Язык программирования Python

Рекурсия. Работа с файлами в Python. Сложность основных операций.

Алексей Андреевич Сорокин

спецкурс, ОТИПЛ МГУ,
осенний семестр 2017–2018 учебного года
10 октября 2017 г.

Чтение и запись файлов

- Чтение и запись в Python происходят в два этапа: открытие файлового потока и собственно операции с файлами.

Чтение и запись файлов

- Чтение и запись в Python происходят в два этапа: открытие файлового потока и собственно операции с файлами.
- Открытие потока происходит командой **open**(⟨имя_файла⟩, ⟨режим⟩, ⟨кодировка⟩).

```
fin = open("infile.txt", "r", encoding="utf8")
```

- "infile.txt" — файл, находящийся в текущей директории (он должен существовать).
- "r" — режим чтения ("w" — запись).
- encoding — кодировка, лучше использовать utf-8 всегда, когда это только возможно.

Чтение и запись файлов

- Чтение и запись в Python происходят в два этапа: открытие файлового потока и собственно операции с файлами.
- Открытие потока происходит командой **open**(⟨имя_файла⟩, ⟨режим⟩, ⟨кодировка⟩).

```
fin = open("infile.txt", "r", encoding="utf8")
```

- "infile.txt" — файл, находящийся в текущей директории (он должен существовать).
- "r" — режим чтения ("w" — запись).
- encoding — кодировка, лучше использовать utf-8 всегда, когда это только возможно.
- После работы с файлом его обязательно нужно закрыть:

```
fin.close()
```

Чтение файла

- Функция **open** возвращает итератор по файлу.
- Можно перебирать его элементы стандартным образом:

```
fin = open("infile.txt", "r", encoding="utf8")
for line in fin:
    # делаем что-нибудь с каждой строкой
fin.close()
```

Чтение файла

- Функция **open** возвращает итератор по файлу.
- Можно перебирать его элементы стандартным образом:

```
fin = open("infile.txt", "r", encoding="utf8")
for line in fin:
    # делаем что-нибудь с каждой строкой
fin.close()
```

- Часто вначале нужно удалить пробелы на концах строк:

```
for line in fin:
    line = line.strip()
    # делаем что-нибудь с каждой строкой
```

Чтение файла

- Функция **open** возвращает итератор по файлу.
- Можно перебирать его элементы стандартным образом:

```
fin = open("infile.txt", "r", encoding="utf8")
for line in fin:
    # делаем что-нибудь с каждой строкой
fin.close()
```

- Часто вначале нужно удалить пробелы на концах строк:

```
for line in fin:
    line = line.strip()
    # делаем что-нибудь с каждой строкой
```

- Можно прочитать весь файл в одну строку:

```
text = fin.read()
```

Чтение файла

- Функция **open** возвращает итератор по файлу.
- Можно перебирать его элементы стандартным образом:

```
fin = open("infile.txt", "r", encoding="utf8")
for line in fin:
    # делаем что-нибудь с каждой строкой
fin.close()
```

- Часто вначале нужно удалить пробелы на концах строк:

```
for line in fin:
    line = line.strip()
    # делаем что-нибудь с каждой строкой
```

- Можно прочитать весь файл в одну строку:

```
text = fin.read()
```

- Или в список строк:

```
lines = fin.readlines()
```

- Но тогда пробелы придётся удалять вручную.

Запись в файл

- Открытие файла на запись аналогична чтению:

```
fout = open("outfile.txt", "w", encoding="utf8")
```

- Если файл не существовал, он будет создан.

Запись в файл

- Открытие файла на запись аналогична чтению:

```
fout = open("outfile.txt", "w", encoding="utf8")
```

- Если файл не существовал, он будет создан.
- Запись осуществляется командой **write**.

Запись в файл

- Открытие файла на запись аналогична чтению:

```
fout = open("outfile.txt", "w", encoding="utf8")
```

- Если файл не существовал, он будет создан.
- Запись осуществляется командой **write**.
- В файл пишутся только строки (не как в функции **print**):

```
k = 2
fout.write(str(k))  # правильно
fout.write(k)  # неправильно
```

Запись в файл

- Открытие файла на запись аналогична чтению:

```
fout = open("outfile.txt", "w", encoding="utf8")
```

- Если файл не существовал, он будет создан.
- Запись осуществляется командой **write**.
- В файл пишутся только строки (не как в функции **print**):

```
k = 2
fout.write(str(k))  # правильно
fout.write(k)  # неправильно
```

- В отличии от **print**, нужно явно задать перевод строки:

```
lines = ["first line", "second line", "last line"]
fout = open("lines.txt", "w", encoding="utf8")
for line in lines:
    fout.write(line + "\n")
fout.close()
```

Запись в файл

- Можно не закрывать файл после работы с ним.
- Для этого существует оператор **with**:

```
lines = ["first line", "second line", "last line"]
with open("lines.txt", "w", encoding="utf8") as fout:
    for line in lines:
        fout.write(line + "\n")
```

Запись в файл

- Можно не закрывать файл после работы с ним.
- Для этого существует оператор **with**:

```
lines = ["first line", "second line", "last line"]
with open("lines.txt", "w", encoding="utf8") as fout:
    for line in lines:
        fout.write(line + "\n")
```

- Аналогично для чтения файла:

```
with open("infile.txt", "r", encoding="utf8") as fin:
    for line in fin:
        line = line.strip()
        # делаем что-нибудь с каждой строкой
```

Сложность операций

- Разные типы данных имеют разную сложность операций (n — текущее число элементов):

		Список	Словарь	Множество
Вставка (в конец)	Операция	a.append(x)	–	–
	Сложность	$O(1)$	–	–
Вставка (по ключу)	Операция	a.insert(x, i)	a[i] = x	a.add(x)
	Сложность	$O(n)$	$O(1)$	$O(1)$
Поиск (по ключу)	Операция	x **in** a	x **in** a	x **in** a
	Сложность	$O(n)$	$O(1)$	$O(1)$
Удаление (из конца)	Операция	x.pop()	–	–
	Сложность	$O(n)$	–	–
Удаление (по ключу)	Операция	x.pop(i)	x.pop(i)	x.discard(i)
	Сложность	$O(n)$	$O(1)$	$O(1)$

- Сложность вставки для словаря и множества — *амортизированная*, то есть в среднем. В худшем случае она достигает $O(n)$.