# Soubory



## Otevření souborů

Při práci se soubory je musíme otevřít, provedeme nějaké operace s obsahem a zase je zavřeme.

V programu se k souboru přistupuje **sekvenčně**. To znamená, že čteme od jeho začátku, postupně po jednom řádku. Z programu obvykle soubor **otvíráme** jen pro **čtení** nebo jen **pro zápis**. Při zápisu můžeme vytvořit zcela nový soubor (nebo můžeme přepisovat obsah již existujícího souboru) nebo **obsah připojujeme na konec** (*append*) existujícího souboru.

**Otevření souboru** provedeme funkcí pro vytvoření objektu typu file (tento objekt je její návratovou hodnotou):

**open**(*nazev_souboru, typ*)

**Typ souboru:**

- **'r' –** Soubor bude otevřen pro čtení. Pokud neexistuje, nastane výjimka.
- **'r+' –** Soubor bude otevřen pro čtení a zápis. Pokud neexistuje, nastane výjimka.
- **'w' –** Soubor bude otevřen pro zápis. Existující soubor bude zkrácen na nulovou délku, neexistující soubor bude vytvořen.
- **'a' –** Soubor bude otevřen pro zápis na konec souboru. Neexistující soubor bude vytvořen.

## Uzavření souborů

Po ukončení práce se souborem je třeba soubor uzavřít, neboť se tím uvolní systémové zdroje. V jednoduchých skriptech není uzavírání souborů až tak nutné, protože při ukončení skriptu nebo programu se soubory uzavírají automaticky. Ve velkém projektu je skutečně potřebné soubory uzavírat, zvýší se tím spolehlivost běhu a předejde se vyčerpání systémových zdrojů a zhroucení programu.

**Pro uzavření souboru** se užívá **metoda close().**

## Funkce pro čtení a zápis textu nebo binárních dat

### Metody write() a writelines()

**Metoda write()** zapíše do souboru textový řetězec. Na konci nezapíše znak nového řádku, pokud ho tam chceme mít, musíme ho zapsat sami.

**Metoda writelines()** nemá právě výstižný název, protože nezapisuje řádky, ale vezme seznam řetězců uvedených v jejím argumentu a zapíše je jeden po druhém do příslušného souborového objektu, aniž je oddělí znakem konec řádku (pochopitelně, když řetězce končí znakem konec řádku, pak je zapíše jako řádky).

### Metody read(), readline() a readlines()

**Metoda readline()** načte jeden řádek a vrátí jej jako řetězec. Pokud se dostane na konec soubor, vrátí prázdný řetězec.

**Metoda read()** se chová stejně jako readline(), může však mít **argument typu int,** který určí **maximální velikost přečteného řetězce.**

**Metoda readlines()** vrátí seznam, jehož jednotlivými položkami jsou všechny řádky souboru.

### Informativně další moduly

**Modul struct** umožňuje číst a zapisovat binární data.

**Moduly cPickle a shelve** umožňují jednoduché a bezpečné možnosti ukládání a přístupu k libovolně složitým datovým strukturám Pythonu bez potřeby definování formátu souboru.

## Příklad: Počítání slov

Podívejme se blíže na tělo funkce *pocetSlov()*. Nejdříve chceme z daného řádku vytvořit seznam slov. V referenční příručce jazyka Python v modulu *string* najdeme funkci nazvanou *split* (rozdělit na části), která z řetězce vytvoří seznam částí oddělených mezerovými znaky (nebo jiným znakem, který můžeme určit). Funkce l*en()* vrací pro seznam počet v něm umístěných prvků. V našem případě to bude počet slov v řetězci, což je přesně to, co potřebujeme.

Konečná podoba zdrojového kódu vypadá takto:

```
import string
def pocetSlov(s):
    seznam = string.split(s)  # funkce split() se nachází v modulu string
    # Poznámka překladatele: Od verze Python 2 lze psát
    # seznam = s.split()
    return len(seznam) # vrátíme počet prvků seznamu

vstup = open("menu.txt", "r")
celkem = 0  # vytvoříme proměnnou a nastavíme jí počáteční hodnotu nula

for radek in vstup.readlines():
    celkem = celkem + pocetSlov(radek) # sečti počty za každý řádek
print "Soubor ma %d slov." % celkem

vstup.close()
```

**Zapamatujte si:**

- Před použitím souborů je musíte otevřít.
- Funkce readlines() jazyka Python přečte všechny řádky souboru najednou, zatímco funkce readline() přečte jen jeden řádek. Může nám to pomoci šetřit pamětí.
- Po použití soubor uzavřete.

**Literatura:**

[1] Rubeš, J.: Nebojte se programovat, Computer Media, Bedihošť 2001
[2] Lutz, M., Ascher, D.: Naučte se Python, Grada, Praha 2003
[3] Beazley, D. M.: Python, Neocortex, Praha 2002
[4] Python Reference Manual
[5] Švec, J.: Létající circus, Python tutoriál, 2003