

C2184 Úvod do programování v Pythonu (2021)

10. 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>
- `csv.reader` a `csv.write` se vytváří z již otevřeného souboru

- Při otevírání souboru je nutné použít `newline= ''`, jinak se mohou vypisovat prázdné řádky navíc (zejména na Windows).

Čtení

```
[1]: with open('data/auta.csv', 'r', encoding='utf8') as f:
      print(f.read()) # Klasické čtení bez CSV readeru
```

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

```
[2]: import csv
      from pathlib import Path

      with open('data/auta.csv', 'r', encoding='utf8', newline='') as f:
          reader = csv.reader(f)
          print(next(reader)) # Čte jeden řádek
          print('-----')
          for row in reader: # Čte postupně všechny řádky
              print(row)
```

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

```
[3]: with open('data/auta.csv', 'r', encoding='utf8', newline='') as f:
      reader = csv.reader(f)
      table = list(reader) # Čte všechny řádky najednou → seznam seznamů
      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('data/auta.csv', encoding='utf8', newline='') as f:
      reader = csv.DictReader(f, ['year', 'brand', 'model', 'price'])
      for row in reader:
```

```
print(row)
```

```
{'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('data/auta.csv', encoding='utf8', newline='') as f:  
      reader = csv.DictReader(f)  
      for row in reader:  
          print(row)
```

```
{'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('data/distances.csv', 'w', encoding='utf8', newline='') as f:  
      ↪ f:  
          writer = csv.writer(f)  
          writer.writerows(distances)
```

```
[8]: print(Path('data/distances.csv').read_text(encoding='utf8'))
```

```
,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.

```
[9]: with open('data/auta2.csv', encoding='utf8', newline='') as r:
      reader = csv.reader(r)
      for row in reader:
          print(row)
```

```
['1995', 'Opel', 'Vectra', 'klimatizace, střešní okno', '45000']
['1998', 'Škoda', 'Felicia "Fun"', '', '80000']
['2002', 'Škoda', 'Octavia', 'klimatizace, ABS\nbouraná', '70000']
```

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')

```
[10]: with open('data/distances.csv', 'w', encoding='utf8', newline='') as f:
      ↪ f:
```

```
writer = csv.writer(f, delimiter=';', quoting=csv.
↪QUOTE_NONNUMERIC)
writer.writerows(distances)
```

```
[11]: print(Path('data/distances.csv').read_text(encoding='utf8'))
```

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

```
[12]: with open('data/distances.csv', encoding='utf8', newline='') as f:
      for row in csv.reader(f, delimiter=';', quoting=csv.
↪QUOTE_NONNUMERIC):
          print(row)
```

```
['', 'Brno', 'Praha', 'Ostrava']
['Brno', 0.0, 202.0, 165.0]
['Praha', 202.0, 0.0, 362.0]
['Ostrava', 165.0, 362.0, 0.0]
```

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/>
- Ukázka:

```
{
    "name": "John",
    "age": 35,
    "married": true,
    "cars": [
        "Mercedes",
        "BMW",
        "Volkswagen"
    ]
}
```

- Mapování na typy Pythonu:

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

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

```
[13]: with open('data/bob.json', encoding='utf8') as f:
        bob = f.read() # Načte řetězec
bob
```

```
[13]: '{\n  "name": "Bob",\n  "age": 30,\n  "married": false,\n  "cars": [\n    ↪ "Ford",\n    "BMW", "Fiat"]\n}\n'
```

```
[14]: import json

with open('data/bob.json', encoding='utf8') as f:
    bob = json.load(f) # Načte slovník
bob
```

```
[14]: {'name': 'Bob', 'age': 30, 'married': False, 'cars': ['Ford', 'BMW',  
↪ 'Fiat']}
```

```
[15]: john = json.loads('{ "name": "John", "age": 35, "married": true,  
↪ "cars": ["Mercedes", "BMW", "Volkswagen"] }')  
john
```

```
[15]: {'name': 'John',  
      'age': 35,  
      'married': True,  
      'cars': ['Mercedes', 'BMW', 'Volkswagen']}
```

Zápis

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

```
[17]: with open('data/alice.json', 'w', encoding='utf8') as f:  
      json.dump(alice, f) # Zapisuje do souboru
```

```
[18]: print(Path('data/alice.json').read_text(encoding='utf8'))
```

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

```
[19]: with open('data/alice.json', 'w', encoding='utf8') as f:  
      json.dump(alice, f, indent=4) # Zapisuje do souboru, s odsazením  
↪ 4
```

```
[20]: print(Path('data/alice.json').read_text(encoding='utf8'))
```

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

```
[21]: text = json.dumps(alice) # Nezapisuje do souboru, ale vrací řetězec  
text
```

```
[21]: '{"name": "Alice", "age": 28, "married": false, "cars": ["Ford",  
↪ "Trabant"]}'
```

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"}}

Přehled užitečných modulů

- Standardní knihovna: <https://docs.python.org/3.8/library/>
- Python Package Index: <https://pypi.org/>
- Měli byste umět:
 - math
 - random
 - statistics
 - argparse
 - shutil, os, sys z minula (zhruba)

Modul math

- <https://docs.python.org/3/library/math.html>
- Matematické funkce

```
[22]: import math
      math.comb(15, 3)  # Kombinační číslo 15 nad 3
```

[22]: 455

Modul cmath

- <https://docs.python.org/3/library/cmath.html>
- Matematické funkce nad komplexními čísly
- Komplexní složka se označuje j a vždy má před sebou číslici

```
[23]: import cmath
      cmath.sqrt(-1)
```

[23]: 1j

```
[24]: cmath.sin(2+1j)
```

[24]: (1.4031192506220405-0.4890562590412937j)

Modul random

- <https://docs.python.org/3/library/random.html>
- Generování pseudo-náhodných čísel

```
[25]: import random
      random.random() # Náhodné reálné číslo z intervalu [0, 1)
```

[25]: 0.1593346609756927

```
[26]: random.randint(0, 100) # Náhodné celé číslo z intervalu [0, 100]
```

[26]: 65

```
[27]: random.choice(['červená', 'zelená', 'modrá', 'černá']) # Náhodný
      ↪výběr
```

[27]: 'zelená'

```
[28]: random.gauss(5, 1) # Náhodné číslo z normálního rozdělení ( $\mu=5$ ,  $\sigma=1$ )
```

[28]: 4.75444577014393

Modul statistics

- <https://docs.python.org/3/library/statistics.html>
- Statistické funkce

```
[29]: import statistics
      data = [8, 44, 38, 13, 76, 30, 49, 27, 30, 74, 39, 65, 50, 30, 33,
      ↪43, 66, 25, 15, 77]
      statistics.mean(data) # Aritmetický průměr
```

[29]: 41.6

```
[30]: statistics.median(data) # Medián
```

[30]: 38.5

```
[31]: statistics.mode(data) # Modus (nejčastější hodnota)
```

```
[31]: 30
```

```
[32]: statistics.stdev(data) # Směrodatná odchylka vzorku
```

```
[32]: 21.02980341275085
```

```
[33]: statistics.pstdev(data) # Směrodatná odchylka populace
```

```
[33]: 20.497316897584426
```

Modul argparse

- <https://docs.python.org/3/library/argparse.html>
- Předávání argumentů z příkazového řádku
- Stejný účel jako `sys.argv`, ale sofistikovanější a hezčí pro uživatele

Argumenty z příkazového řádku (netýká se pouze Pythonu):

- Poziční
\$ program **arg1 arg2**
- Volby/přepínače/options/switches/flags
 - Modifikují chování programu
 - Začínají - (jednopísmenné) nebo -- (vícepísmenné)
 - Bez parametrů:
\$ program arg1 arg2 **-s**
\$ program arg1 arg2 **--switch**
 - S parametrem:
\$ program arg1 arg2 **-o value**
\$ program arg1 arg2 **--option value**
- Různé kombinace:
\$ program --option value -s arg1 arg2 --switch
- Volba -h nebo --help -> program nemá nic dělat, pouze vypsát nápovědu:
\$ program --help

Načtení argumentů pomocí argparse

1. Vytvoříme parser
2. Přidáme parseru jednotlivé argumenty a volby
3. Načteme (zparsujeme) argumenty a volby pomocí parseru
 - Soubor pizza.py:

```
[ ]: import argparse

def main() -> None:
    # Vytvoření parseru
    parser = argparse.ArgumentParser(description='Demo module for_
    ↪ordering pizza')

    # Poziční argumenty (povinné)
    parser.add_argument('address', help='Address for delivery',
                        type=str)
    parser.add_argument('pizza', help='Type of pizza',
                        choices=['Margherita', 'Funghi', '4Formaggi'])
    # Volba typu bool
    parser.add_argument('-x', '--spicy',
                        help='Extra spicy',
                        action='store_true')
    # Volba typu str s výběrem
    parser.add_argument('-s', '--size',
                        help='Pizza size',
                        choices=['small', 'medium', 'large'],
                        default='medium')
    # Volba typu float
    parser.add_argument('-t', '--tip',
                        help='Tip for the delivery ($)',
                        type=float,
                        default=0.0)

    # Načtení argumentů a voleb pomocí parseru
    args = parser.parse_args()

    print('Address:   ', args.address)
    print('Pizza type:', args.pizza)
    print('Size:      ', args.size)
    print('Spicy:     ', args.spicy)
    print('Tip:       ', args.tip)

if __name__ == '__main__':
    main()
```

- Spouštíme z příkazového řádku:

```
$ python pizza.py
$ python pizza.py --help
$ python pizza.py 'Kamenice 5' Funghi
$ python pizza.py 'Kamenice 5' 4Formaggi --size large --spicy --tip 1.20
$ python pizza.py 'Kamenice 5' 4Formaggi -s large -x -t 1.20
```

- (Uvozovky jsou nutné u víceslovných argumentů, aby shell poznal, kde argument začíná a končí. Python už dostane řetězec bez uvozovek.)

Modul datetime

- <https://docs.python.org/3/library/datetime.html>
- Práce s časovými údaji
- Základní typy:
 - datetime - datum a čas
 - timedelta - trvání
 - date - datum
 - time - čas
 - timezone - časová zóna

```
[34]: from datetime import datetime, timedelta
```

```
start = datetime.now()
math.factorial(500_000)
end = datetime.now()
calculation_time = end - start
print(start)
print(end)
print(calculation_time)
```

```
2021-11-26 14:43:44.639741
2021-11-26 14:43:46.903156
0:00:02.263415
```

```
[35]: today = datetime.now().date()
in_1_week = today + timedelta(weeks=1)
print(f'Today is: {today}')
print(f'In one week it will be: {in_1_week}')
```

```
Today is: 2021-11-26
In one week it will be: 2021-12-03
```

- Formátování času

```
[36]: datetime.now().strftime('%A, %d %B %Y, %H:%M:%S')
```

[36]: 'Friday, 26 November 2021, 14:44:00'

- Parsování času

```
[37]: datetime.strptime('1. 7. 2000, 13:30:05', '%d. %m. %Y, %H:%M:%S')
```

```
[37]: datetime.datetime(2000, 7, 1, 13, 30, 5)
```

Modul itertools

- <https://docs.python.org/3/library/itertools.html>
- Různé možnosti iterování

```
[38]: import itertools
      for pair in itertools.combinations(['A', 'B', 'C'], 2):
          print(*pair)
```

A B
A C
B C

```
[39]: for pair in itertools.permutations(['A', 'B', 'C'], 2):
      print(*pair)
```

A B
A C
B A
B C
C A
C B

```
[40]: for letter, number in itertools.zip_longest('ABC', [1, 2, 3, 4, 5],
      ↪ fillvalue='?'):
      print(letter, number)
```

A 1
B 2
C 3
? 4
? 5

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

- <https://docs.python.org/3/library/pickle.html>
- 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

- <https://pypi.org/project/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*)

```
[41]: import requests

URL = 'http://endless.horse' # URL = Uniform Resource Locator = ↵
    ↪ webová adresa
response = requests.get(URL) # Používáme HTTP metodu GET
```

```
STATUS: 200
TEXT: le="padding-top: 222px">
      <pre>    ,
```

Modul re

- ```
[42]: import re
```

15

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

```
[43]: re.findall(r'\b[Hh]ello+\b', text)
```

```
[43]: ['Helloooo']
```

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

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

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

```
[45]: ['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
- \w = word characters = [a-zA-Z0-9\_]
- \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

```
[46]: text = 'Tak se mějte hezky, já jdu žehlit a pokračoval v cestě.'
re.findall(r'\w+', text)
```

```
[46]: ['Tak',
 'se',
 'mějte',
 'hezky',
 'já',
 'jdu',
 'žehlit',
 'a',
```

```
'pokračoval',
'v',
'cestě']
```

```
[47]: from pathlib import Path
text = Path('data/no_side_effects.txt').read_text()
re.findall(r'"(.+)" *- *([0-9]{1,2}:[0-9]{2})', text)
```

```
[47]: [('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')]
```

## Další užitečné moduly

- subprocess
- email
- codecs
- warnings
- numpy
- matplotlib
- sklearn
- pandas

• ...