Datové typy a jejich reprezentace v počítači.

Celá čísla. Reálná čísla. Semilogaritmický tvar. Komplexní čísla. Řetězce.

Tomáš Bayer | bayertom@natur.cuni.cz

Katedra aplikované geoinformatiky a kartografie, Přírodovědecká fakulta UK.

Obsah přednášky

1. Základní datové typy

Logické, celočíselné, reálné a znakové.

Existují prakticky ve všech programovacích jazycích.

Definován rozsah hodnot v bajtech (velikost uloženého čísla).

Python dynamicky typovaný jazyk, datový typ u proměnné nemusíme uvádět.

Celočíselné datové typy:

Reprezentace diskrétních jevů, výpočty “bezchybné”.

Reálné datové typy:

Reprezentace spojitých jevů, výpočty zatíženy chybou.

Vyjádření čísla a v různých číselných soustavách o základu z

$$(a)_z = \pm \sum_{i=0}^{n} b_i z^i, \qquad b_i \in < 0, z-1 >,$$

Dvojková (binární): $z = 2$, $b_i = 0, 1$. Např.: $(a)_2 = 0011$,

Desítková (decimální): $z = 10$, $b_i = 0, 1, ..., 9$. Např. $(a)_{10} = 1987$,

Šestnáctková (hexadecimální): $z = 16$, $b_i = 0, 1, ..., 9, a, ...f$. Např. $(a)_{16} = 16af$.

Znakové typy:

Vyjádření textových informací, znak reprezentován celým číslem.

2. Celá čísla a jejich reprezentace v počítači

Reprezentovány celočíselnými datovými typy,

Liší se rozsahem hodnot, které lze uložit.

Klasifikace dle:

- ***Přesnosti:***
  Typy s nižší přesností, označovány `short`.
  Typy s vyšší přesností, označovány `long`.
- ***Znaménka:***
  Typy se znaménkem ($\mathbb{Z}$) a bez znaménka ($\mathbb{Z}^+$).
  Bez znaménka označovány `unsigned`.
  Se znaménkem označovány `signed`.

V konkrétním jazyku nemusí být zastoupeny všechny kombinace typů.

Volí se dle oboru funkčních hodnot konkrétní proměnné.

Pozor na přeplnění.

Informace o typu v Pythonu:

```
a = -8
type(a)
>> <class 'int'>
```

3. Přímý, inverzní, doplňkový kód

Číslo v PC uloženo v binárním tvaru

$$(a)_2 = \pm \sum_{i=0}^{n} b_i 2^i, \qquad b_i = 0, 1.$$

1 bit znaménkový $m = n - 1$.

Varianty znázornění:

- *Přímý kód:* $a \in \langle -2^m - 1, 2^m - 1 \rangle$
  Na první pozici znaménkový bit ,dvojí reprezentace nuly

  $$P(0)_{10} = 00000000, \qquad P(-0) = 10000000.$$

  Nezachovává relace

  $$P(105)_{10} = (01101001)_2 < P(-105)_{10} = (11101001)_2.$$

- *Inverzní kód:* $a \in \langle -2^m - 1, 2^m - 1 \rangle$
  Záporné číslo negací (jedničkový doplněk), dvojí reprezentace nuly

  $$I(0)_{10} = 00000000, \qquad I(-0) = !00000000 = 11111111.$$

  Zachovává relace

  $$I(105)_{10} = (01101001)_2 > I(-105)_{10} = !(01101001)_2 = (10010110)_2.$$

- *Doplňkový kód:* $a \in \langle -2^m, 2^m - 1 \rangle$
  Připočtení 1 k jedničkovému doplňku (dvojkový doplněk), jedna reprezentace 0

  $$D(0)_{10} = 00000000 = D(-0) = 11111111 + 1 = 00000000.$$

  Zachovává relace

  $$D(105)_{10} = (01101001)_2 > D(-105)_{10} = I(-105)_{10} + 1 = (10010111)_2.$$

  Standardní reprezentace v PC.

4. Celočíselné hodnoty

Rozsah závisí na počtu bitů n použitých pro uložení číselné hodnoty v paměti

Sig.	$\pm$	2^{m-1}	$\cdots$	2^3	2^2	2^1	2^0

Unsig.	2^{n-1}	$\cdots$	2^4	2^3	2^2	2^1	2^0

Přehled celočíselných typů:

Velikost	Rozsah	Znaménko	C++	Java	Python 2.x	Python 3
1B	$-128, 127$	ano	signed char	byte	-	
1B	$0, 255$	ne	unsigned char	-	-	
2B	$-32768, 32767$	ano	int	short	int	
2B	$0, 65536$	ne	unsigned	-	-	
4B	$-2^{31}, 2^{31}-1$	ano	long	int	long	
4B	$0, 2^{32}-1$	ne	unsigned long	-	-	
8B	$-2^{63}, 2^{63}-1$	ano	-	long	-	
	$-\infty, \infty$					int

Pokud nevíme, jaký datový typ použít, a nemáme nějaké speciální požadavky, zpravidla `int`.

Python v. Python 3: sjednocení `int` a `long` $\Rightarrow$ `int`.

Omezení pouze dostupnou pamětí, nemají horní limit (netypické).

5. Logické hodnoty

Vyjádřeny jako booleovské proměnné.
V Pythonu typ `Boolean`.

Obor hodnot: pravda `True`, nepravda `False`.

Mají celočíselnou reprezentaci (z Pythonu 2.x):

```
1 = True
0 = False.
```

Příklad:

```
a = False
b = True
c = 2 * b + a;
>> 1
```

Uplatňuje se zejména při konstrukci logických výrazů.
Typické u podmínek.

```
if logicky_vyraz:
    # do something
```

6. Reálná čísla a jejich reprezentace v počítači

Většina výpočtů probíhá právě s reálnými čísly $\mathbb{R}$.

Zahrnují přirozená $\mathbb{N}$, celá $\mathbb{Z}$, racionální $\mathbb{Q}$ čísla, 0.

V binární soustavě lze složením celé části (n bitů), desetinné části (m bitů), znaménkového bitu

$$(a)_2 = \pm(a_c + a_d),$$

kde

$$a_c = \sum_{i=0}^{n} b_i 2^i, \qquad a_d = \sum_{i=-1}^{-m} b_i 2^i.$$

Reálné číslo lze vyjádřit konečným/nekonečným desetinným rozvojem.

Používána pro popis spojitých jevů.

Výpočty pomalejší než nad celými čísly.

Desetinná čísla na rozdíl od celých reprezentována “nepřesně” (s chybou).

Důležitá volba správného datového typu.

Reálná čísla lze v počítači reprezentovat dvěma způsoby:

- s pevnou řádovou čárkou (FX),
- s pohyblivou řádovou čárkou (FP).

7. Zobrazení s pevnou řádovou čárkou

Řádová čárka oddělující umístěna na pevné pozici.

$\pm$	2^{n-1}	$\dots$	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	$\dots$	2^{-m}

První bit znaménkový, pro celočíselnou část vyhrazeno n, pro desetinnou m bitů.

Rozsah zobrazitelných čísel: $a \in \langle -2^{n-1} - 2^{-m}, 2^{n-1} + 2^{-m} \rangle$.
Přesnost reprezentace závisí na počtu bitů použitých pro jejich reprezentaci.

"Přesná" čísla mohou být znázorněna nepřesně: $(a)_{10} \neq ((a)_2)_{10}$.

Absolutní chyba reprezentace

$$\Delta = \left| (a)_{10} - ((a)_2)_{10} \right| .$$

Relativní chyba reprezentace

$$\delta = \frac{\Delta}{(a)_{10}} .$$

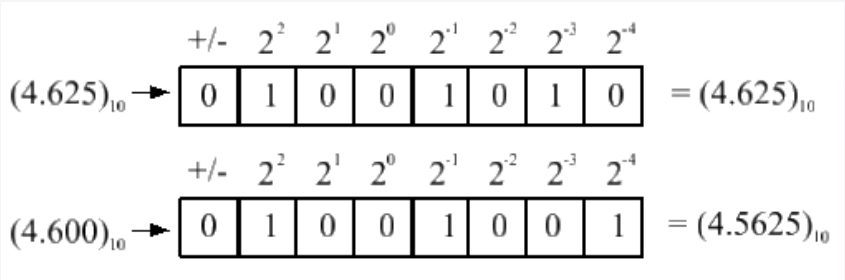
- Všechna čísla uchována se stejnou ***absolutní*** chybou Δ .
- "Malá" čísla: $a \to 0 \Rightarrow \delta \to \infty$, malá relativní přesnost.
- Nelze reprezentovat taková čísla, která ve FP jdou.

Při vědeckých výpočtech: všechna čísla se stejnou ***relativní*** chybou ε.

8. Příklady

8 bitová FX reprezentace: (3b+ 4b), 1b znaménko.

Znázornění čísel $(a_1)_{10} = 4.625$, $(a_2)_{10} = 4.600$ v FX reprezentaci.



První číslo v binární soustavě lze reprezentovat přesně.

Druhé číslo v binární soustavě periodické

$$(4.6)_{10} = (100.\overline{1001})_2.$$

Absolutní chyba FX reprezentace

$$\Delta_1 = |(a_1)_{10} - ((a_1)_2)_{10}| = 0, \quad \Delta_2 = |(a_2)_{10} - ((a_2)_2)_{10}| = 0.0375.$$

Relativní chyba FX reprezentace

$$\delta_1 = \frac{\Delta_1}{a_1} = 0, \quad \delta_1 = \frac{\Delta_2}{a_2} = 0.0082 \doteq 0.8\%.$$

Vliv hodnoty čísla na δ:

Čísla $(a_1)_{10} = 1000.6$, $(a_2)_{10} = 1.6$.

Absolutní chyba FX reprezentace: $\Delta_1 = \Delta_2 = 0.0375$.

Relativní chyba FX reprezentace

$$\delta_1 = \frac{\Delta_1}{a_1} \doteq 0.0023\%, \quad \delta_2 = \frac{\Delta_2}{a_2} \doteq 2.3\%.$$

9. Zobrazení čísel s pohyblivou řádovou čárkou (FP)

Semilogaritmický tvar čísla

$$a = q \cdot z^e,$$

kde q je mantisa, z základ číselné soustavy, e exponent.

Splňuje *normalizační podmínku* ve tvaru

$$\frac{1}{z} \leq q < 1.$$

Levá strana: prevence ztráty přesnosti.

Pravá strana: prevence přeplnění.

Číslo splňující normalizační podmínku nazýváme *normalizované*.

Binární číslo

$$a = q \cdot 2^e, \qquad \frac{1}{2} \leq q < 1.$$

Mantisa reprezentována co nejpřesněji (zpravidla 23 b).

Pro uchovávání exponentu postačí méně míst (zpravidla 8 b).

$\pm$	2^{n-1}	$\ldots$	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	$\ldots$	2^{-m}
	exponent e						mantisa q							

\+ Efektivnější ukládání dat.

\+ Stejná relativní přesnost všech čísel.

\+ Vhodná pro příliš malá nebo příliš velká čísla.

10. Příklady: znázornění čísla v FP

Znázornění čísla $2^{16} = 65536$ v FP reprezentaci (1+5):

$$a = (65536)_{10} = \left((65536/2^{17})2^{17}\right)_{10} = \left(+0,5 \cdot 2^{+17}\right)_{10} = (0|1 \cdot 10001)_2 ,$$

$$\Delta = (65536 - 0.5 \cdot 2^{+17}) = 0.0, \delta = \Delta/a = 0.0\%.\text{Přesné.}$$

0	1	0	0	0	1	1

Znázornění čísla 4.625 v FP reprezentaci (6+4):

$$a = (4.625)_{10} = ((4.625/2^3)2^3)_{10} = \left(+0,578125 \cdot 2^{+3}\right)_{10} = (0|100101 \cdot 1000)_2 ,$$

$$\Delta = (4.625 - (2^{-1} + 2^{-4} + 2^{-6})8) = 0, \delta = \delta/a = 0\%.\text{Přesné.}$$

0	1	0	0	0	1	0	0	1	0	1

Znázornění čísla 4.6 v FP reprezentaci (6+4):

$$a = (4.600)_{10} = \left((4.600/2^3)2^3\right)_{10} = \left(+0,575 \cdot 2^{+3}\right)_{10} \approx (0|100100 \cdot 1000)_2 ,$$

$$\Delta = (4.600 - (2^{-1} + 2^{-4})8) = 0.100, \delta = \Delta/a = 2.2\%.\ \text{Nepřesné.}$$

0	1	0	0	0	1	0	0	1	0	0

V FX reprezentaci (rámcově) srovnatelná přesnost.

11. Přetečení, podtečení

Dochází k nim při práci s “příliš velkými” či “příliš malými” čísly.

- *Podtečení:*
  Pokud $a < a_{min}$, dochází k podtečení.
- *Přetečení:*
  Pokud $a > a_{max}$, dochází k přetečení.

Důsledkem obou operací je ztráta přesnosti, dojde k zaokrouhlení čísla.

Důsledek přetečení:

Výsledek nevejde do počtu bitů určených pro exponent.

Přetečení do znaménkového bitu, změna znaménka.

Znázornění čísla $2^{32} = 4294967296$ v FX reprezentaciích (1,5), (4,6):

$$a = (4294967296)_{10} = \left((4294967296/2^{33})2^{33}\right)_{10} = \left(+0.5 \cdot 2^{+33}\right)_{10} = (0|1 \cdot 100001),$$

$$\Delta = (0,5 \cdot 2^{+33} - (-0.5)), \delta = \Delta/a = 100.0\%.\text{Přetečení}.$$

1	0	0	0	0	1	1

0	1	0	0	0	0	1	1	0	0	0

Důsledek podtečení:

Pokud dojde k podtečení, je výsledkem operace hodnota $+0$ nebo -0.

12. IEEE 754

Definice standardů pro dvojkovou aritmetiku v FP (1985):

- *Přesnost*
  Jednoduchá přesnost (32 bit), dvojitá (64 bit), dvojitá rozšířená (80 bit).

Typ	$\pm$	Mantisa	Exponent	Platné cifry	ε
`float`	1	23	8	7	$2^{-22} \approx 2.38 \cdot 10^{-7}$
`double`	1	47	16	15	$2^{-46} \approx 1.42 \cdot 10^{-14}$

- *Nečíselný výsledek, NaN (Not a Number).*
  Výsledek operace není definován (např. odmocnina ze záporného čísla).

```
math.sqrt(-1)
>>> ValueError: math domain error
```

- *Nekonečno, Inf (Infinity)*
  Výsledek aritmetických operací (např. dělení 0).

```
math.sqrt(1/0)
>>> ZeroDivisionError: division by zero
```

- *Přetečení, podtečení*
  Mezní hodnoty: $x_{min} = 2^{-126}$, $x_{max} = 2^{127}$.

  $x < 0 \wedge x < -x_{max}$: negative overflow,
  $x < 0 \wedge x > -x_{min}$: negative underflow,
  $x > 0 \wedge x < x_{min}$: positive underflow,
  $x > 0 \wedge x > x_{max}$: negative overflow.

13. Reálná čísla a jejich reprezentace v počítači

Přehled reálných datových typů v programovacích jazycích:

- C++,
- Java,
- Python.

Velikost	Rozsah	Znaménko	C++	Java	Python
4B	$1.4 \cdot 10^{-45}, 3.4 \cdot 10^{38}$	ano	`float`	float	float
8B	$4.9 \cdot 10^{-234}, 1.7 \cdot 10^{308}$	ano	`double`	double	-
10B	$1,190 \cdot 10^{4932}$	ano	`long double`	-	-
		ano	-	-	complex

14. Problémy při práci s reálnými čísly (porovnání):

V průběhu výpočtů dochází k zaokrouhlování a následné ztrátě přesnosti.

Porovnáváme -li $a, b \in \mathbb{R}$, kde $a = b$, pak výraz

$$a == b$$

v obecném případě vyhodnocen jako `false`.

Nepoužívat podmínky

```
if a == b: nebo while a == b:
```

Testujeme hodnotu $|a - b|$ nebo $|(a - b)/a|$ vzhledem k $\varepsilon \gg 0$:

$$|a - b| < \varepsilon, \qquad \left|\frac{a - b}{a}\right| < \varepsilon.$$

Upravená varianta podmínky

```
if abs(a - b) < eps: nebo while abs(a - b) < eps:
```

V důsledku zaokrouhlení nemusí platit:

- komutativní zákony,
- asociativní zákony,
- distributivní zákon.

15. Problémy při práci s reálnými čísly (zákony):

Komutativní zákony

$$\begin{aligned} a+b &\neq b+a, \\ a-b &\neq a+(-b), \\ -(a+b) &\neq -a-b, \\ ab &\neq ba, \\ (-a)b &\neq -(ab), \end{aligned}$$

Asociativní zákony

$$\begin{aligned} a+(b+c) &\neq (a+b)+c, \\ a(bc) &\neq (ab)c. \end{aligned}$$

Distributivní zákon

$$(a+b)c \neq ac+bc.$$

Rychlost aritmetických operací:

- U $\mathbb{Z}$ řádově vyšší než u $\mathbb{R}$.
- Při návrhu algoritmu zvážit, které proměnné $\mathbb{Z}$, a které $\mathbb{R}$.

16. Problémy při práci s reálnými čísly (podtečení)

Problém 1: Odečtení malého čísla od jiného čísla

Výsledek operace je špatný.

```
x = 1.0;
y = 0.00000000000000000005
z = x - y
>>> 1.0
```

Problém 2: Přičtení malého čísla k velkému číslu

Porovnání dá špatný výsledek.

```
x = 1.0e20
x == x + 1
>>> True
```

Problém 3: Nefunkčnost asociativity pro malá a velká čísla

```
a = 1
b = 1e20
c = -1e20
a + ( b + c )
( a +  b ) + c

>>> 1.0
>>> 0.0
```

17. Reálná čísla s volitelnou přesností, komplexní čísla

Reálná čísla s volitelnou přesností:

Přesnost `float` reprezentace nemusí být postačující.

Pro přesnější výpočty existuje v Pythonu modul `decimal`.

Pomalejší než float, před použitím nutno importovat...

```
import decimal
     a = decimal.Decimal(1234)
b = decimal.Decimal("5678.012345678901234567890")
a+b
>>> Decimal('6912.012345678901234567890')
```

Použití pro vědecké výpočty s vysokou přesností (souřadnicové).

Komplexní čísla:

Imaginární část označena identifikátorem j.

```
z = -2.3 + 3.4j
z.real
>>> -2.3
z.imag
>>> 3.4
z.conjugate()
>>>(-2.3 - 3.4j)
print(abs(z))
>>> 4.104875150354758
print(pow(z,2))
>>> (-6.27-15.639999999999999j)
```

18. Kódování znaků

Reprezantace znaků v PC (zpravidla) číselným kódem.

Ze znaků skládány posloupnosti, tzv. *textové řetězce*.

Standardizace znakových sad v informatice:

- *ASCII (American Standard Code for Information Interchange), 1967*
  127 znaků (písmena, číslice, spec. znaky), 7b + 1b.
  Nevyhovující, reprezentace malého množství znaků.
  Problémy s ČJ, celkem 6 speciálních kódování: CP 1250, ISO 8859-2, Latin 2...
- *UNICODE (Universal Coded Character Set), 1991*
  16b kódování, nástupce ASCII, 1 114 112 znaků.
  Umožňuje vyjádřit libovolný znak z libovolného jazyka.
  Nejčastější varianta kódování: UTF-8.

Speciální (řídící) znaky reprezentovány ***Escape sekvencemi.***

Uvozující znak \ (backslash).

Escape sekvence	UNICODE	Význam
\n	\u000A	Nová řádka
\r	\u000D	Návrat na začátek řádku
\t	\u0009	Tabulátor
\\	\u005C	Zpětné lomítko
\'	\u002C	Apostrof
\"	\u0022	Uvozovky

```
print("We are \t ready to \n learn \"Python\"")
>>> We are    ready to
>>> learn "Python"
```

19. ASCII tabulka

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

20. Textový řetězec v Pythonu

Posloupnost znaků uzavřená v apostrofech/uvozovkách (Single/Double/Triple Quotes).

Podpora UNICODE.

```
a = 'Hello'      #Single quotes
b = "Hello"      #Double quotes
c = '"Hello"'    #Combination
c = "'Hello'"    #Combination
c ='''Hello'''   #Triple quotes
d ="""Hello"""   #Triple double quotes
```

Triple Quotes pro víceřádkový text

```
e = """We are ready to
     learn "Python""")
```

V Pythonu řetězec konstantní (řetězcová konstanta), nelze modifikovat.

Sčítání řetězcových konstant prostřednictvím operátoru +.

```
s = "Hello \n" + "world";
     >>> Hello
     >>> world
```

Délka řetězce:

```
len(s)
     >>> 13
```

Výskyt podřetězce v řetězci:

```
"wo" in s
     >>> True
```

21. Operace s řetězci v Pythonu (1/2)

Práce s řetězci podobná práci s polem, každý znak má index.

Index obousměrný.

s[-11]	s[-10]	s[-9]	s[-8]	s[-7]	s[-6]	s[-5]	s[-4]	s[-3]	s[-2]	s[-1]
H	e	l	l	o		w	o	r	l	d
s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]	s[9]	s[10]

Lze vytvářet podřetězce (slices):

```
seq[index] #1 znak
seq[start:end] #interval od-do
seq[start:] #vsechny znaky od
seq[:end] #vsechny znaky od
seq[start:end:step] #Interval od-do s krokem
```

Příklady:

```
s[0:10:2]
>>> Hlwr
s[::2]
>>> Hlwl
s[::-2]
>>> drwolH
```

22. Operace s řetězci v Pythonu (2/2)

Replikace řetězců:

```
print(s * 3)
>>> Hello worldHello worldHello world
```

Konverze znaku na číslenou reprezentaci:

```
ord(d)
>>> 100
```

Konverze číselné reprezentace na textovou:

```
chr(100)
>>> d
```

Znak s maximálním/minimálním ASCII kódem:

```
min(s)
>>>  #mezera
max(s)
>>> w
```

Delimitace:

```
s = "Hello*my*new*world"
s.split('*')
>>>  ['Hello', 'my', 'new', 'world']
```

Další užitečné funkce jsou v modulu `string`:

```
import string
```