IB113 Úvod do programování a algoritmizace

Cvičení 12

Jaromír Plhák

Práce se soubory - Motivace

- Doteď jsme pracovali jen interaktivně
  - Vstup z klávesnice, výstup na obrazovku
- Co když chceme
  - Číst vstup ze souboru a/nebo
  - Zapisovat výstup do souboru?
- Python obsahuje mnoho vestavěných funkcí pro I/O (= input/output; vstup/výstup):
  - https://docs.python.org/3.5/tutorial/inputoutput.html
- Mějme soubor input.txt s tímto obsahem:

I am a little testfile.

One day, I will be awesome.

Vstup (čtení) ze souboru po řádcích

```
with open("file", "mode") as variable:
    # Read (or write to) the file
    # using the file handle object
```

- Příklad

```
with open("input.txt", "r") as textfile:
    for line in textfile:
        print(line, end="")
```

- Výstup našeho skriptu

```
I am a little testfile.
One day, I will be awesome.
```

Vstup (čtení) ze souboru – znak \n

- Znak \n představuje konec řádku
- Mějme stejný vstupní soubor a skript

```
lst = []
with open("input.txt", "r") as textfile:
    for line in textfile:
        lst.append(line)
print(lst)
```

- Výstup

```
['I am a little testfile.\n', 'One day, I
    will be awesome.']
```

Vstup (čtení) ze souboru – znak \n

- Ekvivalent: metoda readlines()
- Pokud se chceme zbavit znaku \n při čtení souboru
  - Metoda rstrip()
  - Např. lst.append(line.rstrip())

Výstup (zápis) do souboru

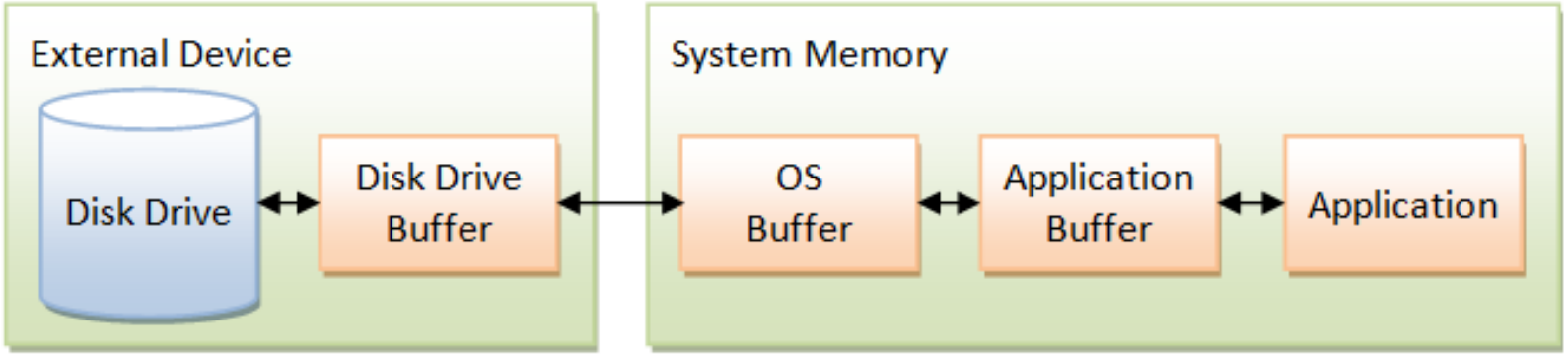
- Režim w (write): Předešlý obsah se přepíše novým!
- Režim a (append): Nový obsah se zapíše na konec souboru

```
with open("output.txt", "mode") as f:
    f.write("Data to be written.")
```

- Funkce write(s) zapíše proměnnou typu řetězec do souboru (tak jak je, nepřidává znak nového řádku)

Nutnost zavírat soubory

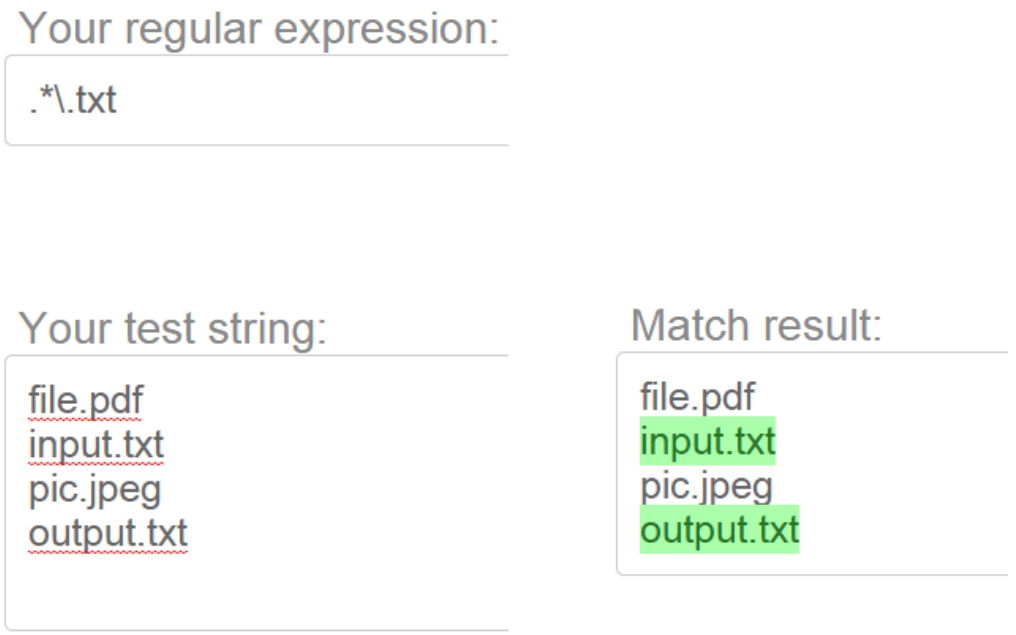
- Pokud nepoužíváte příkaz with, musíte zavírat soubory
- Při zápisu se data zapisují nejprve do vyrovnávací paměti, až potom do souboru na disku (tzv. buffering)
- Může se stát, že data zůstanou v bufferu, ale nestihnou se přenést do souboru
- Zavření souboru je dobrý způsob, jak vyprázdnit (spláchnout do souboru) obsah bufferu



Zdroj: https://www3.ntu.edu.sg/home/ehchua/programming/java/J5b_IO_advanced.html

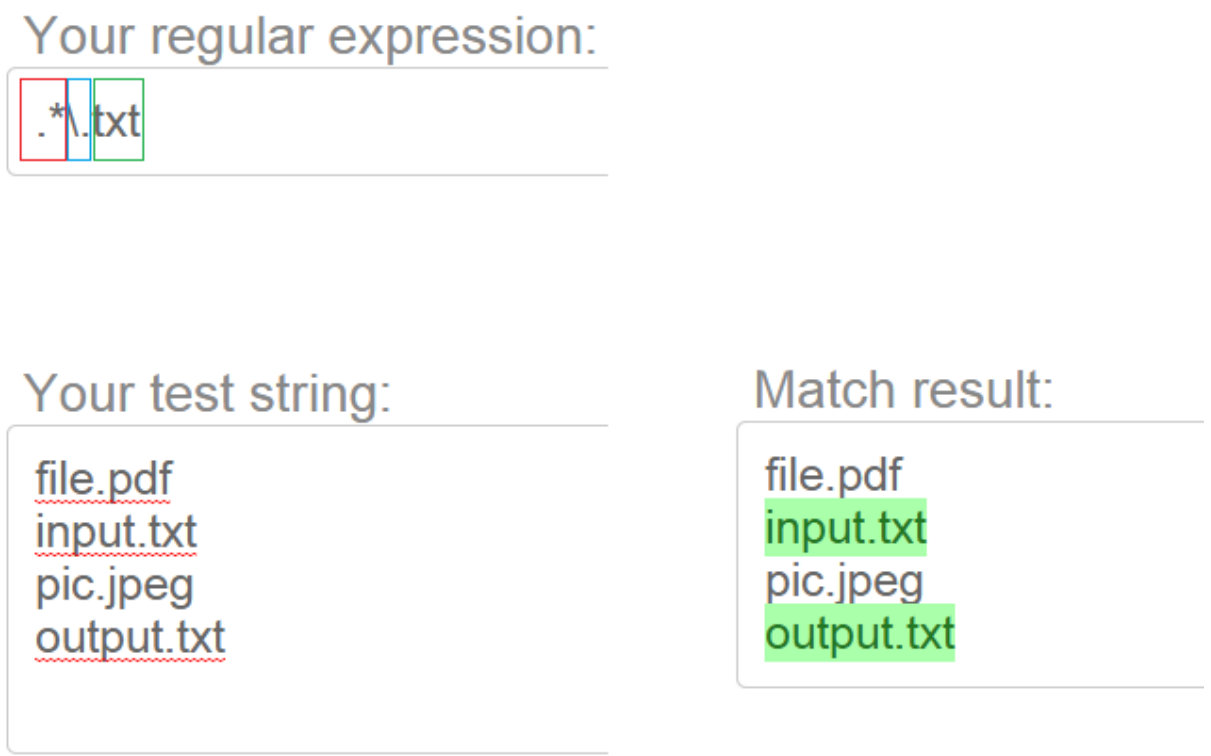
Regulární výraz (regex, regular expression)

- Řetězec znaků, který popisuje vzor v textu
  - Nezávislý na programovacím jazyce!
- Podobné jako *wildcard* v správci souborů – např. *.txt (regex ekvivalent je .*\.txt), ale mnohem mocnější



Regulární výraz – analýza

.	Libovolný znak
*	0 až *n*-krát (greedy; nejdelší možný řetězec)
\.	Tečka (escaping)
txt	Řetězec txt



Regulární výraz – použití

- **Hledání vzorů v textu**
- Náhrada řetězců v textu
- Normalizace textu (odstranění bílých znaků)
- …
- Elegantní a krátké řešení pro mnoho problémů
  - http://www.hovnokod.cz/422
- Nevýhody:
  - Nenaučíte se je za půlhodinu
  - Komplikovanost při nesprávném použití:
    - http://ex-parrot.com/~pdw/Mail-RFC822-Address.html

Tahák z http://pythex.org

Special characters

`\`	escape special characters
`.`	matches any character
`^`	matches beginning of string
`$`	matches end of string
`[5b-d]`	matches any chars '5', 'b', 'c' or 'd'
`[^a-c6]`	matches any char except 'a', 'b', 'c' or '6'
`R|S`	matches either regex `R` or regex `S`
`()`	creates a capture group and indicates precedence

Quantifiers

`*`	0 or more (append `?` for non-greedy)
`+`	1 or more (append `?` for non-greedy)
`?`	0 or 1 (append `?` for non-greedy)
`{m}`	exactly `m` m occurrences
`{m, n}`	from `m` to `n`. `m` defaults to 0, `n` to infinity
`{m, n}?`	from `m` to `n`, as few as possible

Special sequences

`\A`	start of string
`\b`	matches empty string at word boundary (between `\w` and `\W`)
`\B`	matches empty string not at word boundary
`\d`	digit
`\D`	non-digit
`\s`	whitespace: `[ \t\n\r\f\v]`
`\S`	non-whitespace
`\w`	alphanumeric: `[0-9a-zA-Z_]`
`\W`	non-alphanumeric
`\Z`	end of string
`\g<id>`	matches a previously defined group

Special sequences

`(?iLmsux)`	matches empty string, sets re.X flags
`(?:...)`	non-capturing version of regular parentheses
`(?P...)`	matches whatever matched previously named group
`(?P=)`	digit
`(?#...)`	a comment; ignored
`(?=...)`	lookahead assertion: matches without consuming
`(?!...)`	negative lookahead assertion
`(?<=...)`	lookbehind assertion: matches if preceded
`(?<!...)`	negative lookbehind assertion
`(?(id)yes|no)`	match 'yes' if group 'id' matched, else 'no'

Regulární výrazy v Pythonu

- Modul re
- Funkce:
  - match() – hledá shodu od začátku řetězce
  - search() – hledá shodu kdekoliv v reakci (pomalejší)
- Regulární výraz: r'výraz'
  - Python neinterpretuje žádné jeho speciální znaky, což je dobře, jelikož nemusíme nic escapovat

```
>>> import re
>>> x = re.match("as", "fast")  # Nothing
>>> y = re.search("as", "fast") # Match
```

Regulární výrazy v Pythonu

- Návratová hodnota: MatchObject alebo None

```
>>> print(x)
None
>>> print(y)
<_sre.SRE_Match object at 0x7fd4bf238238>
```

```
>>> print(x)
None
>>> print(y.group(0))
as
```

Úkoly

- Regulární výrazy
  - https://is.muni.cz/auth/el/1433/podzim2017/IB113/um/skupiny_01_a_02/cv12.py
  - https://is.muni.cz/auth/el/1433/podzim2017/IB113/um/skupiny_01_a_02/slovnik.txt
- Analýza textu
  - https://is.muni.cz/auth/el/1433/podzim2017/IB113/um/skupiny_01_a_02/sherlock.txt
  - Napište funkce, které analyzují text a vypíší:
    - Top 10 nejčastějších slov
    - Top 10 nejčastějších slov délky alespoň 3 znaky
    - Top *n* nejčastějších slov délky alespoň *k* znaků
    - Průměrnou délku slova / věty v textu
    - …

Zadání domácí úlohy 6 – Soubory a slovník

- V ISu v odevzdávárně
  - https://is.muni.cz/auth/el/1433/podzim2017/IB113/ode/72006396/72006443/hw06_zadani.py (skupina 1)
  - https://is.muni.cz/auth/el/1433/podzim2017/IB113/ode/72006400/72006457/hw06_zadani.py (skupina 2)
- Veškeré informace v souboru
- Odevzdejte pouze tento soubor do odevzdávárny
  - https://is.muni.cz/auth/el/1433/podzim2017/IB113/ode/72006396/72006434/ (skupina 1)
  - https://is.muni.cz/auth/el/1433/podzim2017/IB113/ode/72006400/72006455/ (skupina 2)
- Do **čtvrtka** 21. 12. 2017, 23:59