| **К Г Э У** | МИНИСТЕРСТВО ВЫСШЕГО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  **Федеральное государственное бюджетное образовательное учреждение науки и высшего образования**  **«КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ ЭНЕРГЕТИЧЕСКИЙ УНИВЕРСИТЕТ»**  (ФГБОУ ВО «КГЭУ») |
| --- | --- |

**Кафедра Информатики и информационных управляющих систем**

**ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 1**

**дисциплина «Алгоритмизация и программирование»**

| **Выполнил:** | Банницин Д.А. |
| --- | --- |
| **Группа:** | ТРП-3-22 |
| **Проверил:** | Шорина Т.В. |

Казань, 2023

# Изучение работы функций библиотеки random.

Модуль random предоставляет функции для генерации случайных чисел, букв, случайного выбора элементов последовательности.

## Основные команды модуля random

| **random.seed**([X], version=2) | инициализация генератора случайных чисел. Если X не указан, используется системное время. |
| --- | --- |
| **random.getstate**() | внутреннее состояние генератора. |
| **random.setstate**(state) | восстанавливает внутреннее состояние генератора. Параметр state должен быть получен функцией getstate(). |
| **random.getrandbits**(N) | возвращает N случайных бит. |
| **random.randrange(start, stop, step)** | возвращает случайно выбранное число из последовательности. |
| **random.randint(A, B)** | случайное целое число N, A ≤ N ≤ B. |
| **random.choice(sequence)** | случайный элемент непустой последовательности. |
| **random.shuffle(sequence, [rand])** | перемешивает последовательность (изменяется сама последовательность). Поэтому функция не работает для неизменяемых объектов. |
| **random.sample(population, k)** | список длиной k из последовательности population. |
| **random.random()** | случайное число от 0 до 1. |
| **random.uniform(A, B)** | случайное число с плавающей точкой, A ≤ N ≤ B (или B ≤ N ≤ A). |
| **random.triangular(low, high, mode)** | случайное число с плавающей точкой, low ≤ N ≤ high. Mode - распределение. |
| **random.betavariate(alpha, beta)** | бета-распределение. alpha>0, beta>0. Возвращает от 0 до 1. |
| **random.expovariate(lambd)** | экспоненциальное распределение. lambd равен 1/среднее желаемое. Lambd должен быть отличным от нуля. Возвращаемые значения от 0 до плюс бесконечности, если lambd положительно, и от минус бесконечности до 0, если lambd отрицательный. |
| **random.gammavariate(alpha, beta)** | гамма-распределение. Условия на параметры alpha>0 и beta>0. |
| **random.gauss(значение, стандартное отклонение)** | распределение Гаусса. |
| **random.lognormvariate(mu, sigma)** | логарифм нормального распределения. Если взять натуральный логарифм этого распределения, то вы получите нормальное распределение со средним mu и стандартным отклонением sigma. mu может иметь любое значение, и sigma должна быть больше нуля. |
| **random.normalvariate(mu, sigma)** | нормальное распределение. mu - среднее значение, sigma - стандартное отклонение. |
| **random.vonmisesvariate(mu, kappa)** | mu - средний угол, выраженный в радианах от 0 до 2π, и kappa - параметр концентрации, который должен быть больше или равен нулю. Если каппа равна нулю, это распределение сводится к случайному углу в диапазоне от 0 до 2π. |
| **random.paretovariate(alpha)** | распределение Парето. |
| **random.weibullvariate(alpha, beta)** | распределение Вейбулла. |

## Примеры обработки данных с использованием модуля random

**В следующем примере показано, как получить случайно сгенерированное число в промежутке между 0 и 9.**

**Python**

| **1**  **2**  **3**  **4**  **5** | **from random import randint**      **print("Вывод случайного целого числа ", randint(0, 9))**  **print("Вывод случайного целого числа ", randrange(0, 10, 2))** |
| --- | --- |

**Вывод:**

**Shell**

| **Вывод случайного целого числа 5**  **Вывод случайного целого числа 2**  **Предположим, вам дан** [**python список**](https://python-scripts.com/lists-tuples-dictionaries#list) **городов, и вы хотите вывести на экран случайно выбранный элемент из списка городов. Посмотрим, как это можно сделать:**  **Python**   | **1**  **2**  **3**  **4**  **5** | **import random**      **city\_list = ['New York', 'Los Angeles', 'Chicago', 'Houston', 'Philadelphia']**  **print("Выбор случайного города из списка - ", random.choice(city\_list))** | | --- | --- |   **Вывод:**  **Shell**   | **1** | **Выбор случайного города из списка - Houston** | | --- | --- |   **Пример использования random.randint() в Python:**  **Python**   | **1**  **2**  **3**  **4**  **5**  **6** | **import random**      **print("Использование random.randint() для генерации случайного целого числа")**  **print(random.randint(0, 5))**  **print(random.randint(0, 5))** | | --- | --- |   **Вывод:**  **Shell**   | **1**  **2**  **3** | **Использование random.randint() для генерации случайного целого числа**  **4**  **2** | | --- | --- |   **Пример использования random.randrange() в Python:**  **Python**   | **1**  **2**  **3**  **4**  **5**  **6** | **import random**      **print("Генерация случайного числа в пределах заданного промежутка")**  **print(random.randrange(10, 50, 5))**  **print(random.randrange(10, 50, 5))** | | --- | --- |   **Вывод:**  **Shell**   | **1**  **2**  **3** | **Генерация случайного числа в пределах заданного промежутка**  **10**  **15** | | --- | --- |   **Пример использования random.sample() в Python:**  **Python**   | **1**  **2**  **3**  **4**  **5** | **import random**      **list = [2, 5, 8, 9, 12]**  **print ("random.sample() ", random.sample(list,3))** | | --- | --- |   **Вывод:**  **Shell**   | **1** | **random.sample() [5, 12, 2]** | | --- | --- |   **Пример использования random.choices() в Python:**  **Python**   | **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8** | **import random**      ***# Выборка с заменой***  **list = [20, 30, 40, 50 ,60, 70, 80, 90]**  **sampling = random.choices(list, k=5)**    **print("Выборка с методом choices ", sampling)** | | --- | --- |   **Вывод:**  **Shell**   | **1** | **Выборка с методом choices [30, 20, 40, 50, 40]** | | --- | --- |   **Пример использования random.seed() в Python:**  **Python**   | **1**  **2**  **3**  **4**  **5**  **6**  **7** | **import random**      **random.seed(6)**  **print("Случайное число с семенем ",random.random())**    **print("Случайное число с семенем ",random.random())** | | --- | --- |   **Вывод:**  **Shell**   | **1**  **2** | **Random number with seed 0.793340083761663**  **Random number with seed 0.793340083761663** | | --- | --- |   **Пример использования random.uniform() в Python:**  **Python**   | **1**  **2**  **3**  **4**  **5** | **import random**      **print("Число с плавающей точкой в пределах заданного промежутка")**  **print(random.uniform(10.5, 25.5))** | | --- | --- |   **Вывод:**  **Shell**   | **1**  **2** | **Число с плавающей точкой в пределах заданного промежутка**  **22.095283175159786** | | --- | --- |   **Пример использования random.triangular() в Python:**  **Python**   | **1**  **2**  **3**  **4**  **5** | **import random**      **print("Число с плавающей точкой через triangular")**  **print(random.triangular(10.5, 25.5, 5.5))** | | --- | --- |   **Вывод:**  **Shell**   | **1**  **2** | **Число с плавающей точкой через triangular**  **16.7421565549115** | | --- | --- |   **Пример криптографически надежной генерации данных в Python:**  **Python**   | **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8** | **import random**  **import secrets**      **number = random.SystemRandom().random()**  **print("Надежное число ", number)**    **print("Надежный токен байтов", secrets.token\_bytes(16))** | | --- | --- |   **Вывод:**  **Shell**   | **1**  **2**  **3** | **Надежное число 0.11139538267693572**    **Надежный токен байтов b'\xae\xa0\x91\*.\xb6\xa1\x05=\xf7+>\r;Y\xc3'** | | --- | --- |   **Использование numpy.random.rand() для генерации n-мерного массива случайных вещественных чисел в пределах [0.0, 1.0)**  **Использование numpy.random.uniform() для генерации n-мерного массива случайных вещественных чисел в пределах [low, high)**  **Python**   | **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8** | **import numpy**      **random\_float\_array = numpy.random.rand(2, 2)**  **print("2 X 2 массив случайных вещественных чисел в [0.0, 1.0] \n", random\_float\_array,"\n")**    **random\_float\_array = numpy.random.uniform(25.5, 99.5, size=(3, 2))**  **print("3 X 2 массив случайных вещественных чисел в [25.5, 99.5] \n", random\_float\_array,"\n")** | | --- | --- |   **Вывод:**  **Shell**   | **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8** | **2 X 2 массив случайных вещественных чисел в [0.0, 1.0]**  **[[0.08938593 0.89085866]**  **[0.47307169 0.41401363]]**    **3 X 2 массив случайных вещественных чисел в [25.5, 99.5]**  **[[55.4057854 65.60206715]**  **[91.62185404 84.16144062]**  **[44.348252 27.28381058]]** | | --- | --- |   **Для генерации случайного n-мерного массива целых чисел используется numpy.random.random\_integers():**  **Python**   | **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8** | **import numpy**      **random\_integer\_array = numpy.random.random\_integers(1, 10, 5)**  **print("1-мерный массив случайных целых чисел \n", random\_integer\_array,"\n")**    **random\_integer\_array = numpy.random.random\_integers(1, 10, size=(3, 2))**  **print("2-мерный массив случайных целых чисел \n", random\_integer\_array)** | | --- | --- |   **Вывод:**  **Shell**   | **1**  **2**  **3**  **4**  **5**  **6**  **7** | **1-мерный массив случайных целых чисел**  **[10 1 4 2 1]**    **2-мерный массив случайных целых чисел**  **[[ 2 6]**  **[ 9 10]**  **[ 3 6]]** | | --- | --- | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

# Изучение работы функций библиотеки **os** .

Модуль **os** предоставляет множество функций для работы с операционной системой, причём их поведение, как правило, не зависит от ОС, поэтому программы остаются переносимыми.

## Основные команды модуля os

| **os.name** | имя операционной системы. Доступные варианты: 'posix', 'nt', 'mac', 'os2', 'ce', 'java'. |
| --- | --- |
| **os.environ** | словарь переменных окружения. Изменяемый (можно добавлять и удалять переменные окружения). |
| **os.access(path, mode, \*, dir\_fd=None, effective\_ids=False, follow\_symlinks=True)** | проверка доступа к объекту у текущего пользователя. Флаги: **os.F\_OK** - объект существует, **os.R\_OK** - доступен на чтение, **os.W\_OK** - доступен на запись, **os.X\_OK** - доступен на исполнение. |
| **os.chdir(path)** | смена текущей директории. |
| **os.chmod(path, mode, \*, dir\_fd=None, follow\_symlinks=True)** | смена прав доступа к объекту (mode - восьмеричное число). |
| **os.chown(path, uid, gid, \*, dir\_fd=None, follow\_symlinks=True)** | меняет id владельца и группы (Unix). |
| **os.getcwd()** | текущая рабочая директория. |
| **os.getlogin**() | имя пользователя, вошедшего в терминал (Unix). |
| **os.getpid**() | текущий id процесса. |
| **os.uname**() | информация об ОС. возвращает объект с атрибутами: sysname - имя операционной системы, nodename - имя машины в сети (определяется реализацией), release - релиз, version - версия, machine - идентификатор машины. |
| **os.link(src, dst, \*, src\_dir\_fd=None, dst\_dir\_fd=None, follow\_symlinks=True)** | создаёт жёсткую ссылку. |
| **os.listdir(path=".")** | список файлов и директорий в папке. |
| **os.mkdir(path, mode=0o777, \*, dir\_fd=None)** | создаёт директорию. OSError, если директория существует. |
| **os.makedirs(path, mode=0o777, exist\_ok=False)** | создаёт директорию, создавая при этом промежуточные директории. |
| **os.remove(path, \*, dir\_fd=None)** | удаляет путь к файлу. |
| **os.rename(src, dst, \*, src\_dir\_fd=None, dst\_dir\_fd=None)** | переименовывает файл или директорию из src в dst. |

## Примеры обработки данных с использованием модуля os

**Использование метода os.getcwd()**

Этот метод ОС используется для получения пути текущей рабочей директории.

import os

os.getcwd()

​

'/Users/krnlnx'

**Использование метода os.chdir()**

Этот метод используется для изменения текущего каталога на новый, ему передается один параметр — путь к новому каталогу.

import os

os.getcwd()

​

'/Users/krnlnx'

​

os.chdir('Desktop')

os.getcwd()

​

'/Users/krnlnx/Desktop'

**Использование метода os.listdir()**

Этот метод возвращает все файлы и каталоги в текущем каталоге.

import os

os.listdir()

​

['testdir', 'testfile.py', 'testimage.png']

Также можно передать параметр конкретной директории для просмотра ее файлов и каталогов.

import os

os.listdir('/tmp/')

​

['fseventsd-uuid', 'MozillaUpdateLock-2656FF1E876E9973', 'powerlog', 'com.apple.launchd.791fHUaFBR']

**Использование метода os.mkdir()**

Этот метод используется для создания новой директории в текущей папке, например, давайте создадим директорию под названием Products.

import os

os.listdir()

​

['fseventsd-uuid', 'MozillaUpdateLock-2656FF1E876E9973', 'powerlog', 'com.apple.launchd.791fHUaFBR']

​

os.mkdir('Products')

os.listdir()

​

['fseventsd-uuid', 'MozillaUpdateLock-2656FF1E876E9973', 'Products', 'powerlog', 'com.apple.launchd.791fHUaFBR']

**Использование метода os.rename()**

Модуль используется для переименования файла или папки, ему передается два аргумента, старое и новое имя файла или папки. Например, переименуем папку Products в Stuff.

import os

os.listdir()

​

['fseventsd-uuid', 'MozillaUpdateLock-2656FF1E876E9973', 'Products', 'powerlog', 'com.apple.launchd.791fHUaFBR']

​

os.rename('Products', 'Stuff')

os.listdir()

​

['fseventsd-uuid', 'MozillaUpdateLock-2656FF1E876E9973', 'Stuff', 'powerlog', 'com.apple.launchd.791fHUaFBR']

**Использование метода os.rmdir()**

Этот метод используется для удаления каталогов. Имя каталога, который вы хотите удалить нужно передать первым аргументом. os.rmdir() будет работать только для пустых директорий.

Например, удалим каталог Stuff с помощью rmdir().

import os

os.listdir()

​

['fseventsd-uuid', 'MozillaUpdateLock-2656FF1E876E9973', 'Stuff', 'powerlog', 'com.apple.launchd.791fHUaFBR']

​

os.rmdir('Stuff')

os.listdir()

​

['fseventsd-uuid', 'MozillaUpdateLock-2656FF1E876E9973', 'powerlog', 'com.apple.launchd.791fHUaFBR']

**Использование метода os.remove()**

Этот метод используется для удаления файлов. Например, удалим файл powerlog.

import os

os.listdir()

​

['fseventsd-uuid', 'MozillaUpdateLock-2656FF1E876E9973', 'powerlog', 'com.apple.launchd.791fHUaFBR']

​

os.remove('powerlog')

os.listdir()

​

['fseventsd-uuid', 'MozillaUpdateLock-2656FF1E876E9973', 'com.apple.launchd.791fHUaFBR']

**Использование метода os.system()**

Этот метод используется для выполнения команды оболочки внутри Python приложения, как если бы вы были в оболочке. Например, давайте выполним команду «ls -1» с помощью системного метода.

import os

os.system('ls -1')

Вывод программы выглядит следующим образом

Plain Text

MozillaUpdateLock-2656FF1E876E9973

com.apple.launchd.791fHUaFBR

fseventsd-uuid

​

0

Цифра 0 в конце вывода означает что выполнение программы прошло без ошибок.

**Использование метода os.uname()**

Этот метод возвращает информацию, идентифицирующую текущую операционную систему.

import os

​

os.uname()

​

posix.uname\_result(sysname='Darwin', nodename='macbook.local', release='20.3.0', version='Darwin Kernel Version 20.3.0: Thu Jan 21 00:07:06 PST 2021; root:xnu-7195.81.3~1/RELEASE\_X86\_64', machine='x86\_64')

Использование метода os.environ

Environ не является методом в модуле ОС, а является параметром процесса, через который мы можем получить доступ к переменным окружения системы.

Например, давайте получим доступ к переменной окружения HOME

import os

​

os.environ['HOME']

​

'/Users/krnlnx'

Мы также можем установить собственную переменную окружения, как показано в примере ниже.

import os

​

os.environ['test\_key'] = "Test var"

os.environ['test\_key']

​

'Test var'

**Использование метода os.getuid()**

Этот модуль возвращает идентификатор пользователя текущего процесса.

import os

​

os.getuid()

​

501

**Использование метода os.getpid()**

Возвращает реальный идентификатор процесса текущего приложения.

import os

​

os.getpid()

​

4365

# Использованные источники

* [Python 3 для начинающих](https://pythonworld.ru/) Модуль os (https://pythonworld.ru/moduli/modul-cmath.html)
* [Python 3 для начинающих](https://pythonworld.ru/) Модуль random

(https://pythonworld.ru/moduli/modul-random.html)

* Егоров Егор Блог о разработке на Python

(<https://egorovegor.ru/python-os-module>)

* BestProg ([https://www.bestprog.net/ru/2021/10/27/python-the-cmath-module-working-with-complex-numbers-trigonometric-functions-ru/](https://www.bestprog.net/ru/sitemap_ru/))
* [Python 3](https://python-scripts.com/) Модуль random на примерах

(https://python-scripts.com/random)