# Programování 1: Soubory a výjimky

Martin Mareš
mj@ucw.cz

Katedra Aplikované Matematiky
MFF UK Praha

2025

# 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

```
f = open('soubor.txt', 'w')
f.write('Hej mistře!\n')
f.close()
```

**open** otevírá soubor, dostává:

- jméno souboru
- mód přístupu:
  - `r` = čtení (default, pokud mód vynecháme)
  - `w` = zápis (založí/vyprázdní soubor)
  - `a` = zápis na konec (append)
  - `r+`/`w+` = čtení i zápis
  - `rb` = binární čtení (bajty, ne text)
- `encoding`=kódování (default: podle OS)

Explicitnimu **close** se můžeme vyhnout takto:

```
with open('soubor.txt', 'w') as f:
    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

# Formátování

```
>>> "{} leží {}".format("kočka", "na okně")
'kočka leží na okně'

>>> "{1} leží {0}".format("kočka", "na okně")
'na okně leží kočka'

>>> "{kdo} leží {kde}".format(kdo="pes", kde="v
boudě")
'pes leží v boudě'

>>> kdo="lenochod"
>>> kde="na větvi"
>>> f"{kdo} leží {kde}"
'lenochod leží na větvi'

>>> "{0:05} < {1:05}".format(1, 2)
'00001 < 00002'
```

(další varianty viz dříve)

# Hlášení chyb

**test.py:**

```
def f(x, y):
    print(x/y)

f(1, 0)
```

**Po spuštění dostaneme:**

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

```
try:
    x, y = map(int, input().split())
    print(x/y)
except ZeroDivisionError:
    print("Nulou dělit neumím.")
except ValueError as x:
    print("Chyba:", x)
    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!

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

>>> assert 1 == 2, "Svět se nezbláznil"
[...]
AssertionError: Svět se nezbláznil
```