

**КАЛУЖСКИЙ ФИЛИАЛ
ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО
ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ Н.Э. БАУМАНА
(национальный исследовательский университет)»**



Факультет «Информатика и управление»

Кафедра «Программное обеспечение ЭВМ, информационные технологии»

Высокоуровневое программирование

Лекция №7. «Работа с текстовыми файлами в формате csv, json»

Конструкция with

- Конструкция **with** называется *менеджер контекста*.
- Конструкция **with** гарантирует закрытие файла автоматически.

```
1 with open('file_name.txt', 'r') as f:  
2     for line in f:  
3         print(line)
```

From naked stones of agony

I will build a house for me;

As a mason all alone

I will raise it, stone by stone,

And every stone where I have bled

Will show a sign of dusky red.

with() и rstrip()

```
1  # rstrip() - rstrip возвращает строку, копию исходной,  
2  # удаляя из конца исходной строки все символы,  
3  # указанные в скобках (по умолчанию - пробел).  
4  with open('file_name.txt', 'r') as f:  
5      for line in f:  
6          print(line.rstrip())
```

From naked stones of agony
I will build a house for me;
As a mason all alone
I will raise it, stone by stone,
And every stone where I have bled
Will show a sign of dusky red.

Открытие двух файлов

- В блоке **with** можно открывать два файла таким образом:


```
1 with open('f_write.txt', 'r') as src, open('result.txt', 'w') as dest:
2     for line in src:
3         if line.startswith('L'):
4             dest.write(line)
```

```
1 Love is too young to know what conscience is
2 Yet who knows not conscience is born of love?
3 Then, gentle cheater, urge not my amiss,
4 Lest guilty of my faults thy sweet self prove.
```

```
1 Love is too young to know what conscience is
2 Lest guilty of my faults thy sweet self prove.
```

Пример

```
1 human = dict(name="Иван Иванов", age=35, weight=70.5)
2 filename = "13_10_20_data.txt"
3
4 fh = open(filename, "w")
5 fh.write(human["name"] + "\n")
6 fh.write(str(human["age"]) + "\n")
7
8 print(human["weight"], file=fh)
9
10 fh.close()
```

 jupyter 13_10_20_data.txt✓

File Edit View Language

```
1 fh = open(filename)
2 name = fh.readline().strip()
3 age = int(fh.readline())
4 weight = float(fh.readline())
5 print(name, age, weight)
6 fh.close()
```

```
1 Иван Иванов
2 35
3 70.5
4
```

Иван Иванов 35 70.5

Иные форматы файлов

Большое количество данных в совокупности с их разнородностью привело к появлению специальных форматов файлов, позволяющих хранить различные объемы связанной информации и не привязанных к конкретному языку программирования.

Среди них одними из наиболее популярных являются:

- **CSV** (*англ.* **C**omma-**S**eparated **V**alues - значения, разделенные запятыми);
- **JSON** (*англ.* **J**ava**S**cript **O**bject **N**otation) - текстовый формат обмена данными, основанный на JavaScript;
- **XML** (*англ.* e**X**tensible **M**arkup **L**anguage - расширяемый язык разметки);
- **YAML** (*англ.* **Y**AML **A**in't **M**arkup **L**anguage - «YAML - Не язык разметки»);
- **INI** (*англ.* **I**n**i**tialization file - файл инициализации) и др.

Подавляющее большинство форматов поддерживается Python (стандартными или сторонними модулями и пакетами).

CSV

- CSV (англ. Comma-Separated Values - значения, разделенные запятыми, 2005 г.) - текстовый формат, предназначенный для представления табличных данных. Каждая строка файла - это одна строка таблицы, где значения отдельных колонок разделяются разделительным символом (англ. delimiter) запятой , и заключаются в кавычки «
- В Python работа с CSV-файлами поддерживается стандартным модулем csv, предоставляющем следующие основные объекты и функции:

CSV – объекты и функции

```
csv.reader(csvfile, dialect='excel', **fmtparams)
```

Создает и возвращает объект для чтения последовательности из CSV-файла.

Некоторые из параметров:

Параметры

- **csvfile** – итерируемый объект, возвращающий строку на каждой итерации (например, файловый объект в текстовом режиме доступа);
- **dialect** – диалект CSV (набор специальных параметров);
- **fmtparams** – дополнительные настройки (совокупность кавычек, разделителей и т.д.).

CSV – объекты и функции

```
csv.writer(csvfile, dialect='excel', **fmtparams)
```

Создает и возвращает объект для записи последовательности в CSV-файл.

Некоторые из параметров:

Параметры

- `csvfile` – любой объект, поддерживающий метод записи `write()` ;
- `dialect` – аналогично `csv.reader()` ;
- `fmtparams` – аналогично `csv.reader()` .

CSV – объекты и функции

```
class csv.DictReader(csvfile, fieldnames=None, restkey=None, restval=None, dialect='excel', *args,  
**kwargs)
```

Создает и возвращает объект для чтения данных из CSV-файла как словаря значений.

Некоторые из параметров:

Параметры

- **csvfile** – итерируемый объект, возвращающий строку на каждой итерации (например, файловый объект в текстовом режиме доступа);
- **fieldnames** – список наименований столбцов (если не задан, используется первая строка файла).

CSV – объекты и функции

```
class csv.DictWriter(csvfile, fieldnames, restval='', extrasaction='raise', dialect='excel', *args, **kwargs)
```

Создает и возвращает объект для записи данных как словаря значений в CSV-файл.

Некоторые из параметров:

Параметры

- **csvfile** – любой объект, поддерживающий метод записи `write()` ;
- **fieldnames** – список наименований столбцов.

CSV

```
class csv.Writer
```

```
writerow(row)
```

Записывает последовательность `row` в CSV-файл.

```
writerows(rows)
```

Записывает список последовательностей `rows` в CSV-файл.

```
class csv.DictWriter
```

```
writeheader()
```

Записывает в файл заголовки файла, переданные при создании класса

```
writerow(row)
```


Записывает словарь `row` в CSV-файл.

```
writerows(rows)
```

Записывает список словарей `rows` в CSV-файл.

CSV – запись

```
1 import csv
2
3 filename = "13_10_20.csv"
4 studentlist = {"Петров": ["ИУК4-31Б", 20], "Иванов": ["ИУК4-32Б", 21], "Ермолаева": ["ИУК4-31Б", 20]}
5
6 with open(filename, "w", encoding="utf-8", newline="") as fh:
7     writer = csv.writer(fh, quoting=csv.QUOTE_ALL)
8     writer.writerow(["Фамилия", "Группа", "Возраст"])
9     for name, values in sorted(studentlist.items()):
10         writer.writerow([name, *values])
11     writer.writerow(["Илюхина", "ИУК4-32Б", "20"])
```

 jupyter 13_10_20.csv✓

File Edit View Language

```
1 "Фамилия", "Группа", "Возраст"
2 "Ермолаева", "ИУК4-31Б", "20"
3 "Иванов", "ИУК4-32Б", "21"
4 "Петров", "ИУК4-31Б", "20"
5 "Илюхина", "ИУК4-32Б", "20"
```


CSV – чтение из файла

```
1 rows = []
2 with open(filename, "r", encoding="utf-8") as fh:
3     reader = csv.reader(fh)
4     # reader - итерируемый объект и может быть преобразован в список строк
5     rows = list(reader)
6
7 for row in rows:
8     print(row)
```

```
['Фамилия', 'Группа', 'Возраст']
['Ермолаева', 'ИУК4-31Б', '20']
['Иванов', 'ИУК4-32Б', '21']
['Петров', 'ИУК4-31Б', '20']
['Илюхина', 'ИУК4-32Б', '20']
```

CSV – метод DictWriter

```
1 import csv
2
3 filename = "13_10_20_dict.csv"
4 studentlist = {"Петров": ["ИУК4-31Б", 20], "Иванов": ["ИУК4-32Б", 21], "Ермолаева": ["ИУК4-31Б", 20]}
5
6 with open(filename, "w", encoding="utf-8", newline="") as fh:
7     writer = csv.DictWriter(fh, fieldnames=["name", "group", "age"], quoting=csv.QUOTE_ALL)
8     writer.writerow(dict(name="Фамилия", group="Группа", age="Возраст"))
9     for name, values in sorted(studentlist.items()):
10         writer.writerow(dict(name=name, group=values[0], age=values[1]))
```

 jupyter 13_10_20_dict.csv✓

File	Edit	View	Language
------	------	------	----------

1	"Фамилия", "Группа", "Возраст"
2	"Ермолаева", "ИУК4-31Б", "20"
3	"Иванов", "ИУК4-32Б", "21"
4	"Петров", "ИУК4-31Б", "20"
-	

CSV – чтение DictReader

```
1 rows = []
2 with open(filename, "r", encoding="utf-8") as fh:
3     reader = csv.DictReader(fh)
4     # reader - итерируемый объект и может быть преобразован в список строк
5     rows = list(reader)
6
7 for row in rows:
8     print(row)
```

{'Фамилия': 'Ермолаева', 'Группа': 'ИУК4-31Б', 'Возраст': '20'}

{'Фамилия': 'Иванов', 'Группа': 'ИУК4-32Б', 'Возраст': '21'}

{'Фамилия': 'Петров', 'Группа': 'ИУК4-31Б', 'Возраст': '20'}

JSON

- JSON (англ. JavaScript Object Notation, 1999 г.) - текстовый формат обмена данными, основанный на JavaScript. Одно из преимуществ - JSON легко читается людьми (англ. human-readable)

```
1 {  
2     "ФИО": "Иванов Сергей Михайлович",  
3     "ЕГЭ": {  
4         "Математика": 90,  
5         "Физика": 70,  
6         "Информатика": 80  
7     },  
8     "Хобби": ["Рисование", "Плавание"],  
9     "Возраст": 25.5,  
10    "ДомЖивотные": null  
11 }
```

JSON

JSON-текст представляет собой одну из двух структур:

- набор пар ключ: значение (словарь в терминологии Python), где ключ - строка, значение - любой тип;
- упорядоченный набор значений (список в терминологии Python).

Значением может являться:

- строка (в кавычках);
- число;
- логическое значение (true/false);
- null;
- одна из структур.

JSON

- Одним из преимуществ JSON является близкое соответствие Python по типам данных. Работа с JSON-форматом поддерживается стандартным пакетом **json**, предоставляющем следующие основные функции:

JSON – функция

```
json.dumps(obj, skipkeys=False, ensure_ascii=True, check_circular=True, allow_nan=True, cls=None, indent=None, separators=None, default=None, sort_keys=False, **kw)
```

Сериализует объект `obj`, возвращая строку в JSON-формате.

Некоторые из параметров:

Параметры

- `obj` – сериализуемый объект;
- `ensure_ascii` – если равен `False`, запись не-ASCII значений происходит в файл «как есть», без преобразования в Unicode;
- `indent` – величина отступа для вложенных структур.

JSON – функции

```
json.loads(s, encoding=None, cls=None, object_hook=None, parse_float=None, parse_int=None, parse_constant=None, object_pairs_hook=None, **kw)
```

Десериализует объект (в том числе файловый) `s`, возвращая структуру в Python.

При ошибке десериализации возбуждается исключение `JSONDecodeError`.


```
exception json.JSONDecodeError(msg, doc, pos, end=None)
```

Класс исключения, возбуждаемый при ошибке в работе некоторых функций пакета.

JSON – запись в файл

```
1 import json
2
3 filename = "13_10_20.json"
4
5 info = {
6     "ФИО": "Иванов Сергей Михайлович",
7     "ЕГЭ": {
8         "Математика": 90,
9         "Физика": 70,
10        "Информатика": 80
11    },
12    "Хобби": ["Рисование", "Плавание"],
13    "Возраст": 25.5,
14    "ДомЖивотные": None
15 }
16
17 with open(filename, "w", encoding="utf-8") as fh:
18     fh.write(json.dumps(info, ensure_ascii=False, indent=4))
```

JSON – пример файла

 Jupyter 13_10_20.json ✓ несколько секунд назад

File Edit View Language

```
1 {
2   "ФИО": "Иванов Сергей Михайлович",
3   "ЕГЭ": {
4     "Математика": 90,
5     "Физика": 70,
6     "Информатика": 80
7   },
8   "Хобби": [
9     "Рисование",
10    "Плавание"
11  ],
12  "Возраст": 25.5,
13  "ДомЖивотные": null
14 }
```

JSON – чтение из файла

```
20 info_2 = []
21 with open(filename, encoding="utf-8") as fh:
22     info_2 = json.loads(fh.read())
23
24 print(info_2)
```

```
{'ФИО': 'Иванов Сергей Михайлович', 'ЕГЭ': {'Математика': 90, 'Физика': 70, '
e'], 'Возраст': 25.5, 'ДомЖивотные': None}
```


Задачи для самостоятельного решения

1. С клавиатуры в одной строке вводится произвольное количество вещественных чисел. Запишите их в файл, расположив каждое число на отдельной строке.
2. Дан файл, полученный на выходе задачи №1:
 - загрузите список чисел;
 - вычислите их сумму и максимум и допишите их в файл.

Выполнив программу несколько раз, убедитесь, что новые значения учитываются при подсчете.