Programování 1.11: Soubory a výjimky

Petr Čermák
cermak@mag.mff.cuni.cz

Katedra Fyziky Kondenzovaných Látek
MFF UK Praha

2020
forked from https://gitlab.kam.mff.cuni.cz/mj/prm1

Soubory

Soubor je pojmenovaná posloupnost **bajtů** (8-bitových hodnot).

Nás budou zajímat hlavně **textové soubory:**

- Skládají se ze **znaků** převedených na bajty nějakým **kódováním**, nejčastěji
  - ASCII („anglická abeceda“ o 95 znacích)
  - iso-8859-2 (navíc znaky východoevropských jazyků)
  - cp1250 (něco podobného specifického pro Windows)
  - UTF-8 (vícebajtové znaky, pokrývá většinu jazyků světa)
- Soubor je členěn na řádky, každý končí speciálním **znakem konce řádku** (pozor, různé operační systémy ukončují řádky různě, ale Python to překládá).

Jednoduchý příklad

open(jmeno_souboru, mode, encoding) otevírá soubor:

- `mode`:
  - `r` = čtení (default)
  - `w` = zápis (založí/vyprázdní soubor)
  - `a` = zápis na konec (append)
  - `r+` = čtení i zápis
  - `w+` = jako r+ ale založí/vyprázdní soubor
  - `rb` = binární čtení (bajty, ne text)
- `encoding`=kódování (default: podle OS)

Explicitnimu **close()** se můžeme vyhnout pomocí `with`.

```
1 f = open('soubor.txt', 'w')
2 f.write('Hej mistře!\n')
3 f.close()
4 # alternativně:
5 with open('soubor.txt', 'w') as f:
6     f.write('Hej mistře!\n')
```

Co můžeme dělat se souborem

Metody souborů:

- `f.write(text)` – zapíše text
- `f.read(n)` – přečte dalších *n* znaků (`""` na konci)
- `f.read()` – přečte všechny zbývající znaky
- `f.readline()` – přečte další řádek (včetně `\n`) nebo `""`
- `f.seek(...)` – přesune se na danou pozici v souboru

Další operace:

- `print(..., file=f)`
- `for line in f:` – cyklus přes řádky souboru

Pozor, řádky končí na `"\n"`, hodí se zavolat `rstrip()`.

Standardní vstup a výstup

Vždy je k dispozici:

- `sys.stdin` – standardní vstup (odtud čte `input()`)
- `sys.stdout` – standardní výstup (sem píše `print()`)
- `sys.stderr` – standardní chybový výstup

Hlášení chyb

test.py:

```
1 def f(x, y):
2     print(x/y)
3
4 f(1, 0)
```

Po spuštění dostaneme:

```
1 Traceback (most recent call last):
2   File "test.py", line 4, in <module>
3     f(1, 0)
4   File "test.py", line 2, in f
5     print(x/y)
6 ZeroDivisionError: division by zero
```

Výjimky

Chyba vygeneruje **výjimku**, například těchto typů:

- ZeroDivisionError – dělení nulou
- ValueError – chybný argument (třeba `log(-4)`)
- IndexError – přístup k indexu mimo rozsah
- KeyError – čtení neexistujícího klíče ve slovníku
- FileNotFoundError – otevření neexistujícího souboru
- MemoryError – došla paměť
- KeyboardInterrupt – běh programu přerušen pomocí Ctrl-C

Odchycení výjimky

```
1 try:
2     x, y = map(int, input().split())
3     print(x/y)
4 except ZeroDivisionError:
5     print("Nulou dělit neumím.")
6 except ValueError as x:
7     print("Chyba:", x)
8     print("Zadejte prosím dvě čísla.")
```

Výjimky jsou objekty, jejich typy jsou třídy.

Výjimka se umí převést na řetězec (`print` ji vypíše).

Výjimka má atributy (detaily toho, co se stalo a kde).

Typy výjimek tvoří hierarchii (třeba `FileNotFoundError` je potomkem `OSError`), `except` umí zachytit i obecnější typ.

Vyvolání výjimky

```
>>> raise RuntimeError("Jejda!")
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
RuntimeError: Jejda!

>>> if not condition:
>>>     raise AssertionError()
[...]
AssertionError

>>> assert 1 == 2
[...]
AssertionError

>>> assert 1 == 2, "Veselé vánoce!"
[...]
AssertionError: Veselé vánoce!
```

Úkoly na vánoční hodinu

- Vánoční úloha v recodexu (5 bodů, do konce hodiny, první vyhrává vánoční pivo)
- nebo:
- Spočítejte, kolik je v souboru řádků, slov a viditelných znaků (tedy bez mezer a konců řádků)
- Zkopíruje soubor do jiného souboru, aby řádky šly v opačném pořadí
- Najděte na každém řádku všechna čísla (oddělená mezerami) a vypíše jejich součet; slova, která nejsou čísly, ignoruje.