Textové soubory

z pohledu Pythonu je textový soubor postupnost řádků. Každý řádek je zase posloupnost znaků ukončena znakem *"\n",* pro nový řádek.

Pokud chceme v Pythonu se souborem pracovat, musíme ho nejprve otevřít - navázat spojení.
Po té můžeme soubor číst, nebo do něj zapisovat.
Na konci naší práce, musíme soubor uzavřít - ukončit spojení.

Otevření souboru

Soubor otvíráme pomocí funkce *Open()*, kde v závorce uvedeme název souboru, případně i s cestou, není-li uložen v adresáři Pythonu, kde je i script, případně relativní pozici k umístění scriptu. Tato funkce vrátí referenci na souborový objekt. Říká se tomu datový proud, nebo streem.

Pro textové soubory je vhodné pojmenovávat proměnou tak, aby odpovídala vztahu k souboru:

```
subor =open('pribeh.txt', 'r')
kniha =open('c:/dokumenty/python.txt', 'r')
subor_s_cislami =open('../texty/cisla.txt', 'r')
```

Do souborové proměnné se přiřadí spojení s uvedeným souborem.

Čtení souboru

Nejčastěji se informace ze souboru čtou po celých řádcích.
Základní způsob je:

```
riadok = subor.readline()
```

Funkce *readline()* je metoda souborové proměnné a tak za jménem souboru musí být tečka. Funkce vrátí znakový řetězec, obsahující text řádku a koncový znak pro nový řádek *"\n".* Funkce si pamatuje, kde s čtením skončila a každé další volání *readline()* čte další řádek.

Když funkce dojde až na konec souboru, vrátí znak pro prázdný řádek:"

Přečtený řádek nemusíme přiřazovat do proměnné, ale můžeme ho zpracovat i jinak, např:

```
subor =open('subor.txt', 'r')
print(subor.readline())
print('dlzka =', len(subor.readline()))
print(subor.readline()[::-1])
```

V tomto případě se nejprve vypíše obsah prvního řádku, potom délka druhého řádku a na závěr se vypíše třetí řádek, ale otočený.

Po skončení čtení je potřeba soubor uzavřít (a s ním i spojení), voláním metody *close(),* tím se uvolní systémové zdroje (*resources*), které byli potřebné pro otevřený soubor*:*

```
subor.close()
```

Použití for cyklu pro čtení ze souboru

Pokad u souboru dopředu nevíme kolik má řádků, použijeme zápis s odkazem na otevřený soubor:

```
t =open('subor.txt', 'r')
for riadok in t:
        print(repr(riadok))
t.close()
```

Python v tomto případě považuje otevřený soubor za posloupnost řádků (i s koncovým "\n").
Pokud používáme takovýto zápis *for cyklu*, pak už nevoláme *readline()*, protože to by přečetlo další řádek, který by se tak ve výpisu přeskočil.

Přečtení celého souboru do jednoho řetězce:

Namísto toho, abychom procházeli celý soubor pomocí připočítávací šablony cyklu, můžeme použít metodu read(), která to udělá za nás:

```
t =open('subor.txt', 'r')
cely_subor = t.read()
t.close()
print(cely_subor, end='')
```

Tato metoda je přirozeně omezená kapacitou našeho počítače a jeho paměti pro Python.

V případě, že chceme přečíst ze souboru jen určitou délku textu (počet znaků), uvedeme tuto hodnotu do závorek metody read()´:

```
>>>t =open('subor.txt')
>>>print('prvý znak =', repr(t.read(1)))
>>>print('ďalších 10 znakov =', repr(t.read(10)))
>>>print('zvyšok súboru =', repr(t.read()))
```

Čeština v souboru

Python pracuje se sadou Unicode, nicméně soubory mohou být uloženy v různém kódování ('cp1250', 'iso88591', 'utf-8', …) a tak, pokud chceme aby se správně zobrazili (např. s diakritikou) musíme při otvírání souboru uvést toto kódování do závorek jako parametr funkce *open():*

```
subor =open(meno_suboru, 'r', encoding='utf-8')
```

Zjištění konce souboru

Metoda *readline()* vrátí na konci souboru znak pro prázdný znak (dva apostrofy), a podle nich je tedy možné rozpoznat, zda se jedná o konec souboru. V případě prázdných řádků totiž vrací samotný znak pro nový řádek *"\n".*

```
t =open('subor.txt', 'r')
riadok = t.readline()
while riadok !='':
        print(riadok, end='')
        riadok = t.readline()
t.close()
```

Namísto *while riadok != ''*, můžeme použít zkrácenou podobu: *while riadok*

Šablona čtení (zjišťující konec souboru) s nekonečným *whilecyklem*:

```
t =open('subor.txt', 'r')
whileTrue:
        riadok = t.readline()
                if riadok =='':
                break
        print(riadok, end='')
t.close()
```

Pokud pro zobrazení řádku použijeme funkci *print(),* musíme počítat s tím, že funkce *print()* přidává na konec řádku znak pro nový řádek *"\n"*, takže rázem jsou zde dva a je potřeba jeden odstranit. Buď tak, že v funkci *print()* uvedeme *end=''*, nebo tak, že při čtení řádku, poslední znak umažem :
print(riadok[:-1])

Toto však nebude fungovat u posledního řádku, kde tento znak být nemusí.

Funkce repr()

vrací ve výpisu (například funkcí *print()*) úplnou strukturu textu, tak jak ji čte Python, i s apostrofmi a speciálními znaky. Vhodná je pro náhled při ladění a pro kontrolu obsahu.

```
>>>a ='ahoj   \naj "apostrof" \' v texte  \n'
>>>print(a)
   ahoj
   aj "apostrof" ' v texte

>>>print(repr(a))
   'ahoj   \naj "apostrof" \' v texte  \n'
```

Pokud bychom chtěli při čtení odstranit z textu mezery a speciální znaky, můžeme použít textovou metodu *strip():*

print(repr(riadok.strip()))

Pokud bychom chtěli odstranit mezerové znaky jen z konce řetězce (z prava), použijeme metodu *rstrip()*

Zápis do souboru

Při zápisu do souboru Python tento soubor vytváří, nebo přepisuje původní. A stejně jako u čtení začíná se příkazem *open():*

subor =open('meno_súboru', 'w')

Tím se na soubor vytvoří spojení a soubor samotný se uloží na umístění, kde máme i skript. Pokud chceme soubor uložit někam jinam, je potřeba uvést i cestu. Pokud soubor již existoval, tímto příkazem se vymaže - vyprázdní, takže je potřeba být opatrný, aby si člověk nesmazal důležitá data.

Do souboru je pak možno zapisovat pomocí metody *write(),* nebo funkce *print().*

Zápis do souboru pomocí metody write()

subor.write(reťazec)

Tato metoda zapíše zadaný řetězec na momentální konec souboru a pokud chceme, aby se v souboru objevili i znaky pro konce řádku, musíme je uvést v zadávaném řetězci. Jinak se bude vše vpisovat jen do jednoho řádku.

subor.write('Python\nje najlepsi\n')

Zápis do souboru pomocí funkce print()

Zápis probýhá stejně, jako když používáme funkci *print()* pro standardní zobrazení výsledku v "shellu", ale přidáme parametr kterým přesměrujeme výstup do našeho souboru.

print('proměná či text', file=subor)

Pokud bychom chtěli, aby se výsledek vypsal do jednoho řádku, musíme uvést do parametru funkce i end='':

print('proměná či text', end='', file=subor)

Přidávání řádků do souboru

Jsou tři způsoby otevírání souborů:

1) *t = open('subor.txt', 'r')* - soubor se otevře jen na čtení (pokud ještě neexistoval, program spadne)

2) *t = open('subor.txt', 'w')* - soubor se otevře jen na zápis (pokud ještě neexistoval, pak se vytvoří a pokud už existoval, vymaže se obsah)

3)) *t = open('subor.txt', 'a')* - soubor se otevře pro zápis, ale nezruší se jeho původní obsah, namísto toho se budou řádky připisovat na konec souboru

r = read, w = write, a = append

Kopírování souboru

Pokud potřebujeme obsah jednoho souboru překopírovat do druhého (při tom je možné něco i dělat s každým řádkem) můžeme použít 2 souborové proměné:

```
odkial =open('subor.txt', 'r')
kam =open('subor2.txt', 'w')
for riadok in odkial:
        riadok = riadok.strip()
        if riadok !='':
                kam.write(riadok +'\n')
odkial.close()
kam.close()
```

Program postupně přečte všechny řádky, vyhodí mezery ze začátku a konce řádku a pokud je takový řádek neprázdný, zapíše ho do nového souboru (protože *strip()* maže i znak pro nový řádek, je při jeho použití třeba ho znovu na konec řádku v novém souboru dopsat).

V případě, že chceme otevřít a provést zápis do jednoho a toho samého souboru, použijeme takovýto zápis:

```
t =open('subor.txt', 'r')
cely =''
for riadok in t:
        riadok = riadok.strip()
        if riadok !='':
                cely += riadok +'\n'
t.close()

t =open('subor2.txt', 'w')
t.write(cely)
t.close()
```

A pokud bychom při čtení nepotřebovali nic měnit, můžeme použít metodu *read()*:

```
t =open('subor.txt', 'r')
cely = t.read()
t.close()

t =open('subor2.txt', 'w')
t.write(cely)
t.close()
```

Zavření souboru po zápisu

Stejně jako při čtení i při zápisu je potřeba po provedení úkonu soubor uzavřít a zrušit spojení pomocí metody *close().* Bez ukončení nemáme jistotu, zda Python stačil zapsat všechny řetězce volání *write()* na disk. Navíc i operační systém by mohl mít potíže takovýto soubor následně otevřít.

Konstrukce with *(v mé verzi pythonu3.8 nefunguje)*

Na práci se soubormi můžeme použít speciální programovou konstrukci *with*, pomocí které si Python domyslí, že jsme už se souborem skončili pracovat a automaticky ho uzavře.
(Samotný výraz *with* má i jiné použití než jen se soubormy.)

Všeobecný tvar příkazu je:

```
withopen(...) as premenna:
        prikaz
        prikaz
...
```

Touto příkazovou konstrukcí se otevře požadovaný soubor a referenca na soubor se přiřadí do proměné uvedené za *„as“*. Po té se vykonají všechny příkazy v bloku a po jejich skončení se soubor automaticky uzavře.

Tato metoda je vhodná například i se soubory ve funkcích, kdy po príkazu *return*, pokud je použit uvnitř bloku *with*, se soubor automaticky uzavře.

Pár příkladů zápisů *with*:

Přečti a vypiš obsah celého souboru:

```
withopen('subor.txt', 'r') as subor:
print(subor.read())
```

Vytvoř soubor s řádkami:

```
withopen('subor.txt', 'w') as file:
print('prvy\ndruhy\ntreti\n', file=file)
```

Vytvoř soubor za pomoci for cyklu:

```
import random
withopen('cisla.txt', 'w') as file:
        for i inrange(100):
                file.write(f'{random.randint(0, 1000)} ')
```

Automatické zavírání souboru

Pokud do souboru zapisujeme pouze jednou a hned ho zavíráme, není potřeba pro něj vytvářet proměnou, ale přímo při otevření uděláme jen zápis - v tu chvíli se soubor automaticky i uzavře. Využijeme toho, že pro soubor nevytváříme proměnou, ale funkci *open()* použijeme přímo např. při volání nějaké metody, např.:

open('subor2.txt', 'w').write('first line\nsecond line\nend of file\n')

Tímto jediným příkazem jsme vytvořili soubor, zapsali jsme do něj a automaticky se uzavřel.

Podobně to můžeme zapsat i pomocí funkce *print():*

print('first line\nsecond line\nend of file', file=open('subor3.txt', 'w'))

Dalo by se takto i zapsat kopírování souboru:

open('subor2.txt', 'w').write(open('subor.txt', 'r').read())

Ale zde je lepší použít čitelnější variantu:

```
withopen('subor.txt', 'r') as r:
withopen('subor2.txt', 'w') as w:
        w.write(r.read())
```

Nebo:

```
withopen('subor.txt', 'r') as r, open('subor2.txt', 'w') as w:
        w.write(r.read())
```

Příklad s grafikou

Máme soubor v kterém máme souřadnice bodů:

```
100 100
150 200
200 150
150 150
```

na které chceme nakreslit do grafického plochy malé kroužky, program by pak vypadal následovně:

```
import tkinter

canvas = tkinter.Canvas()
canvas.pack()

withopen('body.txt') as subor:
for riadok in subor:
    i = riadok.find(' ')
    x, y =int(riadok[:i]), int(riadok[i:])
    canvas.create_oval(x-10, y-10, x+10, y+10)

tkinter.mainloop()
```