Úvod do práce se soubory – Python

Michal Žemlička

1 K čemu jsou soubory

Soubory jsou entity umožňující uchovávat logicky související data i po skončení programu. Dají se tak využít ke spolupráci více programů, k uchovávání výsledků a mezivýsledků, ...

Se soubory jsme se už setkali – programy a další dokumenty, co jsme vytvořili, jsou uloženy právě v souborech. Abychom mohli vymoženosti souborů využívat nejen při vlastní práci, ale i našimi programy, povíme si, jak se s nimi (resp. s některými z nich) dá pracovat.

2 Základní operace

Chceme-li se slušně najíst, je třeba nejdříve připravit stůl. Po jídle by měl být stůl zase uklizen. Podobně je to se soubory: než s nimi začneme skutečně něco dělat, tak si je necháme připravit (otevřít soubor); až jsme s prací hotovi, necháme po sobě uklidit (zavřít soubor). To platí pro všechny soubory s výjimkou těch, které má každý program již připravené:

- standardní vstup (`stdin`; zpravidla klávesnice)
- standardní výstup (`stdout`; zpravidla obrazovka)
- chybový kanál (`stderr`; také zpravidla obrazovka)

Tyto soubory jsou otevřené již při zahájení programu a jsou zavírány systémem po skončení programu.

3 Textové soubory

Textové soubory obsahují text podobný tomu, co píšeme jako program. Obsahuje text složený z řádků a ty sestávají z jednotlivých znaků.

Textový soubor je možné přečíst celý, lze z něj přečíst řádek (resp. číst jej po řádcích), lze z něj přečíst znak (resp. číst jej po znacích). Obdobně je možné zapsat celý textový soubor naráz, po řádcích, po znacích i po jiných částech.

3.1 Výstup do souboru

K práci s textovými soubory je možné využívat funkci print, jako pro práci se standardním výstupem – jen je třeba uvést jako parametr file=soubor, kde soubor je proměnná určující soubor. Příklad:

```
a, b = 3, 7
print(a,'+',b,'=',a+b,file=mujsoubor)
```

To však ale narazí na problém, že soubor není znám:

```
NameError: name 'mujsoubor' is not defined
```

Důvod je prostý: zapomněli jsme soubor před použitím otevřít. K tomu slouží funkce `open`(*jméno-souboru*[, *mód*]), kde *jméno-souboru* je jménem osuboru, s nímž chceme pracovat a mód udává, co se souborem chceme dělat:

r – soubor chceme číst,

w – soubor chceme vytvořit či přepsat (případný původní obsah je nahrazen novým),

a – chceme do souboru připisovat na konec (případný původní obsah je doplňován).

Není-li mód uveden, je soubor otevřen pro čtení.

Samotné otevírání souboru můžeme udělat dvěma způsoby. V jednom případě uvádíme, že s daným souborem budeme pracovat v uvedeném bloku (a na konci bloku je soubor automaticky zavřen):

```
with open("vystup.txt","w") as mujsoubor:
    print("piseme s dvernikem",file=mujsoubor)
```

Ve druhém případě musíme soubor otevřít a zavřít explicitně:

```
mujsoubor = open("vystup.txt","a")
print("mame nad psanim kontrolu",file=mujsoubor)
mujsoubor.close
```

3.2 Čtení ze souboru

I při čtení ze souboru potřebujme soubor otevřít a zavřít. Funkci open už známe, oba způsoby, jak ji využít, také.

Pro ukázky čtení ze souboru budeme předpokládat existenci souboru „vstup.txt“ v aktuálním adresáři s obsahem:

```
prvni radek
druhy radek
treti radek
```

Jak jej ale číst?

3.2.1 Načtení celého souboru naráz

K načtení celého souboru slouží metoda `read()`:

```
with open("vstup.txt") as mujsoubor:
    obsah = mujsoubor.read()
```

Pokud si načtený obsah necháme i vytisknout,

```
print(obsah)
```

měli bychom dostat

```
prvni radek
druhy radek
treti radek
```

3.2.2 Čtení vstupního souboru po řádcích

K načtení celého řádku ze vstupního souboru slouží metoda `readline()`. Provedením tohoto kódu

```
with open("vstup.txt") as mujsoubor:
    prvni = mujsoubor.readline()
    druhy = mujsoubor.readline()

print(druhy)
```

bychom měli dostat výstup

```
druhy radek
```

Často však nestačí jen načtení daného počtu řádek. Potřebujeme umět načíst celý soubor:

```
with open("vstup.txt") as mujsoubor:
    for radek in mujsoubor:
        print('>',radek.strip(),'<')
```

Poznámka 1: Řádek se čte včetně konce řádku, proto jsme použili metodu strip, aby nás nerušily případné bílé znaky na začátku a konci řádku.

V tomto případě, stejně jako u dalšího příkladu, dostáváme takovýto výstup:

```
> prvni radek <
> druhy radek <
> treti radek <
```

Poznámka 2: Konec řádku na konci řetězce také naznačuje, že už není co číst – že už jsme na konci souboru.

Níže uvedený kód naznačuje, jak číst celý soubor po řádcích s využitím detekce konce souboru pomocí výše uvedeného způsobu:

```
with open("vstup.txt") as f:
    while True:
        radek = f.readline()
        if radek == '':
            break
        print('>',radek.strip(),'<')
```

Je to o poznání nepřehlednější. Poskytuje to však lepší kontrolu nad tím, kdy případnou smyčku ukončit. To může být užitečné, zpracovává-li program více vstupních souborů současně.

3.2.3 Čtení vstupního souboru po znacích

Někdy můžeme chtít číst vstupní soubor po znacích. V takovém případě využijeme již známou funkci metodu `read()`, tentokrát ale využijeme možnost zadat počet znaků, které chceme přečíst:

```
with open("vstup.txt") as f:
    c = f.read(1)
    print(c)
```