Модуль os в Python

<https://all-python.ru/osnovy/os.html>

Модуль os в Python — это библиотека функций для работы с операционной системой. **Методы, включенные в неё позволяют определять тип операционной системы, получать доступ к переменным окружения, управлять директориями и файлами**:

* проверка существования объекта по заданному пути;
* определение размера в байтах;
* удаление;
* переименование и др.

При вызове функций os необходимо учитывать, что некоторые из них могут не поддерживаться текущей ОС.

Чтобы пользоваться методами из os, нужно подключить библиотеку. Для этого в Python используется import os, который необходимо описать в файле до первого обращения к модулю.

Рекомендуется использовать эту инструкцию в начале файла с исходным кодом.

**Функции модуля os**

Методы из библиотеки os могут применяться пользователем для разных целей. Ниже показаны наиболее популярные из них, позволяющие получать данные о операционной системе. Также получать сведения о файлах и папках, хранимых в памяти на жестком диске ПК.

**Получение информации об ОС**

Чтобы узнать имя текущей ОС, достаточно воспользоваться методом name. В зависимости от установленной платформы, он вернет ее короткое наименование в строковом представлении. Следующая программа была запущена на ПК с ОС Windows 10, поэтому результатом работы функции name является строка nt. Увидеть это можно при помощи обычного метода print.

import os

print(os.name)

nt

Получить сведения, которые касаются конфигурации компьютера, можно при помощи метода environ. Вызвав его через обращение к библиотеке os, пользователь получает большой [словарь](https://all-python.ru/osnovy/slovari.html) с переменными окружения, который выводится в консоль или строковую переменную. Таким образом, можно узнать название системного диска, адрес домашней директории, имя системы и массу другой информации. Следующий пример демонстрирует применение environ.

import os

print(os.environ)

environ({'ALLUSERSPROFILE': 'C:\\ProgramData', …})

При помощи функции getenv можно получить доступ к различным переменным среды. Чтобы сделать это, достаточно передать в качестве аргумента необходимое название переменной, как в следующем примере, где print выводит на экран сведения о TMP на дисплей в консоль.

import os

print(os.getenv("TMP"))

C:\Users\admin\AppData\Local\Temp

**Изменение рабочей директории**

По умолчанию рабочей директорией программы является каталог, где содержится документ с ее исходным кодом. Благодаря этому, можно не указывать абсолютный путь к файлу, если тот находится именно в этой папке. Получить сведения о текущей директории позволяет функция getcwd, которая возвращает полный адрес рабочего каталога на жестком диске. В следующем фрагменте кода показано что будет, если передать результат работы этого метода в print. Как можно заметить, рабочей директорией является каталог program на системном диске C.

import os

print(os.getcwd())

C:\Users\admin\source\repos\program

При желании, рабочую директорию можно настроить по своему усмотрению, применив метод chdir из библиотеки os. Для этого необходимо передать ему в качестве параметра абсолютный адрес к новому каталогу. Если указанного пути на самом деле не существует, программа будет завершена в аварийном режиме из-за выброшенного исключения. В следующем примере кода продемонстрирован переход к новой рабочей директории под названием folder на диске D.

import os

os.chdir(r"D:\folder")

D:\folder

**Проверка существования пути**

Чтобы избежать ошибок, связанных с отсутствием определенного файла или директории, которые должны быть обработаны программой, следует предварительно проверять их наличие с помощью метода exists. Передав ему в качестве аргумента путь к нужному файлу или папке, можно рассчитывать на лаконичный ответ в виде булевого значения true/false, сообщающего о наличии/отсутствии указанного объекта в памяти компьютера. В следующем примере идет проверка текстового файла test.txt из корневого каталога D, которая возвращает True.

import os

print(os.path.exists("D:/test.txt"))

True

Если объект на диске реально существует, это не всегда значит, что он имеет подходящую для дальнейшей обработки форму. **Проверить, является ли определенный объект файлом, поможет функция isfile**, которая принимает его адрес. Ознакомиться с результатом его работы можно из следующего примера, где print отображает на экране значение True для файла test.txt.

import os

print(os.path.isfile("D:/test.txt"))

True

Аналогичные действия можно выполнить и для проверки объекта на принадлежность к классу директорий, вызвав для его адреса метод isdir из библиотеки os. Как можно заметить, в данном случае print выводит на экран булево значение False, поскольку test.txt не является папкой.

import os

print(os.path.isdir("D:/test.txt"))

False

**Создание директорий**

Возможности модуля os позволяют не только отображать информацию об уже существующих в памяти объектах, но и генерировать абсолютно новые. Например, с помощью метода mkdir довольно легко создать папку, просто указав для нее желаемый путь. В следующем примере в корневом каталоге диска D производится новая папка под названием folder через mkdir.

import os

os.mkdir(r"D:\folder")

Однако на этом возможности по генерации директорий не заканчиваются. Благодаря функции makedirs можно создавать сразу несколько новых папок в неограниченном количестве, если предыдущая директория является родительской для следующей. Таким образом, в следующем примере показывается генерация целой цепочки папок из folder, first, second и third.

import os

os.makedirs(r"D:\folder\first\second\third")

**Удаление файлов и директорий**

**Избавиться от ненужного в дальнейшей работе файла можно с помощью метода remove**, отдав ему в качестве аргумента абсолютный либо относительный путь к объекту. В небольшом коде ниже демонстрируется удаление документа test.txt из корневой директории диска D на ПК.

import os

os.remove(r"D:\test.txt")

Чтобы **стереть из памяти папку, следует воспользоваться встроенной функцией rmdir**, указав ей адрес объекта. Однако здесь присутствуют определенные нюансы, поскольку программа не позволит беспрепятственно удалить директорию, в которой хранятся другие объекты. В таком случае на экране отобразится ошибка и будет выброшено исключение. В следующем примере производится процедура удаления пустой директории folder при помощи метода rmdir.

import os

os.rmdir(r"D:\folder")

**Для быстрого удаления множества пустых папок следует вызывать функцию removedirs**. Она предоставляет возможность избавиться сразу от нескольких каталогов на диске, при условии, что все они вложены друг в друга. Таким образом, указав путь к конечной папке, можно легко удалить все родительские директории, но только если они в результате оказываются пустыми. В примере показано мгновенное удаление четырех разных папок: folder, first, second, third.

import os

os.removedirs(r"D:\folder\first\second\third")

**Запуск на исполнение**

Встроенные функции библиотеки os позволяют запускать отдельные файлы и папки прямиком из программы. С этой задачей прекрасно справляется метод startfile, которому стоит передать адрес необходимо объекта. Программное обеспечение, используемое для открытия документа, определяется средой автоматически. Например, при запуске обычного файла test.txt, как это сделано в следующем примере, задействуется стандартный блокнот. Если передать функции ссылку на директорию, она будет открыта при помощи встроенного менеджера файлов.

import os

os.startfile(r"D:\test.txt")

**Получение имени файла и директории**

Иногда для взаимодействия с документом необходимо получить его полное имя, включающее разрешение, но не абсолютный путь к нему на диске. Преобразовать адрес объекта в название позволяет функция basename, которая содержится в подмодуле path из библиотеки os. Таким образом, следующий пример показывает преобразование пути test.txt в простое имя файла.

import os

print(os.path.basename("D:/test.txt"))

test.txt

Обратная ситуация возникает тогда, когда пользователю нужно получить только путь к файлу, без самого названия объекта. Это поможет сделать метод dirname, который возвращает путь к заданному документу в строковом представлении, как это продемонстрировано в небольшом примере ниже. Здесь print выводит на экран адрес текстового документа в папке folder.

import os

print(os.path.dirname("D:/folder/test.txt"))

D:/folder

**Вычисление размера**

Чтобы определить размер документа или папки, стоит воспользоваться функцией getsize, как это показано в следующем примере для файла test.txt. Функция print выводит размер данного документа в байтах. Воспользоваться getsize можно и для измерения объема директорий.

import os

print(os.path.getsize("D:\\test.txt"))

136226

**Переименование**

Библиотека os предоставляет возможность быстрой смены названия для любого файла или же каталога при помощи метода rename. Данная функция принимает сразу два разных аргумента. Первый отвечает за путь к старому наименованию документа, в то время как второй отвечает за его новое название. В примере показано переименование директории folder в catalog. Стоит помнить, что метод может генерировать исключение, если по указанному пути нет файла.

import os

os.rename(r"D:\folder", r"D:\catalog")

Переименовывать можно не только один каталог, но и несколько папок сразу, только если все они находятся в одной иерархической цепочке. Для этого достаточно вызвать метод renames и передать ему путь к конечной директории в качестве первого аргумента. В роли же второго параметра выступает аналогичный адрес к папке, но только с новыми именами всей цепочки. Следующий пример демонстрирует правильное использование функции renames, за счет чего было произведено переименование директорий folder, first и second в catalog, one и two.

import os

os.renames(r"D:\folder\first\second", r"D:\catalog\one\two")

**Содержимое директорий**

Проверить наличие в каталоге определенных объектов позволяет функция listdir. С её помощью можно получить информацию о файлах и папках в виде [списка](https://all-python.ru/osnovy/spiski.html). В программе немного ниже показано, как метод принимает в качестве параметра путь к каталогу folder на диске D, а затем выводит название внутренней папки first и документа test.txt, вывод в консоль осуществляется с помощью print.

import os

print(os.listdir(r"D:\folder"))

['first', 'test.txt']

Воспользовавшись методом walk, можно получить доступ к названиям и путям всех подпапок и файлов, относящихся к заданному каталогу. Применив один внешний цикл for, а также два вложенных, несложно получить информацию об объектах в каталоге folder через специальные списки directories и files. Сведения выдаются с помощью многократного обращения к print.

import os

for root, directories, files in os.walk(r"D:\folder"):

print(root)

for directory in directories:

print(directory)

for file in files:

print(file)

D:\folder

first

D:\folder\first

second

D:\folder\first\second

third

D:\folder\first\second\third

test.txt

**Информация о файлах и директориях**

Вывести на экран или в любое другое место основные сведения об объекте можно через метод stat. Поместив ему в качестве параметра расположение файла или папки на диске компьютера, стоит ожидать небольшой массив информации. Здесь можно найти данные о размере объекта в байтах, а также некие числовые значения, отображающие доступ и режим его работы.

import os

print(os.stat(r"D:\test.txt"))

os.stat\_result(st\_mode=33206, …)

**Обработка путей**

Возвращаясь к [классу](https://all-python.ru/osnovy/klassy.html) path из библиотеки os, стоит принять во внимание функцию split, позволяющую очень легко разъединять путь к файлу и имя файла в различные строки. Это демонстрируется на следующем примере с текстовым документом test.txt в папке folder.

import os

print(os.path.split(r"D:\folder\test.txt"))

('D:\\folder', 'test.txt')

Обратное действие выполняет функция join, позволяя легко соединить путь к документу с его названием. Как видно из результатов работы данного кода, благодаря print на экране будет отображаться путь, который ссылается на текстовый файл test.txt в каталоге folder на D.

import os

print(os.path.join(r"D:\folder", "test.txt"))

D:\folder\test.txt

Это были базовые возможности модуля os, реализуемые в программах на языке Python за счет множества встроенных методов по управлению установленной ОС. Таким образом, они дают возможность не только получать полезные сведения о платформе, но и работать с содержимым диска, создавая новые директории и файлы, переименовывая и полностью удаляя их.