

(!) "День чистоты". Плановая смена паролей. [Подробнее](#).



Если у вас возникли проблемы со входом или сменой пароля => **XMPP:**
admin@thesecure.biz | **TG:** t.me/xssadmin

YALE LODGE | HIGH QUALITY CVV SHOP

Servers/VDS for pentest and scanning!

YALE LODGE | HIGH QUALITY CVV SHOP

Программирование. Разработка. > **Python** >

Python с абсолютного нуля. Учимся работать со строками, файлами и интернетом

flantod · 12.07.2021

Перейти к новому

Отслеживать

flantod

floppy-диск

Пользователь

12.07.2021

Новое



#1

Содержание статьи

- Форматируем строки
- Способ 1 — с помощью метода .format()
- Способ 2 — через f-строки
- Файлы
- Работа с вебom
- Обработка ошибок
- Пишем сканер портов
- Домашнее задание

Однажды крокодилу Гене и Чебурашке поручили написать сочинение на тему «Как я провел лето». Проблема была в том, что все лето друзья пили пиво. Гена, не умеющий врать, так и написал, поэтому Чебурашке пришлось заменить некоторые слова. А поскольку Чебурашка был кодером на питоне, то сделал он это при помощи строковой функции. В этой статье я покажу, как не отставать от Чебурашки и научиться работать со строками, файлами и делать запросы к веб-сайтам на Python.

Код:

Скопировать в буфер обмена

От редакции

Недавно мы провели опрос среди читателей и выяснили, что многие хотели бы изучить Python, при-

Эта статья, как и предыдущая, доступна без платной подписки, так что смело делись этими ссылк

Начнем со строк. Чтобы решить вставшую перед друзьями проблему, Чебурашка использовал функцию `replace()`, которая заменяет в строке одну подстроку другой.

Сначала он объявил переменную `s` и поместил туда строку, которую прислал ему Гена.

Python:

Скопировать в буфер обмена

```
s = 'Все лето мы пили пиво. Вот как-то открываю дверь, а на пороге Чебурашка, весь такой пьянь
```

Дальше Чебурашка определил словарь из слов, которые требовалось заменить.

Python:

Скопировать в буфер обмена

```
slova = {'пили': 'читали', 'пиво': 'книги', 'пьяный': 'начитанный', 'бутылка': 'энциклопедия'}
```

И теперь при помощи цикла `for` Чебурашка перебрал словарь, чтобы заменить каждое из слов (`key`) на соответствующее значение из словаря (`slova[key]`):

Python:

Скопировать в буфер обмена

```
for key in slova:  
    s = s.replace(key, slova[key])  
print(s)
```

INFO

Словари во многом похожи на списки, но значения в них записаны парами: ключ и значение. По ключу можно узнать значение. Можно считать, что в списках ключи — это индексы (0, 1, 2...), а в словарях — строки.

Функцию `replace()` удобно использовать, чтобы начисто удалить какие-то слова из строки. Для этого будем заменять их пустой строкой (если открыть и закрыть кавычку, то получится пустая строка):

Python:

Скопировать в буфер обмена

```
s = '''Я не люблю пить пиво.  
Оно невкусное и бесполезное!'''  
s = s.replace('не', '')  
print(s)
```

INFO

Чтобы записать в переменную несколько строк, можно обернуть их в три одинарные кавычки и делать переносы прямо в коде.

Чтобы получить количество символов в строке, используется функция `len()`.

Python:

[Скопировать в буфер обмена](#)

```
s = 'Если очень вам не понравится, код пишите как придется!'  
n = len(s)  
print(n)
```

И, как я уже рассказывал в прошлой статье, от строк можно брать срезы как от массивов, если указать начало и конец подстроки в квадратных скобках после переменной. Позиция начинается с нуля.

Python:

[Скопировать в буфер обмена](#)

```
s = 'Меня зовут Бонд, Джеймс Бонд'  
a = s[11:15]  
print('Фамилия: ' + a)
```

Если нужно сделать срез с начала строки, первую цифру можно не писать.

Предположим, тебе нужно найти в списке строки, которые начинаются на `https`. Перебираем их с помощью `for`, для каждой проверяем, совпадают ли первые пять знаков со строкой `https`, и если да, то выводим строку:

Python:

[Скопировать в буфер обмена](#)

```
mas = [ 'Это просто строка', 'https://xakep.ru', 'Еще одна строка', 'https://habr.ru' ]  
for x in mas:  
    if x[:5] == 'https':  
        print(x)
```

Чтобы посчитать количество вхождений подстроки в строку, можно использовать метод `.count()`:

Python:

[Скопировать в буфер обмена](#)

```
s = 'Прикинь, короче, я такой, короче, ему бах эксплоитом по порту, а он, короче, упал сразу!'
n = s.count('короче')
print(n)
```

Иногда в начале или в конце строки могут быть лишние пробелы или переносы строк. Давай удалим их специальной командой `.strip()`:

Python:

Скопировать в буфер обмена

```
s = ' Пива много не бывает! \n'
s = s.strip()
print(s)
```

INFO

Переносы строк можно добавить с помощью символов `\n` (используется во всех ОС) либо `\r\n` (в Windows). Есть и другие спецсимволы. Например, `\t` — знак табуляции.

Чтобы определить наличие подстроки в строке `s`, можно использовать метод `.find()`:

Python:

Скопировать в буфер обмена

```
n = s.find('строка, которую ищем')
```

Если искомая подстрока найдена, то в переменную `n` попадет ее позиция в строке, а если не найдена, `n` станет равной `-1`.

Давай попробуем определить, есть ли в строке адрес электронной почты с Хакер.ru, то есть будем искать подстроку `@хакер.ru`.

Но сначала нам понадобится еще один строковый метод — `.split()`. Он позволяет разделить строку на части, указав в качестве аргумента строку-разделитель. Например, `s.split("\n")` разделит текст на абзацы по символу переноса строки. Если же оставить скобки пустыми, то будет использован разделитель по умолчанию — пробел.

Python:

Скопировать в буфер обмена

```
s = 'Это обычная строка, а в ней адрес почты vasya@xakep.ru'
words = s.split()
for w in words:
    n = w.find('@xakep.ru')
    if n != -1:
        print('Найден e-mail: ' + str(w) + ' в позиции ' + str(n))
```

Метод `join()` позволяет, наоборот, склеивать строки. Он принимает список и возвращает строку,

где каждый элемент списка соединен с другим через строку, у которой ты вызвал этот метод.

Python:

Скопировать в буфер обмена

```
s = 'вирус внедряется '  
list1 = ['раз, ', 'два, ', 'три...']  
print(s + s.join(list1))
```

ФОРМАТИРУЕМ СТРОКИ

Мы не раз печатали разные вещи, соединяя строки простым сложением. Это не всегда удобно, особенно учитывая, что если попадутся числа, то их придется переводить в строки функцией `str()`. Есть более красивый и удобный способ подставлять значения переменных внутрь строк. Точнее, два немного разных способа.

Способ 1 — с помощью метода `.format()`

Мы можем вставить в строку парные фигурные скобки, а затем вызвать строковый метод `.format()` и передать ему нужные значения в порядке их подстановки в строку.

Python:

Скопировать в буфер обмена

```
name = 'Вася Пупкин'  
age = 20  
address = 'улица Пушкина, дом Колотушкина'  
info = 'Имя: {}. Возраст: {}. Адрес: {}'.format(name, age, address)  
print(info)
```

Можно передать информацию списком через звездочку:

Python:

Скопировать в буфер обмена

```
data = ['Вася Пупкин', 20, 'улица Пушкина, дом Колотушкина']  
info = 'Имя: {}. Возраст: {}. Адрес: {}'.format(*data)  
print(info)
```

Способ 2 — через f-строки

Другой вариант — написать букву `f` перед строкой и затем в фигурных скобках указывать непосредственно переменные.

Python:

Скопировать в буфер обмена

```
name = 'Вася Пупкин'  
age = 20
```

```
address = 'улица Пушкина, дом Колотушкина'  
info = f'Имя: {name.upper()}. Возраст: {age}. Адрес: {address}'  
print(info)
```

Главное преимущество этого способа в том, что ты можешь вставить значение в строку несколько раз. К тому же можно менять значения прямо в фигурных скобках: сперва Python выполнит все действия в них, а затем подставит полученный результат в строку. Так, метод `.upper()` в примере выше делает все буквы заглавными.

ФАЙЛЫ

Перечисленных методов достаточно, чтобы ты мог делать со строками что угодно. Но откуда эти строки возьмутся? Чаще всего они записаны в файлах, поэтому сейчас я расскажу, как в Python с ними управляться.

Чтобы работать с файлом, его нужно открыть. Для этого служит функция `open()`, а работает она вот так:

Python:

Скопировать в буфер обмена

```
f = open('имя файла с путем и расширением', 'режим работы с файлом', encoding='Кодировка текста')
```

Режимов работы с файлами несколько, но тебя интересует в основном:

- `r` — открыть файл для чтения из него информации;
- `w` — открыть файл для записи в него информации (создает новый файл);
- `a` — открытие файла для дозаписи информации в конец файла (дописывает информацию в конец существующего файла);
- `a+` — дозапись и чтение.

Чтобы избежать проблем с путями в Windows, используй в них двойной слеш `'\\'`, а также перед открывающей кавычкой пути файла ставь букву `u`, указывающую на то, что строка в кодировке Unicode:

Python:

Скопировать в буфер обмена

```
f = open(u'D:\\test.txt', 'r', encoding='UTF-8')
```

Читать строки из файла можно методом `.read()`:

Python:

Скопировать в буфер обмена

```
f = open('test.txt', 'r', encoding='UTF-8')  
s = f.read()  
print(s)
```

Как вариант — можно последовательно читать из файла отдельные строки с помощью цикла `for`:

Python:

Скопировать в буфер обмена

```
f = open('test.txt', 'r', encoding='UTF-8')
for x in f:
    print(x)
```

После того как работа с файлом закончена, нужно закрыть его.

Python:

Скопировать в буфер обмена

```
f.close()
```

INFO

Для работы с бинарными файлами при открытии файла добавь к режиму букву `b`:

Python:

Скопировать в буфер обмена

```
f = open('myfile.bin', 'rb')
d = f.read()
print("d = ", d)
```

Подробнее о бинарных данных мы поговорим в одной из следующих статей.

Давай теперь попробуем создать новый текстовый файл в одном каталоге с нашим скриптом и записать в него значения каких-то переменных.

Python:

Скопировать в буфер обмена

```
s1 = 'Раз, два, три, четыре, пять\n'
s2 = 'Я иду сервак ломать...\n'
f = open('poems.txt', 'w', encoding='UTF-8')
f.write(s1)
f.write(s2)
f.close()
```

Обрати внимание, что в конце каждой строки стоит символ `\n` — переход на новую строку.

Допустим, ты хочешь дописать третью строчку в конец этого файла. Тут-то и пригодится режим дозаписи!

Python:

Скопировать в буфер обмена

```
s3 = 'Ох, устанут поднимать!\n'  
f = open('poems.txt', 'a', encoding='UTF-8')  
f.write(s3)  
f.close()
```

Для открытия файлов также очень удобно использовать конструкцию `with open('имя файла с путем и расширением', 'режим работы с файлом') as f`, потому что благодаря слову `with` файл закроется автоматически и тебе не придется думать об этом.

Python:

Скопировать в буфер обмена

```
s = 'Если вы закроете этот файл, ваш диск будет отформатирован!\nШутка\n'  
with open('test.txt', 'w', encoding='UTF-8') as f:  
    f.write(s)
```

РАБОТА С ВЕБОМ

Давай научимся получать информацию с веб-страниц. Для начала нужно установить несколько модулей. Пишем в командной строке:

```
pip install requests  
pip install html2text
```

Модуль `requests` позволяет делать GET- и POST-запросы к веб-страницам.

Модуль `html2text` служит для преобразования HTML-кода веб-страниц в обычный текст, то есть чистит его от тегов HTML.

Импортируем наши новые модули в начале программы и попробуем получить какую-нибудь страницу из интернета.

Python:

Скопировать в буфер обмена

```
import requests  
# Делаем GET-запрос  
s = requests.get('http://xakep.ru')  
# Печатаем код ответа сервера  
print(s.status_code)  
# Печатаем HTML-код  
print(s.text)
```

Программа напечатает много HTML-кода, из которого состоит главная страница журнала. Но что, если тебе нужен только текст сайта, а не мешанина из тегов? Здесь поможет `html2text`. Он выделит из кода текст, заголовки и картинки и отдаст их уже без HTML-тегов.

Python:

Скопировать в буфер обмена

```
import requests
import html2text
# Делаем GET-запрос
s = requests.get('http://xakep.ru')
# Код ответа сервера
print(s.status_code)
# Создается экземпляр парсера
d = html2text.HTML2Text()
# Параметр, влияющий на то, как парсятся ссылки
d.ignore_links = True
# Текст без HTML-тегов
c=d.handle(s.text)
print(c)
```

Кроме GET-запросов, существуют так называемые POST-запросы, которые применяются для отсылки на сервер больших текстов или каких-то файлов. Если видишь на сайте форму, особенно с загрузкой файла, значит, скорее всего, при нажатии на кнопку «Отправить» будет сделан POST-запрос.

Библиотека requests тоже позволяет делать POST-запросы. Тебе это может пригодиться для имитации действий пользователя — например, если нужно автоматизировать работу с сайтом. Можешь даже использовать это в качестве самописного аналога Burp!

Давай посмотрим, как послать обычный POST-запрос. Предположим, на сайте site.ru существует скрипт guest.php, который POST-запросом принимает от формы имя пользователя name и сообщение message, а затем постит их в гостевую книгу.

Python:

Скопировать в буфер обмена

```
import requests
# Переменные, которые нужно отправить POST-запросом
user = 'coolhacker'
message = 'You have beeh pwned!!!'
# Делаем POST-запрос и передаем словарь из полей
r = requests.post("http://site.ru/guest.php", data={'user': user, 'message': message})
print(r.status_code)
```

Теперь давай отправим запрос с файлом payload.php во вложении и теми же двумя полями формы, что и в предыдущем запросе. Файл придет на сервер под именем misc.php.

Python:

Скопировать в буфер обмена

```
import requests

user = 'kitty2007'
message = '(* ^ w ^)'
```

```
# Открываем файл в бинарном режиме
with open('payload.php', 'rb') as f:
    # POST-запрос с отправкой файла
    r = requests.post('http://site.ru/upload.php', files={'misc.php': f}, data={'user': user,
```

Осталось научиться скачивать файлы. Это во многом похоже на запрос страниц, но делать это лучше в потоковом режиме (`stream=True`). Также нам понадобится модуль `shutil`, в котором есть удобная функция `copyfileobj`. Она позволяет копировать содержимое двоичных файлов — в нашем случае из интернета к нам на диск.

Python:

Скопировать в буфер обмена

```
import requests
import shutil
import os

# Файл, который надо скачать
s = 'https://xakep.ru/robots.txt'

# С помощью функции os.path.split(s) вытаскиваем из строки путь к файлу и его имя
dirname, filename = os.path.split(s)

# GET-запрос в режиме stream=True для скачивания файла
r = requests.get(s, stream=True)

# Если ответ сервера удачен (200)
if r.status_code == 200:
    # Создаем файл и открываем его в бинарном режиме для записи
    with open(filename, 'wb') as f:
        # Декодируем поток данных на основе заголовка content-encoding
        r.raw.decode_content = True
        # Копируем поток данных из интернета в файл с помощью модуля shutil
        shutil.copyfileobj(r.raw, f)
```

INFO

Коды ответа сервера помогают понять, как прошел запрос. Код 200 означает, что сервер успешно обработал запрос и отдал нам ответ, код 404 — страница не была найдена, 500 — внутренняя ошибка сервера, 503 — сервер недоступен и так далее. Полный список кодов ты найдешь в Википедии.

ОБРАБОТКА ОШИБОК

Прежде чем разбирать более реальный пример, я должен показать тебе еще одну языковую конструкцию, которая незаменима при работе с файлами и сетью. Это обработка исключительных ситуаций, то есть ошибок.

Часто при работе программы компьютер сталкивается с разными проблемами. Например, файл

не найден, сеть недоступна, кончилось место на диске. Если программист об этом не позаботился, то интерпретатор Python просто завершит работу с ошибкой. Но есть способ предусмотреть неурядицы прямо в коде и продолжать работу — конструкция `try... except`.

Выглядит она вот так:

```
Python:
try:
```

Скопировать в буфер обмена

~/ XSS.is

Наши действия, если ошибка произошла

Можно ловить конкретные типы ошибок, если после слова `except` указать название типа. К примеру, `KeyboardInterrupt` срабатывает, если пользователь пытается завершить программу, нажав `Ctrl-C`. В нашей власти запретить это делать!

Да что там, мы можем даже разрешить делить на ноль, если отловим ошибку `ZeroDivisionError`. Вот как это будет выглядеть:

```
Python:
try:
    k = 1 / 0
except ZeroDivisionError:
    k = 'over 9000'
print(k)
```

Скопировать в буфер обмена

WWW

Полный список видов исключений

ПИШЕМ СКАНЕР ПОРТОВ

А теперь мы напишем собственный сканер портов! Он будет простеньким, но вполне рабочим. Поможет нам в этом модуль `socket`, где реализована работа с сокетами.

INFO

Сокет — это интерфейс обмена данными между процессами. Существуют клиентские и серверные сокеты. Серверный сокет слушает определенный порт в ожидании подключения клиентов, а клиентский подключается к серверу. После того как было установлено соединение, начинается обмен данными.

Вот как будет выглядеть код.

Python:

Скопировать в буфер обмена

```
import socket

# Список портов для сканирования
ports = [20, 21, 22, 23, 25, 42, 43, 53, 67, 69, 80, 110, 115, 123, 137, 138, 139, 143, 161, 179]

host = input('Введи имя сайта без http/https или IP-адрес: ')
print ("Ождай, идет сканирование портов!")

# В цикле перебираем порты из списка
for port in ports:
    # Создаем сокет
    s = socket.socket()
    # Ставим тайм-аут в одну секунду
    s.settimeout(1)
    # Ловим ошибки
    try:
        # Пробуем соединиться, хост и порт передаем как список
        s.connect((host, port))
        # Если соединение вызвало ошибку
    except socket.error:
        # тогда ничего не делаем
        pass
    else:
        print(f"{host}: {port} порт активен")
    # Закрываем соединение
    s.close

print ("Сканирование завершено!")
```

Как видишь, ничего сложного!

ДОМАШНЕЕ ЗАДАНИЕ

1. Сделай, чтобы сканер портов получал список IP из одного файла, а результаты сканирования записывал в другой.
2. В прошлой статье ты научился работать с буфером обмена. Напиши программу, которая постоянно запущена и периодически получает содержимое буфера обмена. Если оно изменилось, то дописывает его в конец файла monitoring.txt. Попробуй записывать в лог только те перехваченные строки, в которых есть латинские буквы и цифры, так более вероятно поймать пароли.
3. Напиши программу, которая читает файл такого вида:

Код:

Скопировать в буфер обмена

```
Иван Иванов|ivanov@mail.ru|Password123
Дима Лапушок|superman1993@xakep.ru|1993superman
Вася Пупкин|pupok@yandex.ru|qwerty12345
Фродо Бэггинс|Frodo@mail.ru|MoRdOr100500
```

Кевин Митник | kevin@xakep.ru | dontcrackitplease
Юзер Юзерович | uswer@yandex.ru | aaaa321

Программа должна сортировать строки по доменам из email, для каждого домена создавать файл и в каждый файл помещать список почтовых адресов.

4. Напиши программу, которая проходит сайты по списку, скачивает файлы robots.txt и sitemap.xml и сохраняет на диск. В случае если файл не найден, выводится сообщение об этом.

На сегодня всё. Из следующей статьи ты узнаешь, как работать с файловой системой ОС, разберешься с функциями, познаешь силу регулярных выражений и напишешь простой сканер SQL-уязвимостей. Не пропусти!

Автор: Иван Сараев

Источник: <https://xakep.ru/2021/07/07/python-for-newbies-2/>

Последнее редактирование: 12.07.2021

🔔 Жалоба

👍 Like + Цитата ↩ Ответ

qwerty0984, N0N1k и solydol

В I U ☒ T A : ☞ </> >_ 🔒 🔍 🔗 😊 🖼 📎
🔗 ↶ ↷ [] 📁 🔍

Напишите ответ...

📎 Прикрепить файлы

↶ Ответ

Программирование. Разработка. > **Python** >

Выбор стиля Русский (RU)

Помощь Главная 📶