

Práce se soubory.

Práce se soubory. Textový režim. Binární režim. Serializace.

Tomáš Bayer | bayertom@natur.cuni.cz

Katedra aplikované geoinformatiky a kartografie, Přírodovědecká fakulta UK.

Obsah přednášky

- 1 Práce se soubory
- 2 Otevření/uzavření souboru
- 3 Textové soubory
- 4 Serializace

1. Soubory a operace

Standardní vstup nepostačuje \Rightarrow zpracování vstupních dat.

Obsah RAM se po vypnutí smaže, výsledek nutno uložit.

Soubor:

Permanentní uložení dat na paměťovém médiu.

Různé atributy a přístupová práva dle operačního systému.

Podpora široké spektra operací:

- *Základní operace:*

Python: Otevření, čtení, zápis, uzavření.

- *Pokročilé operace:*

Externí knihovny: Komprese, dekomprese, serializace, rastrová, vektorová data, grafika, multimédia...

Mnoho podporovaných formátů (Python), velké množství externích knihoven:

TXT, CSV, HTML, XML, JSON, SHP, JPG, TIFF, ZIP, ...

Soubor reprezentován **streamem** (proudem).

Operace se souborem se **nemusí** podařit:

Neexistuje, nelze přečíst/zapsat, chybná přístupová práva, nulová velikost...

Využití výjimek.

2. Práce se soubory, módy

Práce se soubory v Pythonu:

- 1 Ověření existence souboru a přístupu k němu.
- 2 Otevření souboru.
- 3 Čtení nebo zápis do souboru.
- 4 Uzavření souboru.

Bod 1 řešen buď výjimkami / “dotazem”.

V případě neúspěšné operace \Rightarrow vznik výjimky. Dva důležité indikátory:

EOL (End of Line), EOF (End of File).

Dva módy (režimy) při práci se soubory:

- *Binary mode*
Reprezentace znaků (čísla, písmena) v binární/hexadecimální soustavě.
Přímý zápis/čtení po bytech (8 bitů).
Nativní pro PC, přímo nečitelné.
- *Text (ASCII) mode*
Konverze z binárního do textového formátu.
Čitelné, avšak bez formátování.

Python podporuje čtení i zápis v obou módech.

Vstupem resp. výstupem textový resp. binární soubor.

3. Textový soubor

Představován posloupností řádek.

Každá řádka tvořena posloupností znaků tvořící slova.

Na konci řádky EOL (End Of Line).

Nezachovává formátování.

Čitelný, lze snadno editovat:

```
105 723304.23 1023067.98 208.37
106 723297.17 1023071.04 206.92
107 723301.55 1023068.64 207.76
```

Pro některé znaky nepostačují písmena + číslice + speciální znaky: *Escape Sequence*

Uvozující znak \ (Backslash).

Escape Sequence	UNICODE	Název	Význam
\n	\u000A	Line Feed	Nová řádka
\r	\u000D	Carriage Return	Návrat na začátek řádku
\t	\u0009	Horizontal Tab	Tabulátor
\\	\u005C	Backslash	Zpětné lomítko
\'	\u002C	Single Quote	Apostrof
\"	\u0022	Double Quote	Uvozovky

Odchytky EOL v implementacích pod různými OS:

```
\n + \r  Windows
\n      GNU/Linux, \r nema specialni vyznam!
\r      MacOS
```

4*. Binární soubor

Tvoření posloupností bytů zapsaných v hexadecimální soustavě, hodnoty 0-255.

Používán pro zápis jiných než textových informací.

Není přímo čitelný bez znalosti příslušného formátu.

Nelze prohlížet v běžném editoru (TEXT Mode):

```
' AÄ~č řSä164'ÁöÿÔX/0ÁjLôÔŽ&ÁÛÎ7Kr/Á @ F¶ó}Ö!'ÁZd;žv
/Á<lç{~!'ÁázⓈAv/Á F¶ó}Ö!'ÁHáz''¬v/ÁÑ"ŪqÄ!'ÁázⓈAv/Á<lç
{~!'Á%U,v/Á#Ūû>Đ!'ÁZd;žv/ÁF¶ó}Ö!'ÁHáz''¬v/Á @ °rh'E!'
```

Použití prohlížeče s podporou HEX Mode:

```
27 20 41 c3 81 7e c4 8d | 20 c5 99 53 c4 83 c4 ba
36 34 27 c3 81 c3 b4 c3 | bd c3 94 58 2f 30 c3 81
6a c4 bd c3 b4 c3 93 c5 | bd 26 c3 81 c5 ae c3 8e
```

Struktura binárního formátu:

Souvislý blok dat, není dělen na řádky, slova.

Často sdružovány po 8 (1 byte = 8 bitů).

Záhlaví představováno hlavičkou a identifikátorem formátu (nepovinné).

Následuje vlastní datová oblast.

Specifické pro různé formáty.

Používán pro kompilovaný kód (exe, bin).

Ukládání grafiky, zvuků, dokumentů, vektorových data.

5. Otevření souboru

První krok, který nutno učinit před čtením/zápisem.

Soubor lze otevřít v různých módech.

Nutno specifikovat:

- zda pracujeme s binárním/textovým souborem,
- jakou operaci chceme provádět \Rightarrow mód.

Označení	Mód	Popis
r	Read	Otevření souboru pro čtení (read only, default).
w	Write	Otevření souboru pro zápis, původní obsah přepsán.
a	Append	Otevření souboru pro zápis, přidání nových dat na konec
r+	Read/Write	Kombinace čtení i zápisu.
t	Text	Otevření v textovém režimu (default)
b	Binary	Otevření v binárním režimu.
x	Exclusive	Vytvoření souboru + otevření pro zápis.

Módy lze je kombinovat: rt+ nebo wb+.

Příkaz `open()`, vrací file object, resp. handler.

```
file_object = open("file_name", "mode")
```

Otevření souboru v aktuální složce:

```
f=open('test.txt') #Default, open in rt mode
```

Otevření souboru s uvedením plné cesty:

```
f=open('E:/Data/test.txt') #Do not use backslash, open in rt mode
```

Otevření souboru se specifikací módu:

```
f=open('img.jpg', 'rb+') #Open in read and write mode, binary file
```

6. Vlastnosti proudu (streamu)

Existuje několik nástrojů které umožňují zjistit stav streamu:

```
f = open('E:/Tomas/Text/test.txt', 'r')
```

Název souboru/proudu:

```
>>> f.name  
'E:/Tomas/Text/test.txt'
```

Zjištění stavu proudu (otevřený/uzavřený):

```
>>> f.closed  
False
```

Dotaz na mód souboru:

```
>>> f.mode  
'r'
```

Dotaz, zda lze ze souboru číst:

```
>>> f.readable()  
True
```

Dotaz, zda lze do souboru zapisovat:

```
>>> f.writable()  
False
```


7. Uzavření souboru

Po skončení operací se souborem nutno provést jeho uzavření.

OS při práci se souborem:

- používá deskriptor ukazující na otevřený soubor,
- alokuje systémové prostředky.

Neuzavření souboru:

Deskriptor násilně ukončen OS až po skončení procesu.

Pokud proces není ukončen \Rightarrow nestandardní chování + únik systémových prostředků.

Možné problémy:

- Do souboru nejde zapisovat (jeví se jako read only).
- Zapsané změny se nemusejí uložit.

Uzavření souboru, metoda `close()`

```
f.close()    #Close the file
```

Nutno vždy provést.

8. Efektivnější práce se soubory

Využití příkazu `with`.

Univerzální nástroj, lze použít pro čtení resp. zápis ze resp. do souboru. Bezpečný, provádí automatické uzavření souboru.

Doporučován jako výchozí nástroj pro práci se soubory.

Výhodou jednodušší syntaxe:

```
with open(file_name) as file_object:  
    #perform some operation
```

Lze spojit i s dotazem, zda soubor existuje:

```
from os import *  
if path.exists('test.txt'):  
    with open('test.txt', 'rt+') as f:  
        data = f.read()
```

#Read data from file

9. Čtení z textového souboru I.

Předpoklad: soubor otevřen v módu umožňujícím čtení.

Provádí se, dokud nedosáhneme EOF, poté nutno soubor uzavřít.

Tři varianty čtení dat ze souboru:

- načtení souboru nebo jeho části,
- načtení 1 řádky souboru,
- načtení celého souboru po řádkách.

Načtení souboru nebo jeho části.

Výsledek načten do proměnné typu `String`.

Lze specifikovat, kolik bajtů se má načíst (1 byte = 1 znak).

Použita funkce `read()`, dvě varianty.

```
with open('test.txt', 'rt+') as f:           #Open file
    data1 = f.read()                         #Read entire file
    data2 = f.read(5 )                      #Read 5 bytes of file
    data3 = f.read(10)                      #Read next 10 bytes of file
```

Vytisknutí obsahu souboru:

```
'We are learning Python,\nthe best language.' #EOL \n not removed
'We ar'
'e learning'
```

Procházení načteného souboru po znacích

```
for line in data:                            #Browse file by chars
    print(line)                             #Do something
```

Odstranění white spaces:

Odstranění EOL, načtení do seznamu `Stringů`

```
>>>f.read().splitlines()
['We are learning Python,', 'the best language.']
```

10. Čtení z textového souboru II.

Čtení souboru po řádkách.

Načte s každým zavoláním jednu řádku souboru.

Následně skok na další řádku.

Výsledek uložen do řetězce typu String.

```
>>> line1 = f.readline()
>>> line2 = f.readline()
'Monday\n'           #EOL \n not removed
'Tuesday\n'          #EOL \n not removed
```

Ukázka načtení celého souboru po řádkách:

```
with open('test.txt', 'rt+') as f:
    l = f.readline()           #Read first line
    while l:                   #While not EOL
        l = f.readline()       #Read next line
```

Načtení celého souboru po řádkách.

Soubor načten jednorázově, po řádkách.

Výsledek uložen jako seznam Stringů.

```
>>> lines = file.readlines()
>>> print(lines)
['Monday\n', 'Tuesday\n', 'Wednesday'] #EOL \n are not removed
```

11. Čtení souborů ve formátu CSV

Často používaný textový formát pro výměnu tabulkových dat.

CSV = Comma-Separated Values.

Položky odděleny čárkou, středníkem nebo tabulátorem.

Použití: tabulky, matice, seznamy souřadnic.

```
4035 743187.400 1044836.320 195.290
4036 743188.245 1044832.390 195.270
4037 743185.288 1044831.581 195.430
4038 743182.379 1044829.677 195.690
4039 743180.266 1044826.193 195.760
```

Modul CSV reader:

Podpora čtení / zápisu do CSV souborů.

Pohodlnější než readlines().

Možno definovat oddělovač.

```
from csv import *
with open('coord.csv','rt') as f:
    points = csv.reader(f, delimiter = ' ')
    for p in points:
        print(p)
```

12*. Pohyb v otevřeném streamu

V otevřeném souboru se můžeme pohybovat.

Lze se posunout na začátek, konec či konkrétní pozici.

Zjištění aktuální pozice:

Aktuální pozice ve streamu udávána v bytech, nabývá hodnot $\langle 0, n-1 \rangle$.

Po načtení celého souboru ukazuje počet znaků v souboru.

```
>>> f.tell()    #Dotaz na aktualni pozici v souboru
40
```

Posun na konkrétní pozici:

Specifikace pozice v souboru, na kterou se chceme posunout.

Pohyb v souboru oběma směry o k znaků.

```
seek(o kolik, odkud)
```

Tři módy nastavení referenčního bodu:

Mód	Vyjádření	Popis	Mód
0	SEEK_SET	Referenční bod na začátek souboru.	t,b
1	SEEK_CUR	Referenční bod na aktuální pozici.	b
2	SEEK_END	Referenční bod na konec souboru.	b

```
>>> f.seek(0)    #Skok na počátek souboru
0
>>> f.seek(10, 0)    #Skok na 10. znak
>>> f.read()
'rnig Python,\nthe best language.'
```

13. Zápis do souboru

Předpoklad: soubor otevřen v módu umožňujícím zápis.

Tři základní režimy zápisu:

- *Režim w (Write Mode):*
Původní obsah souboru nahrazen.
- *Režim a (Append Mode)*
Připisujeme na konec souboru.
Zachován původní obsah.
- *Režim x (Exclusive Mode)*
Vytvoření nového souboru + otevření pro zápis.
Pokud soubor již existuje, výjimka.

Po skončení zápisu soubor nutno uzavřít, využítí funkce `with()`.

Zápis do souboru po řádkách.

Funkce `write()`, zapisuje řetězec do otevřeného proudu.

```
write(string)
```

Zápis do souboru, smazání původního obsahu:

```
with open("test.txt", 'w') as f:
    f.write("Monday\n")      #Delete all + create first line
    f.write("Tuesday\n")    #Create second line
```

Přidání na konec souboru, ponechání původního obsahu:

```
with open("test.txt", 'a') as f:
    f.write("Wednesday\n")  #Append new line to existing content
```

14. Zápis do CSV souboru

Použití modulu CSV Writer.

Zápis dat do formátu CSV, textový režim.

Možno definovat oddělovač.

Zápis probíhá po jednotlivých řádkách.

```
from csv import *  
with open('coord.csv', mode='w') as file:  
    write = writer(file, delimiter=',')  
  
    write.writerow(['743179.558', '1044817.180', '196.110'])  
    write.writerow(['743177.185', '1044815.932', '196.360'])  
    write.writerow(['743176.454', '1044817.605', '196.330'])  
    write.writerow(['743177.303', '1044818.528', '196.290'])
```

Alternativně lze použít i modul pandas.

15*. Serializace a deserializace

Serializace:

Převod datové struktury uložené v paměti na posloupnost bytů.

Tuto lze následně uložit na paměťové médium.

Deserializace:

Opáčný postup, zpětná rekonstrukce posloupnosti bytů na datovou strukturu.

V praxi mají obě operace velký význam.

Možnost uložení resp. načtení aktuálního obsahu datových struktur do resp. z souboru.

Výsledné soubory textové či binární.

Data mohou být použita při dalším spuštění.

Využití pro ladění programů, ukládání pracovních dat,...

Nevýhoda: obtížná editace obsahu (binární soubory).

Podpora serializace/deserializace pro:

- základní datové typy,
- dynamické datové struktury,
- funkce,
- třídy,

Využití JSON.

16*. Serializace s využitím JSON

Formát JSON (Java Script Object Notation):

Výměnný formát pro přenos dat, podmnožina JavaScript formátu.

Textový formát, snadná čitelnost, přehlednost.

Výhodou textová struktura, snadno lze editovat.

Podpora JSON v Pythonu:

V Pythonu implementace JSON založena na Nested Dictionary.

Načtený JSON parsován na Dictionary.

Univerzální formát pro serializaci a deserializaci objektů v Pythonu.

Uložená data snadno čitelná.

Existuje i jeho prostorová varianta: GeoJSON.

Jedna z možností vstupu prostorových dat do Pythonu.

Serializace/deserializace JSON:

Uložení a načtení, použity metody `dump()` a `load()`.

17*. Serializace a deserializace JSON, ukázka

Serializace objektu:

Uložení objektu jako do souboru JSON.

```
dump(data_object, file)
```

Soubor nutno otevřít ve správném módu, import modulu json.

```
import json
d = {'name' : 'John', 'surname': 'Smith',
     'job': 'cartographer', 'age' : '25'}
with open('test.json', 'w') as f:
    json.dump(d, f)
```

Ukázka souboru test.json:

```
{"age": "25", "surname": "Smith",
 "job": "cartographer", "name": "John"}
```

Deserializace objektu:

Načtení JSON souboru do Dictionary.

```
load(data_object, file)
```

Ukázka deserializace objektu:

```
with open('test.json', 'rb') as f:
    d2 = json.load(f)
```