09_Soubory_CSV_JSON

C2184 Úvod do programování v Pythonu

9. Práce se soubory CSV a JSON

Formát CSV

- CSV = *comma-separated values*
- Slouží pro ukládání tabulkových dat
- Hodnoty jsou do sloupečků rozdělené pomocí separátoru (*delimiter,* většinou čárka) a do řádků pomocí znaku nového řádku
- https://cs.wikipedia.org/wiki/CSV
- Tabulka:

Rok výroby	Značka	Model	Cena
1995	Opel	Vectra	45 000
1998	Škoda	Felicia	80 000
2002	Škoda	Octavia	70 000

- CSV:

```
Rok výroby,Značka,Model,Cena
1995,Opel,Vectra,45000
1998,Škoda,Felicia,80000
2002,Škoda,Octavia,70000
```

Modul `csv` v Pythonu

- `csv.reader` – načítání formátu CSV
- `csv.writer` – ukládání ve formátu CSV
- https://docs.python.org/3/library/csv.html

Čtení

[1]:
```
with open('tabulka_auta.csv', 'r', encoding='utf8') as f:
    print(f.read())
```

```
Rok výroby,Značka,Model,Cena
1995,Opel,Vectra,45000
1998,Škoda,Felicia,80000
2002,Škoda,Octavia,70000
```

[2]:
```
import csv

with open('tabulka_auta.csv', 'r', encoding='utf8') as f:
    reader = csv.reader(f)
    for line in reader:
        print(line)
```

```
['Rok výroby', 'Značka', 'Model', 'Cena']
['1995', 'Opel', 'Vectra', '45000']
['1998', 'Škoda', 'Felicia', '80000']
['2002', 'Škoda', 'Octavia', '70000']
```

[3]:
```
with open('tabulka_auta.csv', 'r', encoding='utf8') as f:
    reader = csv.reader(f)
    table = list(reader)
table
```

[3]:
```
[['Rok výroby', 'Značka', 'Model', 'Cena'],
 ['1995', 'Opel', 'Vectra', '45000'],
 ['1998', 'Škoda', 'Felicia', '80000'],
 ['2002', 'Škoda', 'Octavia', '70000']]
```

- Načtené hodnoty jsou vždy řetězce, musíme si je sami převést na číslo
- `DictReader` – z řádků dělá slovníky

[4]:
```
with open('tabulka_auta.csv', encoding='utf8') as f:
    csvreader = csv.DictReader(f, ['year', 'brand', 'model', 'price'])
    for line in csvreader:
        print(dict(line))
```

```
{'year': 'Rok výroby', 'brand': 'Značka', 'model': 'Model', 'price':␣
 ↪'Cena'}
{'year': '1995', 'brand': 'Opel', 'model': 'Vectra', 'price': '45000'}
{'year': '1998', 'brand': 'Škoda', 'model': 'Felicia', 'price':␣
 ↪'80000'}
```

```
{'year': '2002', 'brand': 'Škoda', 'model': 'Octavia', 'price': '70000'}
```

- `DictReader` bez zadaných názvů sloupců – načte první řádek jako hlavičku

[5]:
```
with open('tabulka_auta.csv', encoding='utf8') as f:
    csvreader = csv.DictReader(f)
    for line in csvreader:
        print(dict(line))
```

```
{'Rok výroby': '1995', 'Značka': 'Opel', 'Model': 'Vectra', 'Cena': '45000'}
{'Rok výroby': '1998', 'Značka': 'Škoda', 'Model': 'Felicia', 'Cena': '80000'}
{'Rok výroby': '2002', 'Značka': 'Škoda', 'Model': 'Octavia', 'Cena': '70000'}
```

Zápis

[6]:
```
distances = [['',      'Brno', 'Praha', 'Ostrava'],
             ['Brno',      0,     202,       165],
             ['Praha',   202,       0,       362],
             ['Ostrava', 165,     362,         0]]
```

[7]:
```
with open('distances.csv', 'w', encoding='utf8') as f:
    csvwriter = csv.writer(f)
    csvwriter.writerows(distances)
```

[8]:
```
with open('distances.csv', 'r', encoding='utf8') as f:
    print(f.read())
```

```
,Brno,Praha,Ostrava
Brno,0,202,165
Praha,202,0,362
Ostrava,165,362,0
```

Zápis speciálních znaků

- Tabulka:

Rok výroby	Značka	Model	Poznámka	Cena
1995	Opel	Vectra	klimatizace, střešní okno	45 000
1998	Škoda	Felicia "Fun"		80 000

2002	Škoda	Octavia	klimatizace, ABS bouraná	70 000

- CSV:

```
1995,Opel,Vectra,"klimatizace, střešní okno",45000
1998,Škoda,"Felicia ""Fun""",,80000
2002,Škoda,Octavia,"klimatizace, ABS
bouraná",70000
```

- A proto je modul `csv` tak užitečný.

Parametry pro upřesnění formátu

- `delimiter` – oddělovač sloupců (default `','`)
- `quotechar` – vyčlenění polí se speciálními znaky (default `'"'`)
- `quoting` – strategie použití quotecharu (povolené hodnoty `csv.QUOTE_ALL`, `csv.QUOTE_MINIMAL`, `csv.QUOTE_NONNUMERIC`, `csv.QUOTE_NONE`)
  - Reader s `quoting=csv.QUOTE_NONNUMERIC` konvertuje na float vše, co není v uvozovkách.
- `doublequote` – zdvojení quotecharu ruší jeho funkci (default `True`)
- `escapechar` – ruší funkci speciálních znaků (delimiteru a quotecharu) (default `None`)
- `skipinitialspace` – ignoruje mezery těsně za oddělovačem (default `False`)
- `dialect` – nastavení více parametrů současně (např. `'excel'`)

[9]:
```
with open('distances.csv', 'w', encoding='utf8') as f:
    csvwriter = csv.writer(f, delimiter=';', quoting=csv.
↪QUOTE_NONNUMERIC)
    csvwriter.writerows(distances)
```

[10]:
```
with open('distances.csv', encoding='utf8') as f:
    print(f.read())
```

```
"";"Brno";"Praha";"Ostrava"
"Brno";0;202;165
"Praha";202;0;362
"Ostrava";165;362;0
```

Otázky:

Jak musíme nastavit `csv.reader`, aby správně načetl tabulku?

```
Wednesday:16 December:'18:00':'Ovečka Shaun ve filmu: Farmageddon':60 Kč
Wednesday:16 December:'20:30':Story of Tantra                     :100 Kč
Thursday :17 December:'18:00':Hungry Bear Tales                   :60 Kč
Thursday :17 December:'21:30':Between the Seasons                 :100 Kč
Friday   :18 December:'18:00':Disco in the Cinema                 :60 Kč
Saturday :19 December:'20:30':Summer of 85                        :100 Kč
Sunday   :20 December:'20:30':Klimt & Schiele - Eros and Psyche   :100 Kč
```

- A) `csv.reader(f)`
- B) `csv.reader(f, delimiter=':', escapechar="'")`
- C) `csv.reader(f, delimiter=':', quotechar="'")`
- D) `csv.reader(f, delimiter=':', quotechar='"', skipinitialspace=True)`

Formát JSON

- *JavaScript Object Notation*
- http://json.org/
- Mapování na typy Pythonu:

Python		JSON		Poznámka
int/float:	`5`, `10.2`	number:	`5`, `10.2`	
string:	`'ahoj'`	string:	`"ahoj"`	vždy dvojité uvozovky
bool:	`True`, `False`	boolean:	`true`, `false`	
None:	`None`	null:	`null`	
list, tuple:	`[]`, `()`	array:	`[]`	načte se vždy jako seznam
dict:	`{}`	object:	`{}`	klíče musí být řetězce

Modul `json`

- `json.load()` – načti JSON ze souboru
- `json.loads()` – načti JSON z řetězce
- `json.dump()` – zapiš JSON do souboru
- `json.dumps()` – zapiš JSON do řetězce
- https://docs.python.org/3/library/json.html

Čtení

[11]:
```python
with open('bob.json', encoding='utf8') as f:
    bob = f.read()
print(type(bob))
print(bob)
```

```
<class 'str'>
{
  "name": "Bob",
  "age": 30,
  "married": false,
  "cars": ["Ford", "BMW", "Fiat"]
}
```

[12]:
```
import json

with open('bob.json', encoding='utf8') as f:
    bob = json.load(f)
print(type(bob))
print(bob)
```

```
<class 'dict'>
{'name': 'Bob', 'age': 30, 'married': False, 'cars': ['Ford', 'BMW',␣
 ↪'Fiat']}
```

[13]:
```
text = '{ "name": "John", "age": 35, "married": true, "cars":␣
 ↪["Mercedes", "BMW", "Volkswagen"] }'
```

[14]:
```
john = json.loads(text)
print(type(john))
print(john)
```

```
<class 'dict'>
{'name': 'John', 'age': 35, 'married': True, 'cars': ['Mercedes',␣
 ↪'BMW',
'Volkswagen']}
```

Zápis

[15]:
```
alice = {'name': 'Alice', 'age': 28, 'married': False, 'cars':␣
 ↪('Ford', 'Trabant'), 10: 20 }
```

[16]:
```
with open('alice.json', 'w', encoding='utf8') as f:
    json.dump(alice, f)
```

[17]:
```
with open('alice.json', encoding='utf8') as f:
    print(f.read())
```

```
{"name": "Alice", "age": 28, "married": false, "cars": ["Ford",␣
 ↪"Trabant"],
"10": 20}
```

[18]:
```
text = json.dumps(alice)
print(type(text))
print(text)
```

```
<class 'str'>
{"name": "Alice", "age": 28, "married": false, "cars": ["Ford",␣
  ↪"Trabant"],
"10": 20}
```

[19]:
```
text = json.dumps(alice, indent=4)
print(type(text))
print(text)
```

```
<class 'str'>
{
    "name": "Alice",
    "age": 28,
    "married": false,
    "cars": [
        "Ford",
        "Trabant"
    ],
    "10": 20
}
```

Otázky:

Které z uvedených je validní JSON?

- A) `{ID: 12345, title: "Harry Potter and Learning Python", translations: ["en", "de", "cs", "sk", "hu", "pl"]}`
- B) `{"ID": 12345, "title": "Harry Potter and Learning Python", "translations": ["en", "de", "cs", "sk", "hu", "pl"]}`
- C) `{'ID': '12345', 'title': 'Harry Potter and Learning Python', 'translations': ['en', 'de', 'cs', 'sk', 'hu', 'pl']}`
- D) `{"ID": "12345", "title": "Harry Potter and Learning Python", "translations": {"en", "de", "cs", "sk", "hu", "pl"}}`

Modul `argparse`

- Předávání argumentů z příkazové řádky
- Stejný účel jako `sys.argv`, ale sofistikovanější a hezčí pro uživatele
- Argumenty z příkazové řádky (netýká se pouze Pythonu):
  - Poziční

  - Volby/přepínače/*options*
    * Začínají `-` (jednopísmenné) nebo `--` (vícepísmenné)
    * Mohou mít vlastní parametry
- Soubor `make_statistics.py`:

[ ]:
```
import argparse

parser = argparse.ArgumentParser(description='Example of argsparse.')
parser.add_argument('input', help='Input CSV file', type=str)
parser.add_argument('-H', '--header',
    help='Interpret the first line as column names',
    action='store_true')
parser.add_argument('-v', '--verbose',
    help='Print extra information',
    action='store_true')
parser.add_argument('-d', '--delimiter',
    help='Delimiter in the CSV file',
    type=str, default=',')
parser.add_argument('-s', '--stat',
    help='Statistics to be computed',
    choices=['mean', 'median', 'min', 'max'], default='mean')
args = parser.parse_args()

print('Input file:', args.input)
print('Header:', args.header)
print('Verbose:', args.verbose)
print('Delimiter:', args.delimiter)
print('Statistics:', args.stat)
```

- Spouštíme z príkazové řádky:

```
python make_statistics.py

python make_statistics.py --help

python make_statistics.py data.csv --stat median --header --verbose

python make_statistics.py data.csv -s median -H -v

python make_statistics.py data.csv -Hv --stat median
```

Další moduly – rozšiřující učivo

Formát XML

- *Extensible Markup Language*

- https://cs.wikipedia.org/wiki/Extensible_Markup_Language

```
<messages>
  <note id="501">
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
  </note>
  <note id="502">
    <to>Jani</to>
    <from>Tove</from>
    <heading>Re: Reminder</heading>
    <body>I will not</body>
  </note>
</messages>
```

Modul `lxml`

- čtení a zápis ve formátu XML (https://lxml.de)
- Externí balíček, nutno doinstalovat pomocí pipu

Modul `pickle`

- Uložení pythonovských dat v binárním formátu
- Dokáže uložit téměř libovolný objekt (např. i funkce)
- Nebezpečí – pickle soubor z cizího zdroje může obsahovat škodlivý kód!

Modul `requests`

- Internetová komunikace přes protokol HTTP
- Nutno doinstalovat pomocí pipu
- Posíláme požadavek (*request*) na server pomocí metod GET, POST, PUT, DELETE...
- Server nám vrací odpověď (*response*)

[20]:
```
import requests

URL = 'http://endless.horse'  # URL = Uniform Resource Locator = webová adresa
response = requests.get(URL)  # Používáme HTTP metodu GET
print('STATUS:', response.status_code)  # Status code: 200 = OK, 404 = Not Found...
print('TEXT:', response.text[-700:])  # Posledních 700 znaků ze stáhnutého textu
```

```
STATUS: 200
TEXT: le="padding-top: 222px">
        <pre>   ,
_,,)\.~,,._
(()`  ``)\))),,_
 |     \ ''((\))))),,_          ____
 |6`    |   ''((\(()) "-.____.-"    `-.-,
 |    .'\     ''))))'                  \)))
 |   |   `.      ''                     ((((
 \, _)     \/                            |))))
  `'         |                           (((((
             \                    |       ))))))
              `|    |            ,\     /((((((
               |   / `-._______.&lt;  \   |  )))))
               |   |  /            `. \  \  ((((
               |  / \ |              `.\  | (((
              \  | | |                )| |  ))
               | | | |                || |  '</pre>
<a href="legs.html"></a>
</div>
  </body>
</html>
```

Modul re

- Regulární výraz = *regular expression* = *regex* = *RE*
- Způsob jak zapsat obecně vzorek textu, který chceme vyhledat/nahradit/...

[21]:
```
import re

text = 'Helloooo! She sells sea shells. Good as hell!'

re.findall('[Hh]ello*', text)
```

[21]:
```
['Helloooo', 'hell', 'hell']
```

[22]:
```
re.findall(R'\b[Hh]ello+\b', text)  # R znamena raw string - \ ztraci␣
 ↪specialni funkci
```

[22]:
```
['Helloooo']
```

[23]:
```
re.sub(R'\b[Hh]ello+\b', 'Ciao', text)
```

[23]:
```
'Ciao! She sells sea shells. Good as hell!'
```

[24]:
```
re.findall('S.*!', text)
```

[24]:
```
['She sells sea shells. Good as hell!']
```

Vysvětlení

- `[Hh]` – jeden znak z výčtu (`'H'` nebo `'h'`)
- `o*` – libovolný počet (včetně 0) opakování znaku `o` (`''` nebo `'o'` nebo`'oo'`...)
- `o+` – aspoň 1 opakování znaku `o` (`'o'` nebo`'oo'`...)
- `.` – libovolný znak
- `.*` – libovolný počet libovolných znaků
- `\b` – hranice slova
- Další možnosti použití:
  - https://docs.python.org/3.8/library/re.html
  - https://docs.python.org/3.7/howto/regex.html
- Pozor, pravidla `re` a `glob` jsou jiná!

Další příklady

[25]:
```
text = 'Tak se mějte hezky, já jdu žehlit a pokračoval v cestě.'
re.findall('\w+', text)  # \w = word characters = [a-zA-Z0-9_]
```

[25]:
```
['Tak',
 'se',
 'mějte',
 'hezky',
 'já',
 'jdu',
 'žehlit',
 'a',
 'pokračoval',
 'v',
 'cestě']
```

[26]:
```
from pathlib import Path

text = Path('no_side_effects.txt').read_text()

re.findall('"(.+)" *- *([0-9]{1,2}:[0-9]{2})', text)
```

[26]:
```
[('Poem', '6:23'),
 ('Flash', '5:24'),
 ('From Red to Rusk', '4:48'),
 ('Broken Pictures', '4:23'),
 ('Shake-up', '8:10'),
 ('Trio Four', '4:36'),
 ('No Side Effects', '3:12'),
 ('Frame Three', '6:30'),
 ('Shag Bark Hickory', '2:25'),
 ("Let's See", '3:57'),
 ('Ruddy', '4:20'),
 ('Vermillon', '4:12'),
 ('When the Winds Blow', '6:11'),
 ('Parched Plain', '13:38'),
 ('Shore Line', '5:22'),
 ('An Afternoon Walk', '6:02'),
 ('Enfold', '6:56'),
 ('Frame Two', '2:51'),
 ('They Danced', '3:35'),
 ('Ride', '3:41'),
 ('Here We Go', '3:39'),
 ('Rolling', '6:15'),
 ('Yellow Night', '6:41'),
 ('Sway', '2:53')]
```

[ ]: