Конспект: Работа с разными форматами данных

1. Формат CSV

Для работы с форматом CSV есть стандартный модуль csv (import scv)

- Чтение файлов

Пример:

```
# вариант 1 - csv.reader
with open("sample.csv", encoding="utf-8", newline="") as f:
    reader = csv.reader(f)
    for row in reader:
        print(row[-1])

# вариант 2 - csv.reader + list. только для небольших файлов!
with open("sample.csv", encoding="utf-8", newline="") as f:
    reader = csv.reader(f)
    news_list = list(reader)
header = news_list.pop(0)
for news in news_list:
    print(news[-1])

# вариант 3 - csv.DictReader
with open("sample.csv", encoding="utf-8", newline="") as f:
    reader = csv.DictReader(f)
    for row in reader:
        print(row["title"])
```

- Запись файлов

Пример:

```
# предварительно прочитаем данные
# данные будут храниться в news_list
with open("sample.csv", encoding="utf-8", newline="") as f:
    reader = csv.reader(f)
    news_list = list(reader)

# вариант 1 - csv.writer и "w" (перезапись файла)
```

```
with open("sample2.csv", "w", encoding="utf-8", newline="") as f:
    writer = csv.writer(f)
    writer.writerow(news_list[0]) # запись одной строки данных
    writer.writerows(news_list) # запись нескольких строк данных

# вариант 2 - csv.writer и "a" (добавление записей в существующий файл)
# добавим в файл sample2.csv еще несколько строк данных
with open("sample2.csv", "a", encoding="utf-8", newline="") as f:
    writer = csv.writer(f)
    writer.writerows(news_list[:3])
```

- Работа с DictWriter

Пример:

```
# предварительно прочитаем данные при помощи DictReader
# данные хранятся в reader
with open("sample.csv", encoding="utf-8", newline="") as f:
    reader = csv.DictReader(f)
    # запишем данные в новый файл sample_d.csv
    with open("sample_d.csv", "w", encoding="utf-8", newline="") as f:
        # необходимо передать ключи из DictReader: fieldnames=reader.fieldnames
        writer = csv.DictWriter(f, fieldnames=reader.fieldnames)
        writer.writeheader() # записываем заголовок файла (ключи DictReader)
        for row in reader:
            # изменим данные: пусть все заголовки новостей будут прописными
(большими) буквами
            row["title"] = row["title"].upper()
            writer.writerow(row) # записываем получившуюся строку данных в файл

# как получить ключи из DictReader:
with open("sample.csv", encoding="utf-8", newline="") as f:
    reader = csv.DictReader(f)
    # получаем ключи из reader
    keys = reader.fieldnames
    # преобразуем ключи из reader в обычный список:
    keys_list = list(keys)
    print(keys_list)
```

- Настройка диалектов

Пример:

```
csv.register_dialect('customcsv', delimiter=';', quoting=csv.QUOTE_MINIMAL,
quotechar="", escapechar='\\')
with open("sample4.csv", "w", encoding="utf-8", newline="") as f:
```

```
    writer = csv.writer(f, dialect="customcsv")
    writer.writerows(news_list)
```

2. Формат JSON

Для работы с форматом JSON есть стандартный модуль json (import json)

- Чтение файлов

Пример:

```
with open(r"d:\temp\files\newsafr.json", encoding="utf-8") as f:
    json_data = json.load(f)
```

- Запись файлов

Пример:

```
with open(r"d:\temp\files\result.json", "w", encoding="utf-8") as f:
    json.dump(json_data, f, ensure_ascii=False)
```

- Сохранение и чтение json в строку

Пример:

```
# сохранение json в строку
json_str = json.dumps(news, ensure_ascii=False)

# чтение json из строки
json_data_from_str = json.loads(json_str)
```

3. Формат YAML

Работа с YAML похожа на работу с json, только используется другая библиотека. Ее нужно предварительно установить с помощью утилиты pip (pip install pyyaml) и потом импортировать в программу с помощью import yaml.

- Чтение файлов

Пример:

```
with open("files/sample.yml", encoding = "utf-8") as f:
    data = yaml.load(f, Loader=yaml. FullLoader)
```

- Запись файлов

Пример:

```
data = {"channel": {"title": "Дайджест новостей о python", "link":
"https://pythondigest.ru/"}}
```

```
with open("files/sample.yml", "w") as f:
    yaml.dump(data, f, allow_unicode=True, default_flow_style=False)
```

4. Формат XML

Для работы с XML опять же используем станадртную библиотеку – xml.
Для упрощения работы импортируем модуль с алиасом (import xml.etree.ElementTree as ET)

- Чтение файлов

Пример:

```
# создаем парсер XML. парсеру обязательно задаем кодировку
parser = ET.XMLParser(encoding="utf-8")
# превращаем исходный текстовый файл sample.xml в дерево XML
tree = ET.parse("sample.xml", parser)

# посмотрим на наше дерево и увидим примерно такой объект:
# <xml.etree.ElementTree.ElementTree object at 0x7f3602e9ffd0>
print(tree)

# для работы с деревом XML нужно получить его корень - root
root = tree.getroot()
print(root.tag) # имя тега для root
print(root.text) # текст root (если ничего нет, выводится пустая строка)
print(root.attrib) # атрибуты root. если они есть, выводится dict
```

- Чтение из строки

Пример:

```
# строка, содержащая XML:
xml_str = '<root><channel type="dict"><title type="str">Дайджест новостей о
python</title><link type="str">https://pythondigest.ru/</link></channel></root>'

# превращаем строку в XML. внимание! в этом случае мы получаем не дерево,
а сразу корень!
root = ET.fromstring(xml_str)
# при необходимости строим дерево XML - просто преобразовываем root к типу
ElementTree
tree = ET.ElementTree(root)
```

- Работа с данными

Пример:

```
# прочитаем XML и получим root
parser = ET.XMLParser(encoding="utf-8")
tree = ET.parse("sample.xml", parser)
root = tree.getroot()

# ищем все теги item (внутри которых находятся новости)
news_list = root.findall("channel/item")
print(f"В этом файле {len(news_list)} новостей")

# читаем заголовки новостей - стандартный подход (прочитать весь блок
новости, найти и вывести title)
for news in news_list:
    title = news.find("title")
    print(title.text)

# читаем заголовки новостей - упрощенный подход (прочитать сразу все title)
# доступен только в XML. в CSV и JSON такого нет
titles_list = root.findall("channel/item/title")
for title in titles_list:
    print(title.text)
```

- Запись файла без форматирования

Пример:

```
# обязательно задать кодировку! иначе кириллица превратится в цифры!
tree.write("sample2.xml", encoding="utf-8")

# ВНИМАНИЕ! write() пишет текст без отступов (на примере из лекции этого не
видно, потому что исходный XML уже форматированный
# зато хорошо видно в этом коде:
xml_str = '<root><channel type="dict"><title type="str">Дайджест новостей о
python</title><link type="str">https://pythondigest.ru/</link></channel></root>'
root = ET.fromstring(xml_str)
tree = ET.ElementTree(root)
tree.write("sample3.xml", encoding="utf-8")
```

- Запись файла с форматированием

Пример:

```
# для XML с отступами используйте модуль xmlformatter. предварительно
нужно установить: pip install xmlformatter
# в параметрах Formatter'a указывается тип отступа (пробел, табулятор) и
количество отступов, в качестве входного значения даем XML,
сериализованный в строку
import xmlformatter
```

```
# предварительно создаем XML (например, из строки)
xml_str = '<root><channel type="dict"><title type="str">Дайджест новостей о
python</title><link type="str">https://pythondigest.ru/</link></channel></root>'
root = ET.fromstring(xml_str)

# создаем formatter и задаем параметры форматирования
formatter = xmlformatter.Formatter(indent="2", indent_char=" ")

# форматируем. не забудьте указать кодировку!
prettyxml = formatter.format_string(ET.tostring(root)).decode("utf-8")

# записываем получившийся результат как текст
with open("sample4.xml", "w", encoding="utf-8") as f:
    f.write(prettyxml)
```