C2184 Úvod do programování v Pythonu

# 11. 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 (https://cs.wikipedia.org/wiki/CSV)

- Tabulka:

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

- 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

## Čtení

In [47]:
```
with open('tabulka_auta.csv', 'r') as f:
    print(f.read())
```

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

In [48]:
```
import csv

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

```
['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

In [49]:
```
with open('tabulka_auta.csv', 'r') as f:
    reader = csv.reader(f)
    tabulka = list(reader)
tabulka
```

Out[49]:
```
[['Rok výroby', 'Značka', 'Model', 'Cena'],
 ['1995', 'Opel', 'Vectra', '45000'],
 ['1998', 'Škoda', 'Felicia', '80000'],
 ['2002', 'Škoda', 'Octavia', '70000']]
```

In [50]:
```
with open('tabulka_auta.csv') as f:
    csvreader = csv.DictReader(f)
    for radek in csvreader:
        print(dict(radek))
```

```
{'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

In [51]:
```
vzdalenosti = [['',     'Brno', 'Praha', 'Ostrava'],
               ['Brno',      0,     202,       165],
               ['Praha',   202,       0,       362],
               ['Ostrava', 165,     362,         0]]
```

In [52]:
```
with open('vzdalenosti.csv', 'w') as f:
    csvwriter = csv.writer(f)
    csvwriter.writerows(vzdalenosti)
```

In [53]:
```
with open('vzdalenosti.csv') as f:
    print(f.read())
```

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

## Zápis speciálních znaků

- Tabulka:

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

- CSV:

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

## Parametry pro upřesnění formátu

- `delimiter` – oddělovač sloupců (default `','`)
- `quotechar` – vyčlenění polí se speciálními znaky (default `'"'`)
- `doublequote` – zdvojení quotecharu ruší jeho funkci (default `True`)
- `escapechar` – ruší funkci speciálních znaků (delimiteru a quotecharu) (default `None`)
- `dialect` – nastavení více parametrů současně (např. `'excel'`)

In [54]:
```
with open('vzdalenosti.csv', 'w') as f:
    csvwriter = csv.writer(f, delimiter=';', quoting=csv.QUOTE_NONNUMERIC)
    csvwriter.writerows(vzdalenosti)
```

In [55]:
```
with open('vzdalenosti.csv') as f:
    print(f.read())
```

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

# Formát JSON

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

| Python | JSON | Poznámka |
|---|---|---|
| int/float `5, 10.2` | `5, 10.2` | |
| řetězec `'ahoj'` | `"ahoj"` | vždy dvojité uvozovky |
| `True, False` | `true, false` | |
| None | null | |
| seznam `[]`, *n*-tice `()` | pole `[]` | načte se vždy jako seznam |
| slovník `{}` | objekt `{}` | 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

## Čtení

In [59]:
```
with open('bob.json') as f:
    bob = f.read()
print(type(bob))
print(bob)
```

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

In [60]:
```
import json

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

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

In [61]:
```
text = '{ "name": "John", "age": 35, "married": true, "cars": ["Mercedes", "BM
W", "Volkswagen"] }'
```

In [62]:
```
john = json.loads(text)
print(type(john))
print(john)
```

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

## Zápis

In [64]:
```
alice = {'name': 'Alice', 'age': 28, 'married': False, 'cars': ('Ford', 'Traban
t'), 10: 20 }
```

In [65]:
```
with open('alice.json', 'w') as f:
    json.dump(alice, f)
```

In [66]:
```
with open('alice.json') as f:
    print(f.read())
```

```
{"name": "Alice", "age": 28, "married": false, "cars": ["Ford", "Trabant"], "1
0": 20}
```

In [16]:
```
text = json.dumps(alice)
print(type(text))
print(text)
```

```
<class 'str'>
{"name": "Alice", "age": 28, "married": false, "cars": ["Ford", "Trabant"], "1
0": 20}
```

In [21]:
```
text = json.dumps(alice, indent=4)
print(text)
```

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

# Formát XML

- *Extensible Markup Language*
- https://cs.wikipedia.org/wiki/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 (https://lxml.de))
- Externí balíček, nutno doinstalovat pomocí pipu
- (Nemusíte umět)

## 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!
- (Nemusíte umět)

## 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):
  - Povinné
  - Volby/přepínače/*options*
    - Začínají `-` (jednopísmenné) nebo `--` (vícepísmenné)
    - Mohou mít vlastní parametry

- Soubor `make_statistics.py`:

In [ ]:
```
import argparse

parser = argparse.ArgumentParser()
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:

```
$ python3 make_statistics.py
$ python3 make_statistics.py --help
$ python3 make_statistics.py data.csv --stat median --header --verbose
$ python3 make_statistics.py data.csv -s median -Hv
```

## 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*)

In [35]:

```
import requests

URL = 'http://endless.horse'  # URL = Uniform Resource Locator = webová adresa
odpoved = requests.get(URL)  # Používáme HTTP metodu GET
print('STATUS:', odpoved.status_code)  # Status code: 200 = OK, 404 = Not Foun
d...
print('TEXT:', odpoved.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/...
- (Nemusíte umět)

- Ukázky RE:

In [68]:
```
import re

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

vzorek1 = re.compile('[Hh]ello*')
vzorek1.findall(text)
```

Out[68]:
```
['Helloooo', 'hell', 'hell']
```

In [45]:
```
vzorek2 = re.compile(r'\b[Hh]ello+\b')
vzorek2.findall(text)
```

Out[45]:
```
['Helloooo']
```

In [46]:
```
vzorek2.sub('Ciao', text)
```

Out[46]:
```
'Ciao! She sells sea shells. Good as hell!'
```

Vysvětlení:

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