

PYTHON ДЛЯ СЕТЕВЫХ ИНЖЕНЕРОВ

РАБОТА С ФАЙЛАМИ

ОТКРЫТИЕ ФАЙЛОВ

ОТКРЫТИЕ ФАЙЛОВ

Для начала работы с файлом, его надо открыть.

```
file = open('file_name.txt', 'r')
```

В функции open():

- 'file_name.txt' - имя файла
 - тут можно указывать не только имя, но и путь (абсолютный или относительный)
- 'r' - режим открытия файла

Функция open() создает объект file, к которому потом можно применять различные методы, для работы с ним.

РЕЖИМЫ ОТКРЫТИЯ ФАЙЛОВ

- r - открыть файл только для чтения (значение по умолчанию)
- r+ - открыть файл для чтения и записи
- w - открыть файл для записи
 - если файл существует, то его содержимое удаляется. Если файла нет, создается новый
- w+ - открыть файл для чтения и записи
 - если файл существует, то его содержимое удаляется. Если файла нет, создается новый
- a - открыть файл для дополнение записи. Данные добавляются в конец файла
- a+ - открыть файл для чтения и записи. Данные добавляются в конец файла

ЧТЕНИЕ ФАЙЛОВ

ЧТЕНИЕ ФАЙЛОВ

В Python есть несколько методов чтения файла:

- `read()` - считывает содержимое файла в строку
- `readline()` - считывает файл построчно
- `readlines()` - считывает строки файла и создает список из строк

ЧТЕНИЕ ФАЙЛОВ

Посмотрим как считывать содержимое файлов, на примере файла r1.txt:

```
!  
service timestamps debug datetime msec localtime show-timezone year  
service timestamps log datetime msec localtime show-timezone year  
service password-encryption  
service sequence-numbers  
!  
no ip domain lookup  
!  
ip ssh version 2  
!
```


ЧТЕНИЕ ФАЙЛОВ. МЕТОД READ()

Метод `read()` - считывает весь файл в одну строку.

Пример использования метода `read()`:

```
In [1]: f = open('r1.txt')

In [2]: f.read()
Out[2]: '!\\nservice timestamps debug datetime msec localtime show-timezone year\\nservice timestamps log date

In [3]: f.read()
Out[3]: ''
```

ЧТЕНИЕ ФАЙЛОВ. МЕТОД READLINE()

Построчно файл можно считать с помощью метода `readline()`:

```
In [4]: f = open('r1.txt')
```

```
In [5]: f.readline()
```

```
Out[5]: '!\\n'
```

```
In [6]: f.readline()
```

```
Out[6]: 'service timestamps debug datetime msec localtime show-timezone year\\n'
```

ЧТЕНИЕ ФАЙЛОВ. МЕТОД READLINE()

Но, чаще всего, проще пройтись по объекту file в цикле, не используя методы read...:

```
In [7]: f = open('r1.txt')

In [8]: for line in f:
...:     print(line)
...:
!

service timestamps debug datetime msec localtime show-timezone year

service timestamps log datetime msec localtime show-timezone year

service password-encryption

service sequence-numbers

!

no ip domain lookup

!

ip ssh version 2

!
```

ЧТЕНИЕ ФАЙЛОВ. МЕТОД READLINES()

Еще один полезный метод - `readlines()`. Он считывает строки файла в список:

```
In [9]: f = open('r1.txt')

In [10]: f.readlines()
Out[10]:
['!\n',
 'service timestamps debug datetime msec localtime show-timezone year\n',
 'service timestamps log datetime msec localtime show-timezone year\n',
 'service password-encryption\n',
 'service sequence-numbers\n',
 '!\n',
 'no ip domain lookup\n',
 '!\n',
 'ip ssh version 2\n',
 '!\n']
```

ЧТЕНИЕ ФАЙЛОВ. МЕТОД READLINES()

Если нужно получить строки файла, но без перевода строки в конце, можно воспользоваться методом `split` и как разделитель, указать символ `\n`:

```
In [11]: f = open('r1.txt')

In [12]: f.read().split('\n')
Out[12]:
['!',
 'service timestamps debug datetime msec localtime show-timezone year',
 'service timestamps log datetime msec localtime show-timezone year',
 'service password-encryption',
 'service sequence-numbers',
 '!',
 'no ip domain lookup',
 '!',
 'ip ssh version 2',
 '!',
 '']
```

Обратите внимание, что последний элемент списка - пустая строка.

ЧТЕНИЕ ФАЙЛОВ. МЕТОД READLINES()

Если перед выполнением `split()`, воспользоваться методом `rstrip()`, список будет без пустой строки в конце:

```
In [13]: f = open('r1.txt')

In [14]: f.read().rstrip().split('\n')
Out[14]:
['!',
 'service timestamps debug datetime msec localtime show-timezone year',
 'service timestamps log datetime msec localtime show-timezone year',
 'service password-encryption',
 'service sequence-numbers',
 '!',
 'no ip domain lookup',
 '!',
 'ip ssh version 2',
 '!']
```

SEEK()

До сих пор, файл каждый раз приходилось открывать заново, чтобы снова его считать. Так происходит из-за того, что после методов чтения, курсор находится в конце файла. И повторное чтение возвращает пустую строку.

Чтобы ещё раз считать информацию из файла, нужно воспользоваться методом seek, который перемещает курсор в необходимое положение.

```
In [15]: f = open('r1.txt')

In [16]: print(f.read())
!
service timestamps debug datetime msec localtime show-timezone year
service timestamps log datetime msec localtime show-timezone year
service password-encryption
service sequence-numbers
!
no ip domain lookup
!
ip ssh version 2
!
```

SEEK()

Если вызывать ещё раз метод `read`, возвращается пустая строка:

```
In [17]: print(f.read())
```

Но, с помощью метода `seek`, можно перейти в начало файла (0 означает начало файла):

```
In [18]: f.seek(0)
```

После того, как, с помощью `seek`, курсор был переведен в начало файла, можно опять считывать содержимое:

```
In [19]: print(f.read())
!  
service timestamps debug datetime msec localtime show-timezone year  
service timestamps log datetime msec localtime show-timezone year  
service password-encryption  
service sequence-numbers  
!  
no ip domain lookup  
!  
ip ssh version 2  
!
```


ЗАПИСЬ ФАЙЛОВ

ЗАПИСЬ ФАЙЛОВ

При записи, очень важно определиться с режимом открытия файла, чтобы случайно его не удалить:

- `append` - добавить строки в существующий файл
- `write` - перезаписать файл
- оба режима создают файл, если он не существует

Для записи в файл используются такие методы:

- `write()` - записать в файл одну строку
- `writelines()` - позволяет передавать в качестве аргумента список строк

WRITE()

Метод write ожидает строку, для записи.

Для примера, возьмем список строк с конфигурацией:

```
In [1]: cfg_lines = ['!',  
...: 'service timestamps debug datetime msec localtime show-timezone year',  
...: 'service timestamps log datetime msec localtime show-timezone year',  
...: 'service password-encryption',  
...: 'service sequence-numbers',  
...: '!',  
...: 'no ip domain lookup',  
...: '!',  
...: 'ip ssh version 2',  
...: '!']
```

WRITE()

Открытие файла r2.txt в режиме для записи:

```
In [2]: f = open('r2.txt', 'w')
```

Преобразуем список команд в одну большую строку с помощью join:

```
In [3]: cfg_lines_as_string = '\n'.join(cfg_lines)
```

```
In [4]: cfg_lines_as_string
```

```
Out[4]: '!\\nservice timestamps debug datetime msec localtime show-timezone year\\nservice timestamps log date
```

4

WRITE()

Запись строки в файл:

```
In [5]: f.write(cfg_lines_as_string)
```

Аналогично можно добавить строку вручную:

```
In [6]: f.write('\nhostname r2')
```

После завершения работы с файлом, его необходимо закрыть:

```
In [7]: f.close()
```

WRITE()

Так как `ipython` поддерживает команду `cat`, можно легко посмотреть содержимое файла:

```
In [8]: cat r2.txt
!  
service timestamps debug datetime msec localtime show-timezone year  
service timestamps log datetime msec localtime show-timezone year  
service password-encryption  
service sequence-numbers  
!  
no ip domain lookup  
!  
ip ssh version 2  
!  
hostname r2
```

WRITELINES()

Метод `writelines()` ожидает список строк, как аргумент.

```
In [1]: cfg_lines = ['!',
...: 'service timestamps debug datetime msec localtime show-timezone year',
...: 'service timestamps log datetime msec localtime show-timezone year',
...: 'service password-encryption',
...: 'service sequence-numbers',
...: '!',
...: 'no ip domain lookup',
...: '!',
...: 'ip ssh version 2',
...: '!']

In [9]: f = open('r2.txt', 'w')

In [10]: f.writelines(cfg_lines)

In [11]: f.close()

In [12]: cat r2.txt
!service timestamps debug datetime msec localtime show-timezone yearservice timestamps log datetime msec loc
```

WRITELINES()

```
In [13]: cfg_lines2 = []
```

```
In [14]: for line in cfg_lines:
.....:     cfg_lines2.append( line + '\n' )
.....:
```

```
In [15]: cfg_lines2
```

```
Out[15]:
```

```
['!\n',
 'service timestamps debug datetime msec localtime show-timezone year\n',
 'service timestamps log datetime msec localtime show-timezone year\n',
 'service password-encryption\n',
 'service sequence-numbers\n',
 '!\n',
 'no ip domain lookup\n',
 '!\n',
 'ip ssh version 2\n',
```


WRITELINES()

```
In [18]: f = open('r2.txt', 'w')
```

```
In [19]: f.writelines(cfg_lines3)
```

```
In [20]: f.close()
```

```
In [21]: cat r2.txt
```

```
!  
service timestamps debug datetime msec localtime show-timezone year  
service timestamps log datetime msec localtime show-timezone year  
service password-encryption  
service sequence-numbers  
!  
no ip domain lookup  
!  
ip ssh version 2  
!
```

ЗАКРЫТИЕ ФАЙЛОВ

ЗАКРЫТИЕ ФАЙЛОВ

В реальной жизни, для закрытия файлов, чаще всего, используется конструкция `with`. Её намного удобней использовать, чем закрытия файла явно. Но, так как в жизни можно встретить и метод `close`, в этом разделе рассматривается его использование.

После завершения работы с файлом, его нужно закрыть. В некоторых случаях, Python может самостоятельно закрыть файл. Но лучше на это не рассчитывать и закрывать файл явно.

CLOSE()

```
In [1]: f = open('r1.txt', 'r')
```

Теперь можно считать содержимое:

```
In [2]: print(f.read())
!  
service timestamps debug datetime msec localtime show-timezone year  
service timestamps log datetime msec localtime show-timezone year  
service password-encryption  
service sequence-numbers  
!  
no ip domain lookup  
!  
ip ssh version 2  
!
```

CLOSE()

У объекта file есть специальный атрибут closed, который позволяет проверить закрыт файл или нет. Если файл открыт, он возвращает False:

```
In [3]: f.closed  
Out[3]: False
```

Теперь закрываем файл и снова проверяем closed:

```
In [4]: f.close()  
  
In [5]: f.closed  
Out[5]: True
```

CLOSE()

Если попробовать прочитать файл, возникнет исключение:

```
In [6]: print(f.read())
-----
ValueError                                Traceback (most recent call last)
<ipython-input-53-2c962247edc5> in <module>()
----> 1 print f.read()

ValueError: I/O operation on closed file
```

ИСПОЛЬЗОВАНИЕ TRY/FINALLY ДЛЯ РАБОТЫ С ФАЙЛАМИ

ИСПОЛЬЗОВАНИЕ TRY/FINALLY ДЛЯ РАБОТЫ С ФАЙЛАМИ

С помощью обработки исключений, можно:

- перехватывать исключения, которые возникают, при попытке прочитать несуществующий файл
- закрывать файл, после всех операций, в блоке finally

Если попытаться открыть для чтения файл, которого не существует, возникнет такое исключение:

```
In [7]: f = open('r3.txt', 'r')
-----
IOError                                Traceback (most recent call last)
<ipython-input-54-1a33581ca641> in <module>()
----> 1 f = open('r3.txt', 'r')

IOError: [Errno 2] No such file or directory: 'r3.txt'
```


ИСПОЛЬЗОВАНИЕ TRY/FINALLY ДЛЯ РАБОТЫ С ФАЙЛАМИ

С помощью конструкции try/except, можно перехватить это исключение и вывести своё сообщение:

```
In [8]: try:
.....:     f = open('r3.txt', 'r')
.....: except IOError:
.....:     print('No such file')
.....:
No such file
```

ИСПОЛЬЗОВАНИЕ TRY/FINALLY ДЛЯ РАБОТЫ С ФАЙЛАМИ

А с помощью части `finally`, можно закрыть файл, после всех операций:

```
In [9]: try:
.....:     f = open('r1.txt', 'r')
.....:     print(f.read())
.....: except IOError:
.....:     print('No such file')
.....: finally:
.....:     f.close()
.....:
!
service timestamps debug datetime msec localtime show-timezone year
service timestamps log datetime msec localtime show-timezone year
service password-encryption
service sequence-numbers
!
no ip domain lookup
!
ip ssh version 2
!

In [10]: f.closed
Out[10]: True
```

КОНСТРУКЦИЯ WITH

КОНСТРУКЦИЯ WITH

```
In [1]: with open('r1.txt', 'r') as f:  
.....:     for line in f:  
.....:         print(line)  
.....:  
!
```

```
service timestamps debug datetime msec localtime show-timezone year
```

```
service timestamps log datetime msec localtime show-timezone year
```

```
service password-encryption
```

```
service sequence-numbers
```

```
!
```

```
no ip domain lookup
```

```
!
```

```
ip ssh version 2
```

```
!
```

КОНСТРУКЦИЯ WITH

```
In [2]: with open('r1.txt', 'r') as f:
.....:     for line in f:
.....:         print(line.rstrip())
.....:
!
service timestamps debug datetime msec localtime show-timezone year
service timestamps log datetime msec localtime show-timezone year
service password-encryption
service sequence-numbers
!
no ip domain lookup
!
ip ssh version 2
!

In [3]: f.closed
Out[3]: True
```

КОНСТРУКЦИЯ WITH

С конструкцией with можно использовать не только такой построчный вариант считывания, все методы, которые рассматривались до этого, также работают:

```
In [4]: with open('r1.txt', 'r') as f:
.....:     print(f.read())
.....:
!
service timestamps debug datetime msec localtime show-timezone year
service timestamps log datetime msec localtime show-timezone year
service password-encryption
service sequence-numbers
!
no ip domain lookup
!
ip ssh version 2
!
```

Конструкция with может использоваться не только с файлами.