

Počítání s čísly
(3. přednáška)



FF UK

Department of Logic

Jonathan L. Verner

- ▶ 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 ...

- ▶ 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

- ▶ **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

- ▶ 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, ...

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
>>> type(myInt)
<class 'float'>
```


Jaká čísla lze takto reprezentovat pomocí n-bitů ?

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

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>

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

- ▶ 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"`).

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

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

a, je-li $k < n$,

$$\left(\sum_{i=0}^n a_i 2^i \right) / 2^k = \sum_{i=k}^n a_i 2^{i-k} + x, \quad x \in [0, 1)$$

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

```
# 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)  
    return (bin_rep)  
  
def main():  
    x = input("Zadej kladne cele cislo:")  
    print(tobin(x))
```