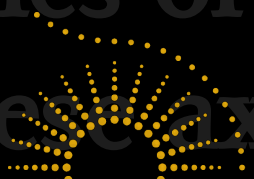
Práce s textem
(3. přednáška)



FF UK
Department of Logic

Jonathan L. Verner

Práce s textem

FF UK
Department of Logic

- Sazba (Word, OpenOffice, TeX, InDesign . . .)
- Vyhledávání (Ctrl-F, Google, . . .)
- Data mining
- Automatický překlad
- Kryptografie
- Čtení textu (převod do zvuku)
- Spellchecking
- Analýza textů (identifikace autora, . . .)

Textové literály

- Zadávají se pomocí jednoduchých nebo dvojitých uvozovek.
- Pokud potřebujeme použít uvozovky, dáme před ně \

```
>>> s1="Ahoj ''Honzo''"!
>>> s2='Jak se ''mas''?'
>>> print s1, s2
Ahoj 'Honzo'! Jak se 'mas'?
>>> type(s1)
<class 'str'>
>>> type(s2)
<class 'str'>
```

Speciální znaky

FF UK
Department of Logic

'Escape sequence'	Význam
`\\`	Zpětné lomítko (\)
`\n`	Line feed (UNIX: nová řádka)
`\r`	Carriage return (MAC: nová řádka)
`\r\n`	CRLF (WIN: nová řádka)
`\t`	Tabulátor
`\'`	Jednoduché uvozovky (')
`\"`	Dvojité uvozovky (")

Pokud před celý string dáme `r` (raw), speciální význam má pouze `\'`, `\"`, které se nahradí `\'` resp. `\"`. Ani raw string nesmí končit lichým počtem `\`.

```
>>> print('\\\\\text')
\\^^Iext
>>> print(r\\\\\text)
\\\\\text
>>> print(r\)
SyntaxError: EOL while scanning string literal
```

Formální definice

FF UK
Department of Logic

```
stringliteral   ::=  [stringprefix](shortstring | longstring)
stringprefix    ::=  "r"
shortstring     ::=  "'" shortstringitem* "'" | '"' shortstringitem* '"'
longstring      ::=  "'''" longstringitem* "'''"
                     | '"""' longstringitem* '"""'
shortstringitem ::=  shortstringchar | escapeseq
longstringitem  ::=  longstringchar | escapeseq
shortstringchar ::=  <any source character except "\" or newline or the quote>
longstringchar  ::=  <any source character except "\">
escapeseq       ::=  "\" <any ASCII character>
```

Práce se stringy indexace

FF UK
Department of Logic

- Řetězec (string) je posloupnost znaků.
- Přes posloupnost lze iterovat:

```
>>> str1 = "Ahoj"
>>> for i in str1:
        print (i,end=" ")

A h o j
>>>
```

- Lze se dotazovat na jednotlivé znaky (tzv. indexace)
- Číslování začíná od 0, záporné indexy indexují od konce!

```
 >>> str1[0]
'A'
>>> str1[-1]
'j'
```

Krájení stringů Slicing

FF UK
Department of Logic

`<string>[start:end]`

- `start`, `end` jsou celočíselné výrazy
- vrátí podřetězec začínajíc `start`-tým znakem a končící `end-1`-ním znakem
- start,end mohou být záporné, pak se počítá odzadu

kladný index	0	1	2	3
`str1`	A	h	o	j
záporný index	-4	-3	-2	-1

```
>>> print(str1[1:2])
h
>>> print(str1[1:3])
ho
>>> print(str1[1:-1])
ho
```

Krájení stringů II.

FF UK
Department of Logic

- `start,end` lze vynechat pak jsou nahrazeny 0, resp. indexem posledního znaku

```
>>> print(str1[:-2])
Ah
>>> print(str1[1:])
hoj
>>> print(str1[:])
Ahoj
```

- indexace za délku stringu je přípustná

```
>>> print(str1[0:23000])
Ahoj
>>> print(str1[23000:-38000])


>>>
```

Další operace s řetězci

FF UK
Department of Logic

Syntax	Význam
`<str1>[<num>]`	Indexace (num-tý znak řetězce str1)
`<str1>[<num>:<num>]`	Krájení (Slicing)
`for <var> in <str1>`	Iterace přes řetězec
`<str1> + <str2>`	Konkatenace (str1str2)
`<str1> * <num>`	Opakování (str1str1...str1)
`len(<str1>)`	Délka řetězce
`str.lower(<str1>)`	Velká na malá písmena
`str.upper(<str1>)`	Malá na velká písmena

```
>>> str2=str1+' Johne!'
>>> str2
'Ahoj Johne!'
>>> len(str2)
11
>>> str3=str2*2
>>> str3
'Ahoj Johne!Ahoj Johne!'
>>>
```

Vyhledávací tabulka

FF UK
Department of Logic

Napište program, který převede číslo měsíce na jeho anglickou trojpísmennou zkratku.

- Jednoduché řešení při použití `if ... then ... else`
- Lze vyřešit i bez nich, svým způsobem elegantněji.

Základní idea

- Uložíme si zkratky do jednoho dlouhého stringu.
- Protože každá zkratka má tři písmena, víme přesně na jaké pozici je a můžeme si ten string rozkrájet

Vyhledávací tabulka

FF UK
Department of Logic

```
# mesic.py
#   Vypíše anglickou zkratku měsíce
# autor: Student Logiky

def main():
    # Vyhledávací tabulka zkratek
    mesice='JanFebMarAprMayJunJulAugSepOctNovDec'

    n = int(input('Zadej číslo měsíce (1-12): '))

    # Misto v retezci, kde se nachazi zkratka
    pos = (n-1)*3

    # Zjisti si zkratku a vypis ji
    zkr = mesice[pos:pos+3]

    print("Zkratka měsíce je", zkr + '.')

main()
```

Vyhledávací tabulka

FF UK
Department of Logic

Funguje jen pro stejně dlouhé zkratky!

- místo řetězce, můžeme použít seznam
- seznamy se chovají jako řetězce, místo znaků jsou libovolné objekty
- stejné operace jako na stringy fungují i na seznamy

```
>>> [1,2] + [3,4]
[1, 2, 3, 4]
>>> [1,"ahoj"]*3
[1, 'ahoj', 1, 'ahoj', 1, 'ahoj']
>>> len([1,"ahoj",[1,2]]*3)
9
>>> [1,"ahoj",[1,2,3]]*2
[1, 'ahoj', [1, 2, 3], 1, 'ahoj', [1, 2, 3]]
>>>
```

Řetězce x Seznamy

FF UK
Department of Logic

Seznamy lze měnit

```
>>> s = [1,2] + [3,4]
>>> s[1]='ahoj'
>>> print(s)
[1, 'ahoj', 3, 4]
```

Řetězce měnit nelze!

```
>>> str1="Ahoj"
>>> str1[0]='B'
Traceback (most recent call last):
  File "<pyshell#96>", line 1, in <module>
    str1[0]='B'
TypeError: 'str' object does not support item assignment
```

Reprezentace textu v počítači

FF UK
Department of Logic

Pro připomenutí

- vnitřně se uchovávají pouze posloupnosti nul a jedniček.
- je třeba se dohodnout, jak se budou interpretovat

Interpretace, t.j. Kódování řetězců

- původně žádná dohoda nebyla - nešlo se domluvit
- toto řešil standard ASCII (American Standard for Information Interchange)
- jde v podstatě o tabulku, která číslům od 0-127 přiřazuje znaky
- protože se lépe pracuje s 8 bity vzniklo Extended ASCII, které obsahuje dalších 128 znaků (t.j. 0-255)
- problémy s nabodeníčkama

Co s háčky a čárky?

ISO 8859-1 (Western Europe), ISO 8859-2 (Central Europe), ISO 8859-3 (South European (Turkish), Maltese plus Esperanto), ISO 8859-4 (Lithuania, Estonia, Latvia and Lapp), ISO 8859-5, ISO 8859-6, ISO 8859-7, ISO 8859-8, ISO 8859-9, ISO 8859-10 ISO 8859-11 ISO 8859-12 ISO 8859-13 ISO 8859-14 ISO 8859-15 ISO 8859-16 CP437, CP737, CP850, CP852, CP855, CP857, CP858, CP860, CP861, CP862, CP863, CP865, CP866, CP869 Windows-1250 Windows-1251, Windows-1252 Windows-1253 Windows-1254 Windows-1255 Windows-1256 Windows-1257 Windows-1258 ASCII, ISCII, TSCII, VISCII, ISO 646, EBCDIC, CP37, CP930, CP1047, Mac OS Roman KOI8-R, KOI8-U, KOI7, MIK, JIS X 0208, Shift JIS, EUC-JP, ISO-2022-JP, JIS X 0213, Shift JIS-2004, EUC-JIS-2004, ISO-2022-JP-2004, Chinese Guobiao, GB 2312, GBK, GB 18030, Taiwan Big5, Hong Kong HKSCS, Korean, KS X 1001, EUC-KR, ISO-2022-KR, ANSEL, ISO/IEC 6937, . . .

Unicode

- 1988, Joe Becker (Xerox, Apple)
- původně zaměřen na aktuálně používané jazyky
- časem rozšířen tak, aby zachytil např. Egyptské hieroglyfy
- stále ještě není plně dokončen
- spousta různých způsobů jak kódovat
- nejčastější nejspíše UTF-8

UTF-8

FF UK
Department of Logic

- každý unicode znak je reprezentován 1–4 byty
- počet bytů je dán počtem 1 na začátku prvního bytu
- znaky ASCII (0–127) jsou kódovány pouze jedním bytem, ASCII text je validním UTF-8 textem
- druhý a každý další byte začíná 10
- všechny nespotřebované bity se spojí dohromady a dají pořadí znaku v unicode tabulce (tzv. code point)
- ne všechny sekvence bitů jsou validním UTF-8

Unicode & Python

FF UK
Department of Logic

- pokud chcete psát háčky a čárky v zdrojovém kódu, je třeba deklarovat kódování

```
# coding: utf-8
a='HáčkyAČárky'
```

https://docs.python.org/3/howto/unicode.html

Funkce

FF UK
Department of Logic

funkce	význam
`str.capitalize()`	vrátí řetězec, první písmeno převede na velké
`str.lower()/str.upper()`	převede řetězec na malá/velká písmena
`str.split(separator)`	rozdělí řetězec `str` na položky oddělené pomocí znaku `separator`
`str.join(list)`	vrátí řetězec sestávající z položek v seznamu oddělených znakem separator
`str.replace(co,cim)`	nahradí v řetězci `str` každý výskyt řetězce `co` řetězcem `cim`
`str.find(co)`	vrátí pozici prvního výskytu řetězce `co` v řetězci `str` nebo -1 pokud se tam nevyskytuje

Funkce (pokračování)

FF UK
Department of Logic

funkce	význam
`str.center(sirka[,vypln])`	doplní řetězec `str` z obou stran mezerami (nebo znakem `vypln`, pokud je zadan) tak, aby vrácený řetězec měl délku alespoň `sirka`

```
>>> from string import split,join
>>> split('cervena,zelena,modra,oranzova',',')
['cervena', 'zelena', 'modra', 'oranzova']
>>> join(['Ahoj','jak','se','mas?'],' ')
'Ahoj jak se mas?'
```

Konverze datumů

Napište program, který převede datum z formátu `dd/mm/rrrr` (např. 1. 6. 1938) do formátu `Mmm. dd, rrrr` (např. Jun. 6, 1938).

```
Načti datum ve formátu dd/mm/rrrr (date_string).
Rozděl date_string do tří stringů (mstr, dstr, rstr)
  odpovídajících měsíci, dnu a roku.
Převeď mstr string měsíce na číslo mnum.
Použij mnum k vyhledání zkratky v tabulce.
Vytvoř string ve výsledném formátu a vypiš ho.
```

Konverze datumů II: Implementace

Práce s řetězci

FF UK
Department of Logic

```
# coding: utf-8
# dateconv.py
#   Program na konverzi formatovanych datumu
# autor: Student Logiky

from string import split

def main():
    abrv=['Jan','Feb','Mar','Apr','May','Jun'
          'Jul','Aug','Sep','Oct','Nov','Dec']
    date_string = input('Zadej datum (dd/mm/rrrr):')
    dstr, mstr, rstr = date_string.split('/')
    mnum = int(mstr)-1
    outstr = abrv[mnum]+'. '+dstr+', '+rstr
    print outstr

main()
```

Práce se soubory

FF UK
Department of Logic

1. Přiřadit soubor na disku k nějaké proměnné (`open`).
2. Práce se souborem (`read`, `readline`, `write`, `...`).
3. Ukončení práce se souborem (`close`)

```
f = open('soubor.txt')
for line in f:
  print(line)
f.close()
```

Funkce open


```
f = open( file_name, mode )
```

- `file_name` je název souboru, adresáře oddělené dopředným lomítkem (’/’)
- `mode` určuje způsob, jakým budeme se souborem pracovat

Mode	Význam
`w`	Do souboru se bude zapisovat. Pokud existuje, je nejprve smazán!
`a`	Bude se zapisovat na konec souboru.
`r`	Ze souboru se bude číst. Soubor musí existovat.
`r+/w+/a+`	Ze souboru se bude číst i do něj zapisovat
`rb/wb/ab`	Soubor se bude interpretovat jako binární

Binární versus textové soubory

FF UK
Department of Logic

- Binární soubory posloupnost bytů (1 byt = 8 bitů, t.j. 8 nul a jedniček)
- Textové soubory posloupnost znaků (default)
  - nutno převést nuly a jedničky na znaky (parametr `encoding`, default závisí na systému)
  - nutno rozdělit na řádky (parametr `newline`, default závisí na systému: UNIX: `\n`, MAC: `\r`, WIN: `\r\n`, PYTHON: `\n`).

Textový soubor

```
Ahoj
Světe

Smysl života 42
```

Binární soubor

```
Ahoj\nSv\xc4\x9bte\n\nSmysl \xc5\xbeivota 42\n\n
```

read, write, ...

FF UK
Department of Logic

funkce	Význam
`read([size])`	Vrátí celý soubor (nebo `size` bytů) jako string
`readline()`	Vrátí další řádku (včetně znaku '`\n`'!)
`readlines()`	Vrátí celý soubor jako seznam řádků.
`write(str)`	Zapíše str do souboru.
`truncate()`	Vyprázdní soubor.
`tell()`	Vrátí aktuální pozici v souboru
`seek(pos)`	Nastaví aktuální pozici na pos
`close()`	Uzavře soubor.
`flush()`	Zapíše (alespoň se tak tváří) vešekrá data na disk.

Pozor: při čtení a zapisování souboru v (defaultním) textovém módu Python automaticky konvertuje znaky konce řádků ze systémové konvence na konvenci pythonovskou (`'\n'`).