C2184 Úvod do programování v Pythonu

3. Řetězce, vstup a výstup

Znak (*character*)

- Je prvek konkrétní znakové sady
- Python 3 používá znakovou sadu *Unicode*
- Příklady znaků v Unicodu:
  - `Abč4()# ,-ΣπЖй日本語¼`
  - Řídící (netisknutelné) znaky (např. nový řádek, zvonek)

Řetězec (*string*)

- Posloupnost znaků
- Datový typ `str`
  - Python nemá speciální datový typ pro znak, jedná se o řetězec délky 1

Zápis řetězců

- Ohraničujeme je pomocí ' nebo " nebo ' ' ' nebo " " "
- Příklad: 4 ekvivalentní zápisy slova Hello

```
'Hello'
"Hello"
'''Hello'''
"""Hello"""
```

Výpis řetězců funkcí `print`

In [1]:
```
retezec = 'Já jsem řetězec.'
```

- Výpis řetězce funkcí `print`
  - V normálním i interaktivním módu
  - Uvozovky se nevypisují

In [2]:
```
print(retezec)
```

```
Já jsem řetězec.
```

- Výpis řetězce jako hodnoty
  - Pouze v interaktivním módu
  - Uvozovky se vypisují

In [3]:
```
retezec
```

Out[3]:
```
'Já jsem řetězec.'
```

Víceřádkové řetězce

- Musíme použít ''' nebo """

In [4]:
```
retezec = "dlouhy retezec
pres hodne radku"
print(retezec)
```

```
  File "<ipython-input-4-76e1db09c7a9>", line 1
    retezec = "dlouhy retezec
                              ^
SyntaxError: EOL while scanning string literal
```

In [5]:
```
retezec = """dlouhy retezec
pres hodne radku"""
print(retezec)
```

```
dlouhy retezec
pres hodne radku
```

Řetězce s uvozovkami / apostrofy

In [6]:
```
print('I'm sorry.')
```

```
  File "<ipython-input-6-e151272baa4b>", line 1
    print('I'm sorry.')
            ^
SyntaxError: invalid syntax
```

In [7]:
```
print("I'm sorry.")
```

```
I'm sorry.
```

In [8]:
```
print("Say "hello".")
```

```
  File "<ipython-input-8-bfbee393ff25>", line 1
    print("Say "hello".")
                      ^
SyntaxError: invalid syntax
```

In [9]:
```
print('Say "hello".')
```

```
Say "hello".
```

In [10]:
```
print('''I can't say "hello".''')
```

```
I can't say "hello".
```

Speciální znaky a escapování

- Speciální znaky je možné zapsat pomocí zpětného lomítka (*backslash*) \
- Nejdůležitější speciální znaky:
  - \n nový řádek
  - \t tabulátor
  - \' apostrof
  - \" uvozovky
  - \\ zpětné lomítko

In [11]:
```
print('A\tB\nC\\nD')
```

```
A	B
C\nD
```

Operace s řetězci

Délka řetězce

- Funkce `len`

In [12]:
```
len('ahoj')
```

Out[12]: 4

In [13]:
```
len('Dobrý den.\n')
```

Out[13]: 11

In [14]:
```
len('')
```

Out[14]: 0

Spojování řetězců (sřetězení, *concatenation*)

- Operátor +

In [15]:
```
'dvě' + ' ' + 'slova'
```

Out[15]:
```
'dvě slova'
```

In [16]:
```
a = 2
b = 'slova'
str(a) + ' ' + b
```

Out[16]:
```
'2 slova'
```

In [17]:
```
a + b
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-17-bd58363a63fc> in <module>()
----> 1 a + b

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Násobení řetězců

- Operátor *

In [18]:
```
'ha' * 5
```

Out[18]:
```
'hahahahaha'
```

Znak po znaku

- Pomocí [ ] a indexu můžeme získat konkrétní znak z řetezce
- Znaky indexujeme zleva, od 0
- Zaporné indexy se počítají zprava, od -1

In [19]:
```
retezec = 'Hello World'
```

In [20]:
```
retezec[0]
```

Out[20]:
```
'H'
```

In [21]:
```
retezec[1]
```

Out[21]:
```
'e'
```

In [22]:
```
retezec[-1]
```

Out[22]:
```
'd'
```

In [23]:
```
retezec[-2]
```

Out[23]:
```
'l'
```

Podřetězec

- Rozsah zapisujeme `[od:do]`
- Index `od` je zahrnut ve výsledku, index `do` nikoliv!
- Prázdne `od` znamená od začatku
- Prázdné `do` znamená do konce

In [24]:
```
retezec = 'Hello World'
```

In [25]:
```
retezec[2:5]
```

Out[25]:
```
'llo'
```

In [26]:
```
retezec[-4:]
```

Out[26]:
```
'orld'
```

In [27]:
```
retezec[:4]
```

Out[27]:
```
'Hell'
```

In [28]:
```
retezec[:]
```

Out[28]:
```
'Hello World'
```

- Přeskakování znaků: `[od:do:krok]`

In [29]:
```
retezec = '0123456789'
retezec[1:8:2]
```

Out[29]:
```
'1357'
```

- Lze vynechat `od`, `do`, nebo oboje

In [30]:
```
retezec[::3]
```

Out[30]:
```
'0369'
```

- Obrácení řetězce: nastavíme `krok` na -1

In [31]:
```
retezec = '0123456789'
retezec[::-1]
```

Out[31]:
```
'9876543210'
```

Řetězce nelze upravovat (*strings are immutable*)

In [32]:
```
retezec = 'Hello World'
```

In [33]:
```
retezec[6]
```

Out[33]:
```
'W'
```

In [34]:
```
retezec[6] = 'X'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-34-9dece0db1f9d> in <module>()
----> 1 retezec[6] = 'X'

TypeError: 'str' object does not support item assignment
```

In [35]:
```
retezec = retezec[:6] + 'X' + retezec[7:]
retezec
```

Out[35]:
```
'Hello Xorld'
```

Hledání podřetězců (*substrings*)

- Operátory `in`, `not in` testují jestli je/není jehla obsažena v kupce sena

In [36]:
```
'123' in 'ABCDefgh1234'
```

Out[36]:
```
True
```

In [37]:
```
'456' in 'ABCDefgh1234'
```

Out[37]:
```
False
```

In [38]:
```
'456' not in 'ABCDefgh1234'
```

Out[38]:
```
True
```

In [39]:
```
'ABCDefgh1234' in '123'
```

Out[39]:
```
False
```

Počítání a hledání

- Pomocí metody `count` počítáme počet výskytů jehel v kupce sena
- Pomocí metody `find` hledáme index prvního výskytu jehly

(Metoda = funkce, kterou voláme přímo na nejakém objektu pomocí tečky.)

In [40]:
```
retezec = 'Nesnese se se sestrou.'
retezec.count('se')
```

Out[40]: 4

In [41]:
```
'se'.count(retezec)
```

Out[41]: 0

In [42]:
```
retezec.find('se')
```

Out[42]: 5

Hledání pouze na začátku / na konci

- Metody `startswith`, `endswith`

In [43]:
```
retezec = 'Nesnese se se sestrou.'
retezec.startswith('se')
```

Out[43]:
```
False
```

In [44]:
```
retezec.startswith('Nes')
```

Out[44]:
```
True
```

In [45]:
```
retezec.endswith('.')
```

Out[45]:
```
True
```

Nahrazování

- Metoda `replace` nahradí starý podřetězec (`old`) za nový (`new`)
- Volitelný třetí parameter `count` nastaví maximalní počet nahrazení; pokud není nastavený, nahradí se všechny výskyty

```
In [46]: 'kormorán'.replace('or', 'ol')
```

```
Out[46]: 'kolmolán'
```

```
In [47]: 'kormorán'.replace('or', 'ol', 1)
```

```
Out[47]: 'kolmorán'
```

Rozdělení řetězce na části

- Metoda `split` rozdělí řetězec dle zadaného separátoru
- Pokud není separátor nastaven, tak se berou defaultně shluky bílých znaků (mezera, `\t`, `\n`)
- (Tato metoda vrací *seznam* řetězců. Seznamy se budeme víc zabývat později.)

In [48]:
```
retezec = 'dvě  slova \t tři celá slova'
retezec.split()
```

Out[48]:
```
['dvě', 'slova', 'tři', 'celá', 'slova']
```

In [49]:
```
retezec.split(' ')
```

Out[49]:
```
['dvě', '', 'slova', '\t', 'tři', 'celá', 'slova']
```

In [50]:
```
jmeno, prijmeni = 'Jan Novák'.split()
jmeno
```

Out[50]:
```
'Jan'
```

In [51]:
```
prijmeni
```

Out[51]:
```
'Novák'
```

Odstranění bílých znaků na okrajích

- Metoda `strip` odstraní bílé znaky z obou konců řetězce
- Metoda `lstrip` odstraňuje pouze zleva (*left-strip*)
- Metoda `rstrip` odstraňuje pouze zprava (*right-strip*)
- Bílé znaky uvnitř řetězce jsou zachovány
- (Psst! Tyto metody fungují podobně jako metoda `split`, přijmají volitelný parametr, který popisuje, co se má z okrajů odstranit.)

In [52]:
```
retezec = '    ja jsem nepovedený \n řetězec \t\n'
```

In [53]:
```
print(retezec)
```

```
    ja jsem nepovedený
 řetězec
```

In [54]:
```
print(retezec.strip())
```

```
ja jsem nepovedený
 řetězec
```

In [55]:
```
print(retezec.lstrip())
```

```
ja jsem nepovedený
 řetězec
```

In [56]:
```
print(retezec.rstrip())
```

```
    ja jsem nepovedený
 řetězec
```

Změna velikosti písma

In [57]:
```
retezec = 'ABCD efgh Ijkl'
```

In [58]:
```
retezec.upper()
```

Out[58]:
```
'ABCD EFGH IJKL'
```

In [59]:
```
retezec.lower()
```

Out[59]:
```
'abcd efgh ijkl'
```

In [60]:
```
retezec.swapcase()
```

Out[60]:
```
'abcd EFGH iJKL'
```

In [61]:
```
retezec.capitalize()
```

Out[61]:
```
'Abcd efgh ijkl'
```

In [62]:
```
retezec.title()
```

Out[62]:
```
'Abcd Efgh Ijkl'
```

Logické operace

- `isalpha` - obsahuje pouze písmena?
- `isdigit` - obsahuje pouze číslice?
- `isalnum` - obsahuje pouze písmena a číslice?
- `isspace` - obsahuje pouze bílé znaky?
- `isupper`, `islower` - jsou všechna písmena velká/malá?

In [63]:
```
'ABCDefgh1234'.isalnum()
```

Out[63]:
```
True
```

In [64]:
```
'ABCD efgh 1234'.isalnum()
```

Out[64]:
```
False
```

In [65]:
```
' \t\n\r'.isspace()
```

Out[65]:
```
True
```

In [66]:
```
'a \t\n\r'.isspace()
```

Out[66]:
```
False
```

In [67]:
```
'Mám 5 jablíček.'.islower()
```

Out[67]:
```
False
```

In [68]:
```
'mám 5 jablíček.'.islower()
```

Out[68]:
```
True
```

In [69]:
```
'A Je To Tady'.istitle()
```

Out[69]:
```
True
```

Formátování řetězce

- Pomocí `%` (zastaralé, nepoužívat)
- Pomocí metody `format`
- Pomocí f-stringů

Metoda `format`

- Máme šablonu, do které umisťujeme značky `{}`, ty se pak nahradí hodnotami z parametrů funkce

In [70]:
```
sablona = 'Jmenuji se {} a je mi {} let.'
```

In [71]:
```
sablona.format('Anička', 5)
```

Out[71]:
```
'Jmenuji se Anička a je mi 5 let.'
```

- Značky můžeme indexovat (od 0, žádné číslo v sekvenci nesmí chybět)

In [97]:
```
sablona = 'Jmenuji se {}, je mi {} let, líbí se mi {}.'
sablona.format('Anička', 5, 'Prasátko Peppa')
```

Out[97]:
```
'Jmenuji se Anička, je mi 5 let, líbí se mi Prasátko Peppa.'
```

In [98]:
```
sablona = 'Jmenuji se {2}, je mi {1} let, líbí se mi {0}.'
sablona.format('Anička', 5, 'Prasátko Peppa')
```

Out[98]:
```
'Jmenuji se Prasátko Peppa, je mi 5 let, líbí se mi Anička.'
```

In [99]:
```
sablona = 'Jmenuji se {2}, je mi {0} let, líbí se mi {1}.'
sablona.format('Anička', 5, 'Prasátko Peppa')
```

Out[99]:
```
'Jmenuji se Prasátko Peppa, je mi Anička let, líbí se mi 5.'
```

- Značky můžeme pojmenovat

In [101]:
```
sablona = 'Jmenuji se {jmeno}, je mi {vek} let, líbí se mi {co}.'
sablona.format(vek=5, co='Prasátko Peppa', jmeno='Anička')
```

Out[101]:
```
'Jmenuji se Anička, je mi 5 let, líbí se mi Prasátko Peppa.'
```

Typy a formátování

- Ve značce za dvojtečkou definujeme
  - Zarovnání: `{:<}`, `{:>}` nebo `{:^}`
  - Délku: `{:10}`
  - Počet desetinných míst: `{:.2}`
  - Typ/formát: `{:s}` řetězec, `{:n}` číslo, `{:f}` reálné číslo, `{:e}` vědecký formát čísla...
- Zadané pořadí je nutné dodržet

```
In [76]: '{}'.format(1000.2)
```

```
Out[76]: '1000.2'
```

```
In [77]: '{:>20.2e}'.format(1000.2)
```

```
Out[77]: '            1.00e+03'
```

```
In [78]: '{x:^20.2E}'.format(x=1000.2)
```

```
Out[78]: '      1.00E+03      '
```

f-strings

- "The best of Python 3.6"
- Fungují podobně jako `format`, ale značky musí být pojmenovány
- Hodnoty se berou přímo z proměnných v prostředí

In [79]:
```
jmeno = 'Anička'
vek = 5
co = 'Prasátko Peppa'
```

In [80]:
```
f'Jmenuji se {jmeno}, je mi {vek:.2f} let, líbí se mi {co:^25}.'
```

Out[80]:
```
'Jmenuji se Anička, je mi 5.00 let, líbí se mi      Prasátko Peppa      .'
```

Vstup a výstup

Vstup (*input*)

- Slouží pro předání informací do běžícího programu
- Funkce `input`

Výstup (*output*)

- Slouží pro předání informací ven z běžícího programu
- Funkce `print`

Funkce `input`

- Uživateli vypíše hlášku
- Čeká na vstup od uživatele až do stisknutí klávesy Enter
- Výsledkem funkce je řetězec, který zadal uživatel

Zkuste si spustiť tento kód:

In [82]:
```
jmeno = input('Jak se jmenuješ? ')
vek = input('Kolik ti je let? ')
co = input('Co se ti líbí? ')
print(f'Jmenuješ se {jmeno}, je ti {vek} let, líbí se ti {co}.')
```

```
Jak se jmenuješ? Honza
Kolik ti je let? 7
Co se ti líbí? Pokémoni
Jmenuješ se Honza, je ti 7 let, líbí se ti Pokémoni.
```

Funkce `print`

- Všechny své parametry přemění na řetězce a vypíše je

In [83]:
```
print('ahoj', 5, True)
```

```
ahoj 5 True
```

Speciální parametry funkce `print`

- Parametr `sep` (default je `' '`)

In [84]:
```
print(1, 2, 3)
```

```
1 2 3
```

In [85]:
```
print(1, 2, 3, sep=', ')
```

```
1, 2, 3
```

In [86]:
```
print(1, 2, 3, sep='\t')
```

```
1	2	3
```

In [87]:
```
print(1, 2, 3, sep='\n')
```

```
1
2
3
```

In [88]:
```
print(1, 2, 3, sep='')
```

```
123
```

- Parametr `end` (default je `'\n'`)

In [89]:
```
print(1, 2, 3)
print(4, 5, 6)
```

```
1 2 3
4 5 6
```

In [90]:
```
print(1, 2, 3, end=';')
print(4, 5, 6)
```

```
1 2 3;4 5 6
```

In [91]:
```
print(1, 2, 3, sep=',', end=' ---> ')
print(4, 5, 6, sep='|', end='.')
```

```
1,2,3 ---> 4|5|6.
```

Reprezentace znaků v počítači

- Každý znak je v znakové sadě reprezentován svým ordinálním číslem
- Funkce `ord` zjišťuje ordinální číslo znaku
- Opakem je funkce `chr`, která vrací znak pro zadané ordinální číslo

- Ordinální čísla pro znakovou sadu ASCII = prvních 128 znaků Unicodu

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Dec = o.č. v desítkové soustavě, Hex = o.č. v šestnáctkové soustavě, Char = znak

In [92]:
```
ord('A')
```

Out[92]:
```
65
```

In [93]:
```
ord('č')
```

Out[93]:
```
269
```

In [94]:
```
chr(97)
```

Out[94]:
```
'a'
```

In [95]:
```
chr(367)
```

Out[95]:
```
'ů'
```

- Escapování pomocí ordinálních čísel:
  - `\x??` kde ?? je ordinální číslo znaku v šestnáctkové soustavě
  - `\u????` kde ???? je ordinální číslo znaku šestnáctkové soustavě
  - `\U????????` kde ???????? je ordinální číslo znaku v šestnáctkové soustavě
  - `\N{name}` kde name je název Unicode znaku

In [96]:
```
print('\x41 \u0041 \U00000041 \xE1 \N{pound sign}')
```

```
A A A á £
```