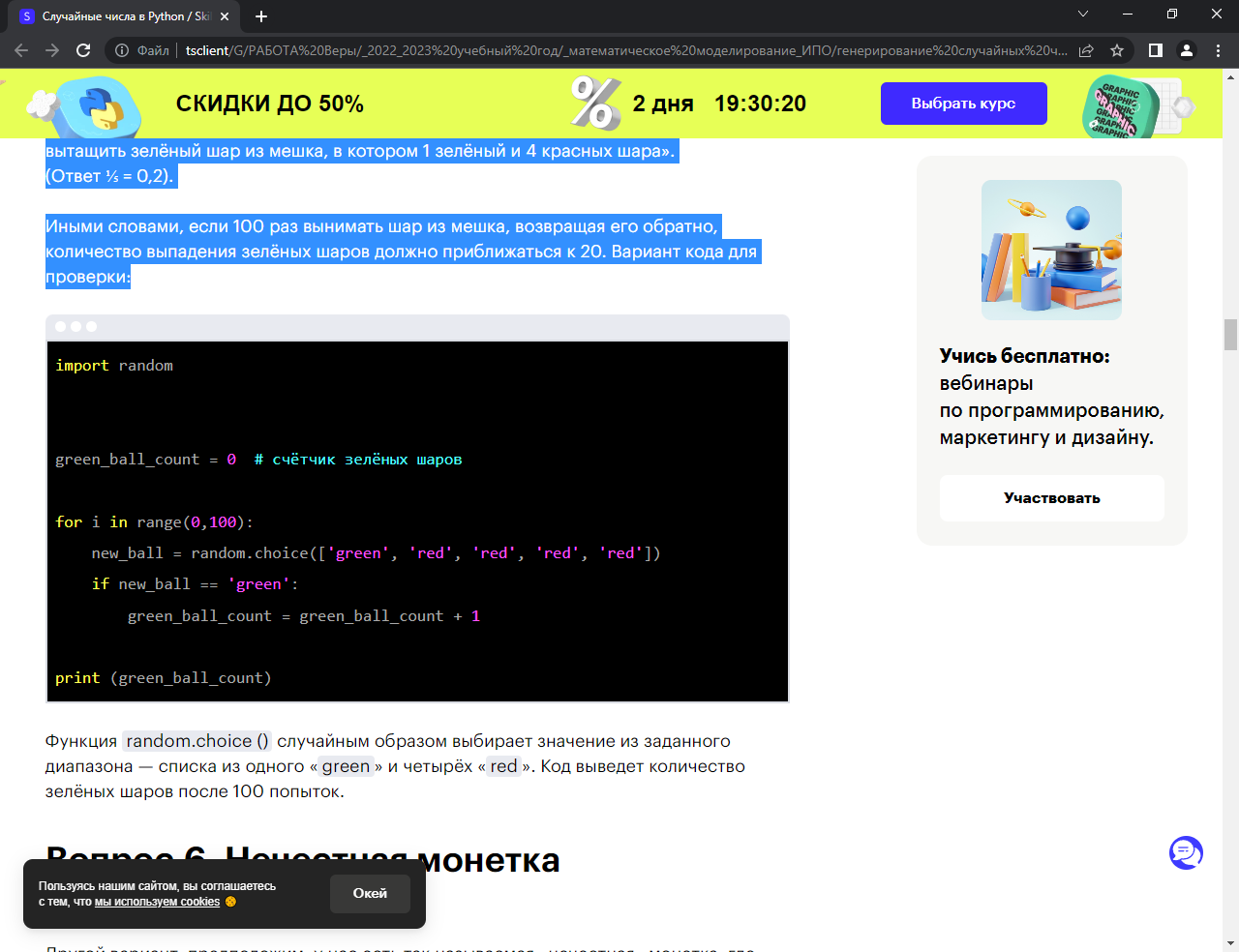


Рис. 5

Другой вариант: предположим, у нас есть так называемая «нечестная» монетка, где орёл (H, «heads») и решка (T, «tails») выпадают не с вероятностью ½, как положено, а по-другому: орёл с вероятностью p (H) = 0,2, а решка, соответственно, p (T) = 0,8.

Тогда код для проверки будет выглядеть примерно так (см. рис. 6):

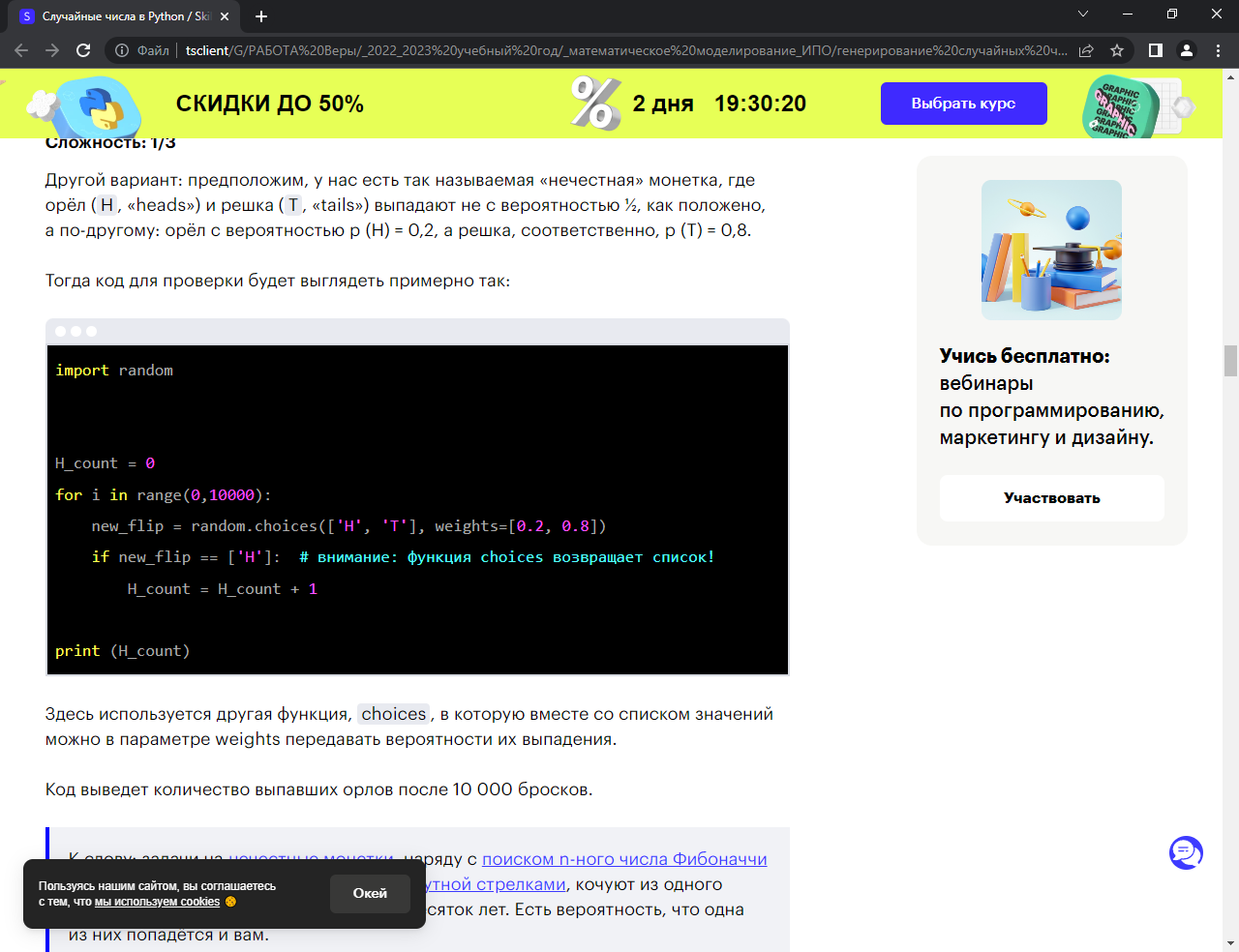


Рис. 6

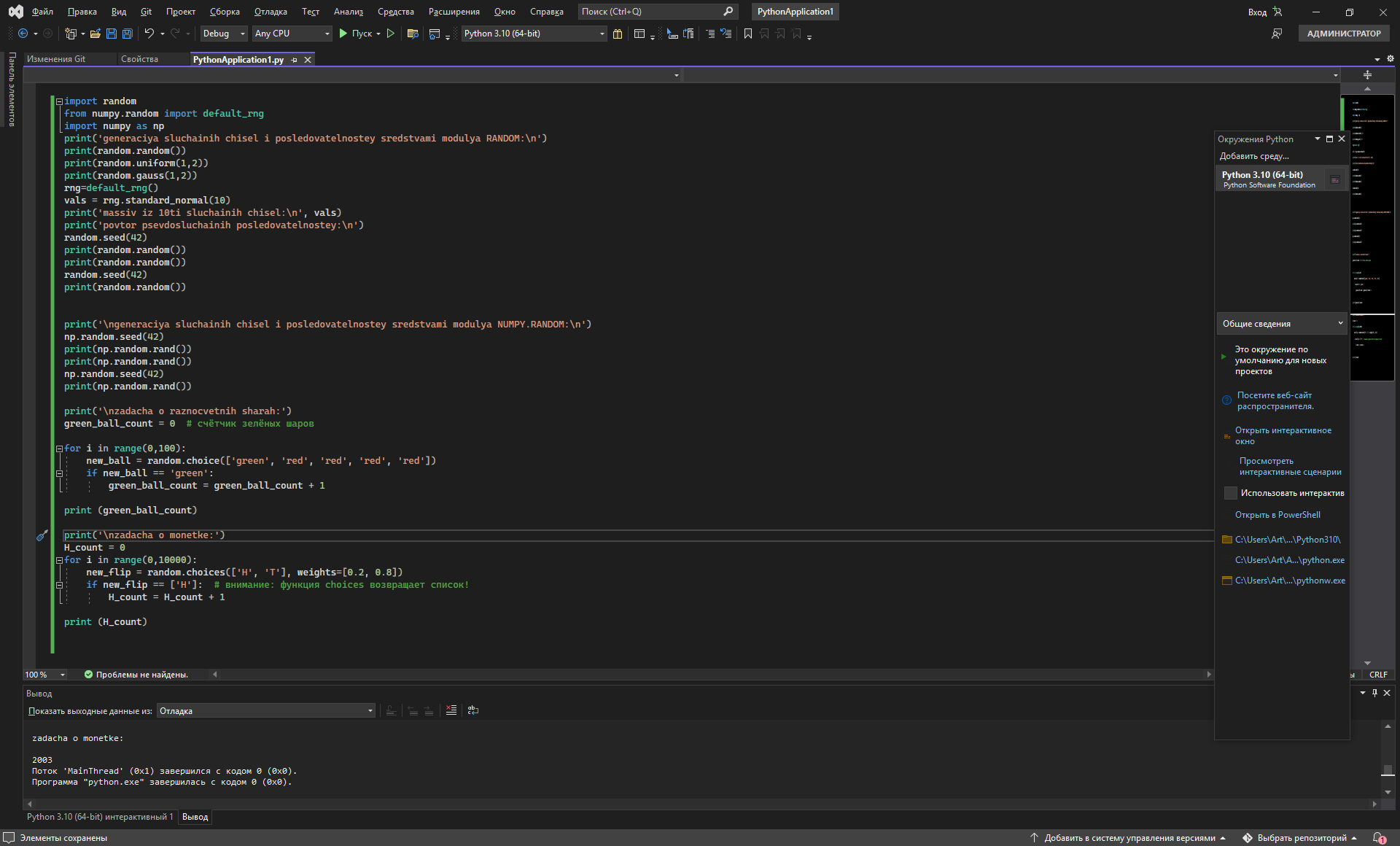


Рис. 7. Листинг, полученный в ходе проверки программного кода, приведенного в примерах, в IDE Microsoft Visual Studio.

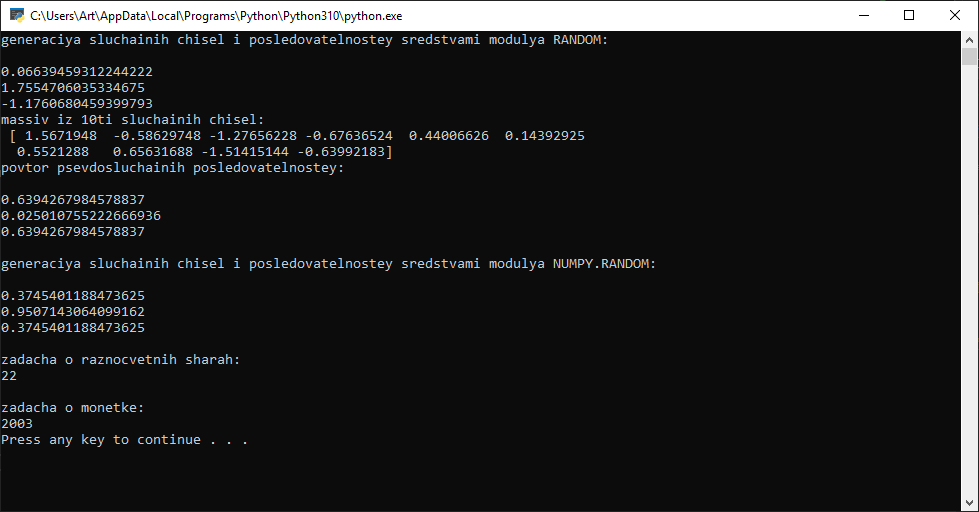


Рис. 8. Результаты работы программы.

1. Все приведенные примеры проверены в IDE Microsoft Visual Studio, см. рис. 7, 8. [↑](#footnote-ref-1)