Počítání s čísly
(3. přednáška)

FF UK
Department of Logic

Jonathan L. Verner

Datové typy

FF UK
Department of Logic

- Počítače původně pracovaly hlavně s čísly.
- Data = Informace ukládané a zpracovávané počítačem.

```
def main():
  print("Program na pocitani uzitneho objemu vozoveho parku
  print()
  typ = input("Zadej typ auta:")
  objem   = float(input("Zadej objem nakladniho prostoru:")
  pocet = int(input("Zadej pocet aut:"))
  celkem = pocet * objem
  print("Celkovy uzitny objem aut typu", typ, "je", celkem)
main()
```

- Tři různé typy dat: objem, počet aut, typ auta
- Další možné: cena benzínu, barva auta, jméno řidiče ...

Datové typy

FF UK
Department of Logic

- Paměť umí ukládat pouze `0,1`
- Počítač potřebuje vědět, jak tyto `0,1` uložené v paměti interpretovat a co s nimi lze dělat (např. asi nedává smysl sčítat mezi sebou dvě jména)
- Datový typ určuje, jak tuto posloupnost interpretovat
- Každá proměnná, výraz, literál má datový typ

Datové typy & Proměnné

FF UK
Department of Logic

- Static typing typ proměnné je znám dopředu
- Dynamic typing typ proměnné se určuje za běhu
- Strong typing nelze sčítat jablka a hrušky
- Weak typing jablka a hrušky sčítat lze

```
a = 5       // a ma typ "cislo"
a = "Ahoj" // Chyba pri static typing
a = a + 5  // Chyba pri strong typing
```

	Static Typing	Dynamic Typing
Weak Typing	Perl	Javascript
Strong Typing	Pascal, Java, C++	Python

Datové typy v Pythonu

FF UK
Department of Logic

- číslo: celá (`int`), reálná (`float`), komplexní, Booleovská
- posloupnost: unicode řetězce, n-tice (dvojice, trojice, ...), seznamy
- množina
- slovník
- volatelný typ: funkce,příkaz, . . .
- None: nic
- moduly, třídy, ...

Datové typy v Pythonu

FF UK
Department of Logic

Jak poznáme datový typ proměnné, literálu ...?

```
>>> type(3)
<class 'int'>
>>> type(3.14)
<class 'float'>
>>> type(3.0)
<class 'float'>
>>> type(myInt)
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    type(myInt)
NameError: name 'myInt' is not defined
>>> myInt=-32
>>> type(myInt)
<class 'int'>
>>> myInt=3.2
```

Celá čísla: `int`

FF UK
Department of Logic

- 16/32/64 bitů
- little-endian x high-endian
- most significant bit znaménko: $0 = +, 1 = -$
- "$-x = 2^n - x$", kde n je počet bitů bez znaménka

+/-	bin. repreze.							dek. rep.
0	1	1	1	1	1	1	1	127
0	1	1	1	1	1	1	0	126
0	0	0	0	0	0	1	0	2
0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	-1
1	1	1	1	1	1	1	0	-2
1	0	0	0	0	0	0	1	-127
1	0	0	0	0	0	0	0	-128

n-bitový int

FF UK
Department of Logic

Jaká čísla lze takto reprezentovat pomocí n-bitů ?

$$[-2^{n-1}, 2^{n-1})$$

Soutěž (2 body) Faktoriál

FF UK
Department of Logic

Nalezněte nejmenší n takové, že $n! > x$

- Povoleno použít pouze Python!

```
x = 9 332621 544394 415268 169923 885626 670049
071596 826438 162146 859296 389521 759999 322991
560894 146397 615651 828625 369792 082722 375825
  118521 091686 400000 000000 000000 000000 0
```

http://jonathan.temno.eu/cislo.html

Číselné operace

FF UK
Department of Logic

operator	operace
`+`	sčítání
`-`	odčítání
`*`	násobení
`/`	dělení
`//`	celočíselné dělení
`%`	zbytek
`**`	mocnění
`abs()`	absolutní hodnota

Převod do binární soustavy Specifikace

FF UK
Department of Logic

- Program na vstupu načte kladné celé číslo M.
- Program vytiskne na výstup binární reprezentaci čísla M, nejméně významný bit (lsb) bude nejvíce vpravo
- Program může použít funkci `math.log(x,2)`, která vrátí binární logaritmus čísla x.
- Program má k dispozici funkci `str(x)`, která vrátí číslo `x` jako řetězec (t.j. `str(1)="1"`).

Převod do binární soustavy Návrh

Povšimneme si, že platí následující: Jsou-li $a_i \in 0, 1$ pro $i = 0, \ldots, n$ pak platí,

$$\left(\sum_{i=0}^{n} a_i 2^i\right) \mod 2 = a_0$$

a, je-li $k < n$,

$$\left(\sum_{i=0}^{n} a_i 2^i\right) / 2^k = \sum_{i=k}^{n} a_i 2^{i-k}$$

- Najdi n tak, aby $M < 2^n$.
- Do k sestupně přiřazuj čísla od n do 0:
  - pomocí druhé rovnice spočti `head`$=\sum_{i=k}^{n} a_i 2^{i-k}$
  - pomocí první rovnice z `head` spočti binární číslici na k-tém místě
  - připoj jí k dosud spočítaným číslicím na vyšších místech

Převod do binární soustavy Implementace

FF UK
Department of Logic

```
# 2bin.py: Program na prevod z desitkove do
#          binarni soustavy
import math

def tobin(M):
    n = int(math.log(M,2))+1
    bin_rep = ""
    for i in range( n + 1 ):
        k  = n - i
        head = M // 2**k
        ak = head % 2
        bin_rep = bin_rep + str(ak)
    print(bin_rep)

def main():
    x = input("Zadej kladne cele cislo:")
```

Reálná (racionální) čísla `float`

FF UK
Department of Logic

- založen na tzv. "scientific notation"

$$1.27618\ \mathrm{e} + 17 = 1.27618 \cdot 10^{17}$$

- První bit určuje znaménko
- Dalších 11 bitů určuje exponent
- Dalších 52 bitů určuje mantissu
- `inf`, `nan`, ± 0
- Při sčítání/odečítání se jedno číslo přenásobí, tak aby obě čísla měla stejný exponent
- Operace jsou pomalejší než odpovídající operace s `int`.
- Při operacích může dojít ke ztrátě přesnosti.

Reálná čísla v binární reprezentaci

FF UK
Department of Logic

- lze mít pouze konečný počet desetinných míst.
- některá čísla, která v desítkové soustavě mají konečný počet desetinných míst, mají ve dvojkové nekonečný.
- např. 0.1 v desítkové se v binární zapíše: $0.1\overline{1100}$
- místo přesných čísel se ukládají jejich aproximace
- přesně lze reprezentovat pouze čísla, která jsou součtem zlomků se jmenovateli mocniny 2.
- paradoxně pak například `0.01 != 0.1 ** 2`

Knihovna `math`

FF UK
Department of Logic

```
import math
```

`pi`	přibližná hodnota π
`e`	přibližná hodnota e
`sin(x)`	sinus x
`cos(x)`	cosinus x
`tan(x)`	tangens x
`asin(x)`	arcus sinus x (inverzní funkce k sin)
`acos(x)`	arcus cosisnus x
`atan(x)`	arcus tangens x
`log(x)`	přirozený logaritmus x (o základu e)
`log10(x)`	dekadický logaritmus x (o základu 10)
`exp(x)`	exponenciála (=`math.e**x`)
`ceil(x)`	Nejmenší celé číslo $\geq$ x
`floor(x)`	Největší celé číslo $\leq$ x

Datové konverze

FF UK
Department of Logic

implicitní

- `int = float ( x = 5.1 / 2 )`

explicitní

- `float(x), int(x), long(x)`
- `5.0 == float(5)`
- !!! `int(5.9999) == 5` !!!