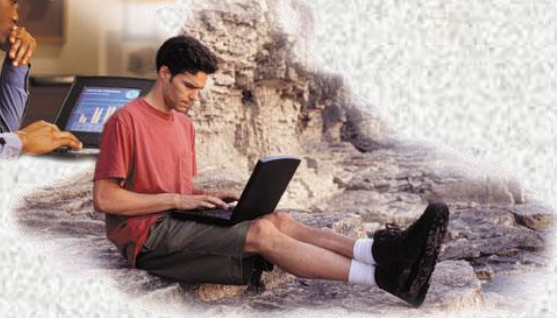




Основи на Програмирането



Лекция 12



Модули за четене и запис на файлове от
различни типове

Какво ще научите

- ✓ Работа с файлове - преговор
- ✓ Сериализация
- ✓ Модул pickle
- ✓ Модул json
- ✓ Модул csv
- ✓ Интерфейс с електронни таблици
- ✓ Интерфейс с електронни документи

Файлове

- ✓ Използват се за постоянно съхраняване на данни
- ✓ В най-простия си вид въвеждането и извеждането на информация във файл наподобява това със стандартния вход (клавиатура) и изход (екран)
- ✓ Входно-изходните операции с файлове са много повече на брой, с много по-големи възможности и позволяват всякаква обработка на информацията съхранявана във файловете
- ✓ Стандартната команда за свързване на обект от тип данни файл с конкретен физически файл:

```
afile = open(<filename>,< mode>)
```

Отваряне на файл

`<file_obj_name> = open(<filename>,< mode>)`

- ✓ `open()` е функция, която отваря физическия файл за вход/изход, създава входно-изходен буфер в паметта и връща като резултат адреса на този буфер
- ✓ `<file_obj_name>` е променлива, която задава името на обект от тип файл, който се свързва с току-що създадения входно-изходен буфер, т.е. тази променлива съдържа като стойност адреса на буфера на файла

Задаване на име за файлове

`<file_obj_name> = open(<filename>,< mode>)`

- ✓ И двата параметъра на функцията `open()` се задават като обекти от тип низ
 - ✓ Като конкретно име на файл (параметър `<filename>`) се задава валидно за конкретната ОС име на файл заедно с път за достъп (абсолютен или относителен)
- "names.txt" – указва име на файл в текущата папка
- "C:\Python\file.txt" – указва име на файл с абсолютен път за достъп към него

Режими за работа с файлове

Примери за стойности на параметъра <mode>:

'r' - въвеждане от файл (четене, read)

'w' - създаване и извеждане във файл (писане, write) – ако съществува съдържанието му се изтрива

'a' - добавяне в края на файл

't' - работа с текстов файл

'b' - работа с двоичен (не текстов) файл

'+' - използване на файла за писане и четене (обновяване)

'x' - създаване и извеждане във файл (писане, write) – ако съществува връща грешка

Начини за работа с файлове

- ✓ Когато файла е отворен като двоичен, четене и записване става чрез обект данни `bytes`
- ✓ Когато файла е отворен като текстов, четене и записване става чрез обект данни `str`, като първо се прави декодиране на съдържанието (четене) или кодиране (записване)
- 'b' - работа с двоичен (не текстов) файл
- '+' - използване на файла за писане и четене (обновяване)
- 'x' - създаване и извеждане във файл (писане, `write`) – ако съществува връща грешка

Други параметри при отваряне на файлове

`buffering` – размер на буфера за четене и писане

`encoding` – метод на кодиране (само за текстов файл)

`errors` – метод за обработка на грешките

`newline` – как да се интерпретира край на ред (само за текстов файл) – има различия в различни ОС

`closefd` – указва как и кога да се затваря физически файла

`opener` – за използване на алтернативна програма за отваряне на файла

Вградени методи за работа с файлове

`aString = input.read()` # чете цял файл в низ

`aString = input.read(N)` # чете следващите N символа
(или байта) в низ

`aString = input.readline()` # чете един ред в низ

`aList = input.readlines()` # чете цял файл в списък низове
по редове

`output.write(aString)` # записва низ от символи (или
байтове) във файл

`output.writelines(aList)` # записва списък от низове
(редове) в цял файл

Вградени методи за работа с файлове 2

<code>aString = input.read()</code>	<code># чете цял файл в низ</code>
<code>output.close()</code>	<code># Затваряне на файл (след # всички записи)</code>
<code>output.flush()</code>	<code># Запис от буфера във файла без # затваряне</code>
<code>anyFile.seek(N)</code>	<code># Позиционира файл до позиция # N за следващо действие</code>
<code>for line in open('data'):</code>	<code># използва line за итеративно # четене по редове от файл</code>

Примери

Преброяване на броя редове в един файл:


```
file1 = open("text.txt", "r")  
line_count = 0  
for line_count in file1:  
    line_count += 1  
print('Брой на редове във файла: ', line_count)
```

Прочитане на цял файл:

```
File2 = open("text2.txt")  
Batch = File2.read()  
print(len(Batch))
```

Пример: копиране от един файл в друг

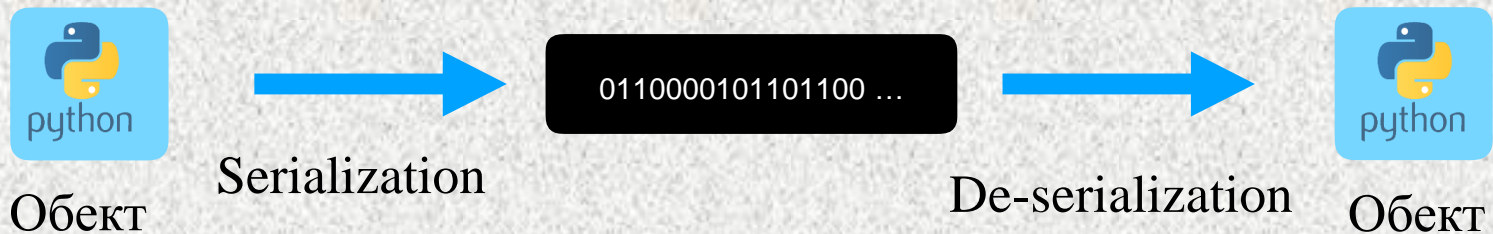
```
input_file = open("in.txt")  
output_file = open("out.txt", "w")  
for line in input_file:  
    output_file.write(line)
```



Сериализация (serialization)

Сериализацията е процес на преобразуване на структури от данни или обекти до поток от байтове, запазвайки техните полета и свойства. Потока може да бъде двоичен или текстов.

Чрез процеса на десериализацията от байтовете се създава идентично копие на оригиналния обект.



За обекти със сложна структура, този процес не е тривиален. Сериализацията на обекти в контекста на ООП не включва методите от класа.

Сериализация (serialization)

Приложения

- За запазване на състояние на сесия
- За отдалечено извикване на процеси
- За прехвърляне на данни по мрежата
- За съхранение на данни (в БД, на твърди дискове).
- За отдалечено обръщение към Уеб услуги
- За комуникация на обекти в компонентно базираното софтуерното инженерство

Сериализация (serialization)

Основни методи:

- XML е използван за създаване на четимо от човек, текст-базирано кодиране на информацията при сериализация. Недостатък е загубата на по-компактното, байт-поток-базирано кодиране, подходящо за пренос на големи обеми от данни.
- JSON е олекотена алтернатива на XML, която много често се използва за комуникация клиент-сървър в уеб приложения.
- Езиково-зависими методи за сериализация използващи байт-поток-базирано кодиране.

Методи за сериализация

```
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Пример за XML от сайта на W3schools

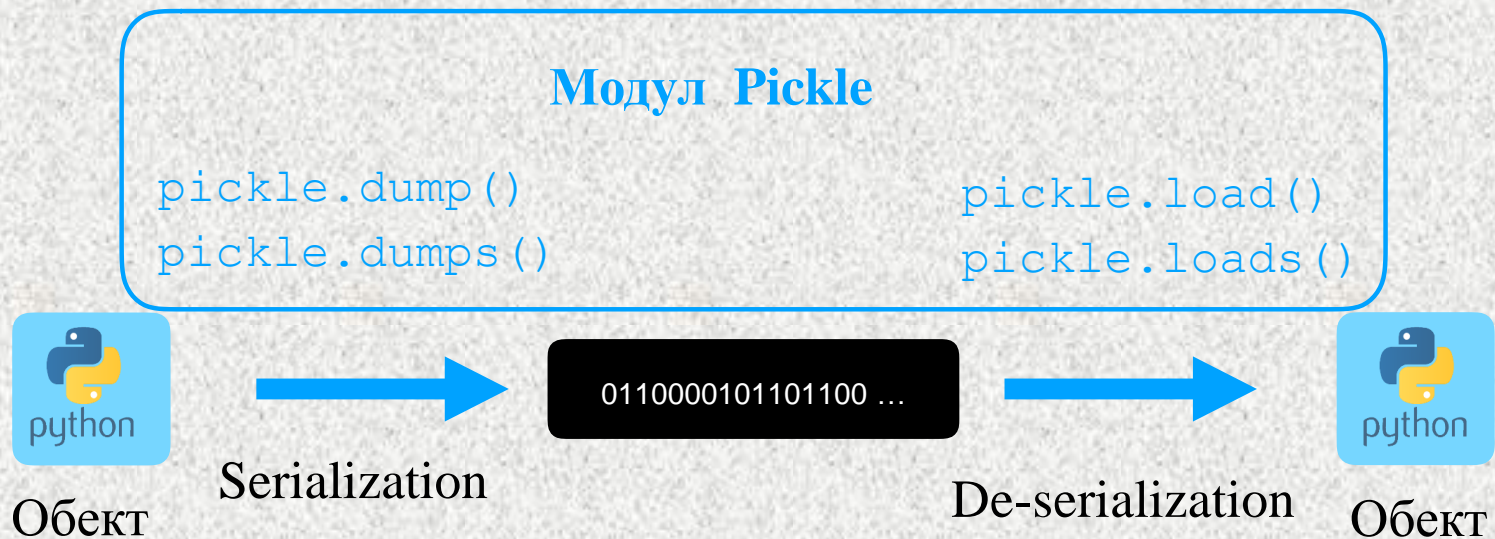
```
message Person {
  required string name = 1;
  required int32 id = 2;
  optional string email = 3;
}
```

Пример от сайт на Гугъл

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}
```

Пример за JSON от Wikipedia

Сериализация в Python - pickle



Сравнение на методите за сериализация

	Pickle	JSON	XML
Четим	НЕ	ДА	ДА
Поддържа Python	ДА	НЕ	НЕ
Поддържа класове	ДА	НЕ	НЕ

Модул pickle

При работа с текстови файлове, преобразуваме обектите в символни низове (от тип str) :

```
>>> f.write(str(12.3))  
>>> f.write(str([1,2,3]))
```

Проблемът възниква при четене на тези низове. Оригиналната информация е загубена. Не се знае от какво е бил получен низа, нито къде се разграничават отделните обекти:

```
>>> f.readline()  
'12.3[1, 2, 3]'
```

За решаване на този проблем се използва модула **pickle**, който запазва информацията за всеки обект. Използването на командите и функциите от този модул става след стандартното му зареждане:

```
>>> import pickle  
>>> f = open("test.pck","w")
```

Основни функции в pickle

За запазване на обект във файл използваме функцията **dump**:

```
>>> pickle.dump(12.3, f)
>>> pickle.dump([1,2,3], f)
>>> f.close()
```

За четене на обект се използва функцията **load**:

```
>>> f = open("test.pck","r")
>>> x = pickle.load(f)
>>> x
12.3
>>> type(x)
<type 'float'>
>>> y = pickle.load(f)
>>> y
[1, 2, 3]
>>> type(y)
<type 'list'>
```

При всяко обръщение към функцията **load**, получаваме един обект, заедно с информацията за неговия тип данни (клас).

Основни функции в pickle

`pickle.dump(obj,file,protocol=None,*,fix_imports=True,buffer_callback=None)`

Записва `obj` във файла `file`

`pickle.dumps(obj,protocol=None,*,fix_imports=True,buffer_callback=None)`

Връща двоичното представяне на `obj`

`pickle.load(file, *, fix_imports=True, encoding="ASCII", errors="strict", buffers=None)`

Чете един обект от файла `file` и го връща като резултат

`pickle.loads(data,*, fix_imports=True, encoding="ASCII", errors="strict", buffers=None)`

Чете един обект от двоичното представяне в двоичната структура `data` и след разпознаване го връща като резултат

Функция `dump(obj, file)`

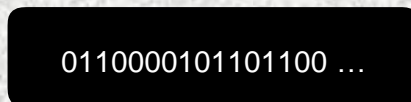
```
pickle.dump(obj, file, protocol=None, *,  
            fix_imports=True, buffer_callback=None)
```



Обект



Записва във



Файл

Функция `dumps(obj, file)`

```
pickle.dumps(obj, protocol=None, *,  
             fix_imports=True, buffer_callback=None)
```



Обект



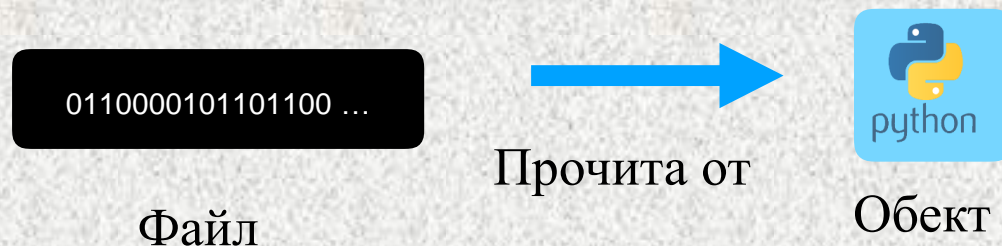
Връща

`b'\x80\x03K\n'`

Обект байт

Функция load(obj, file)

```
pickle.load(file, protocol=None, *,  
            fix_imports=True, buffer_callback=None)
```



Функция loads(obj, file)

```
pickle.loads(data, protocol=None, *,  
            fix_imports=True, buffer_callback=None)
```

b' \x80\x03K\n

Объект байт



Прочита от



Объект

Протоколите в pickle

Протокол	0	1	2	3	4	5
От версия	-	-	2.3	3.0	3.4	3.8
Описан	-	-	<u>PEP 307</u>	<u>Python 3.0</u>	<u>PEP 3154</u>	<u>PEP 574</u>
Нови черти	Четим от човек	двоичен	Произв олни класове	bytes обекти	Големи обеми от данни	Всякак ви данни
Заб.	текстов	двоичен		Не поддържа Python 2.x	Поддържа Unicode	

Кои обекти се запазват с pickle

- ✓ None, True, и False
- ✓ цели, реални и комплексни числа
- ✓ низове (str, bytes, bytearrays)
- ✓ контейнери (редици, списъци, речници, множества) съдържащи обекти които са pickleable
- ✓ класове и функции (без ламбда) дефинирани в глобалното ниво за програма или модул
- ✓ обекти - елементи на класове които са pickleable

Особености с функции и класове

- ✓ Само имената на функциите са запазват с `pickle`, заедно с името на модула където са дефинирани.
- ✓ Функциите се запазват като обръщение, не като стойност. Затова не може да се запазят функциите от тип ламбда.
- ✓ Подобно, класове и методи (които също са функции) се запазват с име (обръщение), а не със стойност (дефиниция).

Сериализация с JSON

- ✓ Форматът на данните в JSON наподобява почти на 100% структурата от данни речник в Python
- ✓ Основните разлики са в:
 - JSON използва само един контейнер – масив
 - Вместо True, False -> true, false
 - Вместо None -> null
 - Ключове могат да бъдат само символни низове
- ✓ `>>> import json # зарежда модула за работа с json`

Пример за JSON

```
{
  "business_id": "PK6aSizckHFWk8i0oxt5DA",
  "full_address": "400 Waterfront Dr E\nHomestead\nHomestead, PA 15120",
  "hours": {},
  "open": true,
  "categories": [
    "Burgers",
    "Fast Food",
    "Restaurants"
  ],
  "city": "Homestead",
  "review_count": 5,
  "name": "McDonald's",
  "neighborhoods": [
    "Homestead"
  ],
  "longitude": -79.910032,
  "state": "PA",
  "stars": 2,
  "latitude": 40.412086,
  "attributes": {
    "Take-out": true,
    "Wi-Fi": "free",
    "Drive-Thru": true,
    "Good For": {
      "dessert": false,
      "latenight": false,
      "lunch": false,
      "dinner": false,
      "breakfast": false,
      "brunch": false
    },
    "Caters": false,
    "Noise Level": "average",
    "Takes Reservations": false,
    "Delivery": false
  }
}
```

Основни функции в модула json

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

Записва obj в текстовия файл file

```
json.dumps(obj, skipkeys=False, ...)
```

Връща текстовото представяне на obj в json формат

```
json.load(file, cls=None, object_hook=None, parse_float=None,  
parse_int=None, parse_constant=None, object_pairs_hook=None, **kw)
```

Чете един обект от файла file и го връща като резултат

```
json.loads(str_data, cls=None, ...)
```

Чете един обект от текстовото представяне в низа data и след
разпознаване го връща като резултат

Преобразуване от Python към json

```
>>> print(json.dumps((3, 9, 'fff', [5, 9, 'asas'])))
```

```
[3, 9, "fff", [5, 9, "asas"]]
```

```
>>> print(json.dumps('String of "" strings?!?"'))
```

```
"String of \"\" strings?!?\"\""
```

```
>>> print(json.dumps({1:None, 2:False}))
```

```
{"1": null, "2": false}
```

```
>>>
```


Преобразуване в Python от json

```
>>> d = json.loads('[3, 9, "fff", [5, 9, "asas"]])
```

```
>>> d
```

```
[3, 9, 'fff', [5, 9, 'asas']]
```

```
>>> data
```

```
"""String of \" Strings?? \""""
```

```
>>> d2 = json.loads(data)
```

```
>>> d2
```

```
'String of " Strings?? '
```

```
>>> json.loads('{"1": null, "2": false}')
```

```
{'1': None, '2': False}
```

Сериализация с файлове - .csv

- ✓ Форматът на данните в .csv файл наподобява този на една електронна таблица
- ✓ Обикновено първият ред съдържа имена на колони
- ✓ Всички следващи редове съдържат данните от таблицата по колони, разделени със запетайки

Лекция 12

Пример за .CSV (без заглавен ред)

1. Giuliano Montaldo;83000000;Machine Gun
McCain;English;Italy;1969;6.2
2. James Cameron;760505847;Avatar;English;USA;2009;7.9
3. James Cameron;658672302;Titanic;English;USA;1997;7.7
4. Colin Trevorrow;652177271;Jurassic World;English;USA;2015;7
5. Joss Whedon;623279547;The Avengers;English;USA;2012;8.1
6. Christopher Nolan;533316061;The Dark
Knight;English;USA;2008;9
7. Andrew Adamson;436471036;Shrek 2;English;USA;2004;7.2
8. Steven Spielberg;434949459;E.T.;English;USA;1982;7.9
9. Francis Lawrence;424645577;The Hunger Games: Catching
Fire;English;USA;2013;7.6

Работа с .csv файлове в Python

Четене от файл в речник:

```
import csv  
fn= 'име на файла.csv '  
with open(fn, 'r', newline='') as myCsvFile:  
    reader = csv.DictReader(myCsvFile)  
    for row in reader:  
        for key in row:  
            print(key, ':', row[key])
```


Четене от .csv файл в речник

```
import csv
reader=csv.DictReader(open('test2.csv',encoding='utf-8'))
for row in data:
    print(row)
```

```
{'Език за програмиране': 'Python', 'Автор': ' Guido van Rossum', '
Година': ' 1991', ' Разширение': ' .py'}
```

```
{'Език за програмиране': 'Java', ' Автор': ' James Gosling', '
Година': ' 1995', ' Разширение': ' .java'}
```

```
{'Език за програмиране': 'C++', ' Автор': ' Bjarne Stroustrup', '
Година': ' 1983', ' Разширение': ' .cpp'}
```

Работа с .csv файлове в Python

Четене от файл

```
import csv  
fn= 'име на файла.csv '  
with open(fn, 'r', newline='') as myCsvFile:  
    data = csv.reader(myCsvFile)  
    for row in data:  
        print(row)
```

```
['Език за програмиране; Автор; Година; Разширение']  
['Python; Guido van Rossum; 1991; .py']  
['Java; James Gosling; 1995; .java']  
['C++; Bjarne Stroustrup; 1983; .cpp']
```

Четене от .csv файл с PANDAS

Четене от файл

```
import pandas as pd  
fn= 'име на файла.csv '  
df_csv=pd.read_csv(fn)
```

	Език за програмиране	Автор	Година	Разширение
0	Python	Guido van Rossum	1991	.py
1	Java	James Gosling	1995	.java
2	C++	Bjarne Stroustrup	1983	.cpp

Запис във .csv файл

```
import csv  
with open('test3.csv', mode='w') as file:  
    writer = csv.writer(file, delimiter=',', quotechar='"',  
                        quoting=csv.QUOTE_MINIMAL)  
    writer.writerow(['Език', 'Автор', 'Година', 'Разширение'])  
    writer.writerow(['Python', 'Guido van Rossum', '1991', '.py'])  
    writer.writerow(['Java', 'James Gosling', '1995', '.java'])  
    writer.writerow(['C++', 'Bjarne Stroustrup', '1985', '.cpp'])
```


Запис във .csv файл от речник

Пример за запис във файл

```
import csv  
fn = 'име на файла.csv '  
with open(fn, 'r', newline='') as myCsvFile:  
    columns = ['column_name_1', 'column_name_2']  
    writer = csv.DictWriter(myCsvFile, fieldnames=columns)  
    writer.writeheader()  
    writer.writerow({'col_name_1': 'Mark', 'col_name_2': 'Twain'})  
    writer.writerow({'column_name_1': 'Foo', 'column_name_2': 'Bar'})
```

Запис във .csv файл от PANDAS

Запис във файл

```
import pandas as pd  
fn= 'име на файла.csv '  
df.to_csv(fn,index=False)
```

Работа с електронни таблица

Основни понятия:

- ✓ *Workbook* - файл с една или няколко таблици
- ✓ *Sheet* - една таблица от файла, всяка си има име
- ✓ *Column* - колона в таблица, имат имена: A, B, ..., Z, AA, AB, ...
- ✓ *Row* - ред в таблица, номерират се от 1 нагоре
- ✓ *Cell* - клетка в таблицата, именува се с името на колоната и реда: A1, YB23, ...

Работа с електронни таблица

```
$ pip install openpyxl
```

```
from openpyxl import Workbook
```

```
wb = Workbook()
```

```
sheet = workbook.active
```

```
sheet["A1"] = "hello"
```

```
sheet["B1"] = "world!"
```

```
workbook.save(filename="hello_world.xlsx")
```


Четене от електронна таблица

[illegible]

Работа с Ексел файлове в PANDAS

Четене от файл

```
import pandas as pd
```

```
fn1= 'име на файла.xlsx '
```

```
sn= 'име на листа '
```

```
df = pd.read_excel(fn1, sn) # ако sn не е указан – Sheet1
```

```
with pd.ExcelFile(fn1) as xls: # няколко листа на един път
```

```
df1 = pd.read_excel(xls, 'Sheet1')
```

```
df2 = pd.read_excel(xls, 'Sheet2')
```

Лекция 12

Пример за Excel

Име на режисьор	Печалба	Заглавие на филм	Език	Страна	Година	Оценка
Giuliano Montaldo	83000000	Machine Gun McCain	English	Italy	1969	6.2
James Cameron	760505847	Avatar	English	USA	2009	7.9
James Cameron	658672302	Titanic	English	USA	1997	7.7
Colin Trevorrow	652177271	Jurassic World	English	USA	2015	7
Joss Whedon	623279547	The Avengers	English	USA	2012	8.1
Christopher Nolan	533316061	The Dark Knight	English	USA	2008	9
George Lucas	474544677	Star Wars: Episode I - The Phantom Menace	English	USA	1999	6.5
George Lucas	460935665	Star Wars: Episode IV - A New Hope	English	USA	1977	8.7
Joss Whedon	458991599	Avengers: Age of Ultron	English	USA	2015	7.5
Christopher Nolan	448130642	The Dark Knight Rises	English	USA	2012	8.5
Andrew Adamson	436471036	Shrek 2	English	USA	2004	7.2
Steven Spielberg	434949459	E.T. the Extra-Terrestrial	English	USA	1982	7.9
Francis Lawrence	424645577	The Hunger Games: Catching Fire	English	USA	2013	7.6

Запис в .xls файл от PANDAS

Запис във файл

```
import pandas as pd
```

```
fn1= 'име на файла.xlsx '
```

```
sn= 'име на листа '
```

```
df.to_excel(fn, sheet_name=sn, index=False)
```


Обработка на .pdf файлове

```
pip install PyPDF2
```

```
# за обработка на текст
```

```
import PyPDF2
```

```
pip install tabula-py
```

```
# за обработка на таблици
```

```
import tabula
```

Четене от .pdf файл

```
pdfFileObj = open('example.pdf', 'rb')  
pdfReader = PyPDF2.PdfFileReader(pdfFileObj)  
print(pdfReader.numPages)  
pageObj = pdfReader.getPage(0)  
print(pageObj.extractText())  
pdfFileObj.close()
```

Сливане на .pdf файлове

```
from PyPDF2 import PdfFilemerger
merger = PdfFilemerger()
pdf_files = [<f1>, <f2>, ...]
for files in pdf_files:
    with open(files, 'rb') as ff:
        merger.append(ff)
with open ('new_merged_file.pdf', 'wb') as ff2:
    merger.write(ff2)
merger.close()
```

Четене на таблици от .pdf файл

```
import tabula, pandas
# Таблиците се четат в DataFrame на pandas
df = tabula.read_pdf("offense.pdf")
df.head() # показва първите 5 реда
# За четене на няколко таблици:
df = tabula.read_pdf("offense.pdf", multiple_tables=True)
# За указване от къде точно да се чете таблицата:
tabula.read_pdf("offense.pdf", area=(126,149,212,462), pages=1)
# За указване преобразуване на таблицата в json:
tabula.read_pdf("offense.pdf", output_format="json")
```


Създаване на .pdf файл

```
pip install reportlab  
from reportlab.pdfgen import canvas  
c = canvas.Canvas("hello.pdf")  
c.drawString(100,750,"Welcome to Reportlab!")  
c.save()  
c.line, c.rect, c.circle, c.drawImage(), c.fill ...  
c.drawRightString(), c.drawCenteredString(), ...  
c.drawText(textobject)  
t = Table(data, ...)
```

Създаване на MSWord документ

```
from docx import Document
from docx.shared import Inches

document = Document()
document.add_heading('Document Title', 0)
p = document.add_paragraph('A plain paragraph having some ')
p.add_run('bold').bold = True
p.add_run(' and some ')
p.add_run('italic.').italic = True
document.add_heading('Heading, level 1', level=1)
document.add_paragraph('Intense quote', style='Intense Quote')
document.add_paragraph(
    'first item in unordered list', style='List Bullet')
document.add_paragraph(
    'first item in ordered list', style='List Number')
document.add_picture('monty-truth.png', width=Inches(1.25))
```

Създаване на MSWord документ

```
records = ((3, '101', 'Spam'),  
           (7, '422', 'Eggs'),  
           (4, '631', 'Spam, spam, eggs, and spam'))  
table = document.add_table(rows=1, cols=3)  
hdr_cells = table.rows[0].cells  
hdr_cells[0].text = 'Qty'  
hdr_cells[1].text = 'Id'  
hdr_cells[2].text = 'Desc'  
for qty, id, desc in records:  
    row_cells = table.add_row().cells  
    row_cells[0].text = str(qty)  
    row_cells[1].text = id  
    row_cells[2].text = desc  
document.add_page_break()  
document.save('demo.docx')
```

Четене на текст от MSWord документ

```
from docx import Document
fn = '<име на word файл>'
doc = Document(fn)
result = []
for line in doc.paragraphs:
    result.append(line.text)
'\n'.join(result)
print(result)
```


Заклучение

- ✓ Работа с файлове - преговор
- ✓ Сериализация
- ✓ Модул pickle
- ✓ Модул json
- ✓ Модул csv
- ✓ Интерфейс с електронни таблици
- ✓ Интерфейс с електронни документи