Číselné typy

Jiří Zacpal

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

KMI/ZPP1 Základy programování v Pythonu 1

Číselné typy

- Celá čísla.
- Desetinná čísla:
  - čísla s dvojitou přesností,
  - čísla s pevnou řádkovou čárkou (Decimal),
  - zlomky (Fraction).
- Komplexní čísla.
- Logické hodnoty.

Celá čísla

Celá čísla

- objekty ze třídy int,
- celé číslo můžeme zadat jako libovolnou posloupnost desítkových číslic nezačínající nulou (s výjimkou 0),

```
>>> 123
123
```

- celá čísla mohou být libovolně velká,
- zápis velkých čísel můžeme zpřehlednit znakem _

```
>>> 123_456_789
123456789
```

- je-li číslo záporné, vložíme před něj znak -,

```
>>> -123
-123
```

Celá čísla v jiných soustavách

- kromě desítkové, můžeme celá čísla zapisovat ještě v:
- šestnáctkové soustavě
  - při zápisu čísla používám kromě číslic i písmena A-F,
  - zápis je uvozen znaky 0x

```
>>> 0xFF
255
>>> 0Xab1
2737
```

- osmičkové soustavě
  - při zápisu čísla používám kromě číslic 0-7,
  - zápis je uvozen znaky 0o

```
>>> 0o12
10
>>> 0064
52
```

Desetinná čísla

Desetinná čísla

- Desetinná čísla zapisujeme s desetinnou tečkou:

```
>>> 0.2
0.2
```

- Aritmetické operátory pracující také s desetinnými čísly:

```
>>> 0.1+0.1
0.2
>>> 0.5*0.2
0.1
>>>
```

- Pokud je jeden z operandů aritmetické operace celé číslo a druhý desetinné číslo, je celé číslo převedeno na desetinné číslo.

```
>>> 4*0.5
2.0
```

Desetinná čísla

- Aritmetický operátor dělení (/). Jeho výsledkem je vždy desetinné číslo:

```
>>> 4/2
2.0
```

- Desetinné číslo také dostaneme při použití operátoru mocniny se záporným mocnitelem.

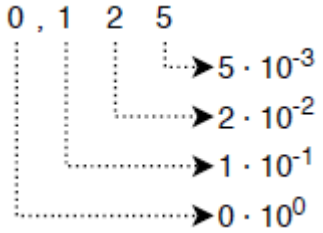
```
>>> 2**-3
0.125
```

- Při zadávání desetinných čísel lze použít i vědeckou notaci. Číslo ve tvaru $a*10^b$ zapíšeme jako aeb

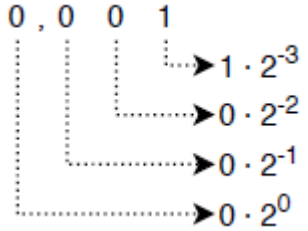
```
>>> 1e-2
0.01
>>> 1.23e2
123.0
>>> 100000000000000000000000000000000.0
1e+32
>>> 0.00001
1e-05
```

Problém s přesností

- Čísla s plovoucí řádovou čárkou v desítkové soustavě lze reprezentovat jako součet zlomků, kde jmenovatel obsahuje mocniny 10.
- V počítači jsou tato čísla reprezentována jako součet zlomků, kde jmenovatel obsahuje mocniny 2 (binární zlomky).
- Příklad:
  - 0, 125 pomocí zlomků:



a pomocí binárních zlomků:



- Problém této reprezentace je nepřesnost:
  - $\frac{1}{3}$ v desítkové soustavě nelze pomocí zlomků vyjádřit,
  - $\frac{1}{10}$ nelze vyjádřit pomocí binárních zlomků.
- Problém nastává při zobrazování těchto čísel, kdy je obvykle zobrazena zaokrouhlená hodnota, samotné číslo je ale uchováváno nezaokrouhlené.

Příklad problému s přesností

```
>>> print(0.1)
0.1
>>> print(1/10)
0.1
>>> print(0.1+0.1+0.1)
0.30000000000000004
>>> print(1/10+1/10+1/10)
0.30000000000000004
>>> print(1/10+1/10+1/10==0.3)
False
```

Řešení problému s přesností

- Tento problém je snadno řešitelný tak, že neporovnáváme čísla samotná, ale jejich absolutní rozdíl porovnáme s požadovanou přesností.

```
>>> print(abs(1/10+1/10+1/10-0.3)<0.001)
True
```

- Lepší je zavést si konstantu pro uložení přesnosti.

```
>>> PRECISSION = 1e-10
>>> print(abs(1/10+1/10+1/10-0.3)<PRECISSION)
True
```

Rozsah velikosti desetinný čísel

- Interpret používá pro práci s desetinnými čísly formát čísel s plovoucí desetinnou čárkou.
- Tento formát se skládá ze tři částí:
  - znaménka,
  - platných číslic,
  - exponentu.
- Počet platných číslic i exponent je omezen. Proto existuje největší desetinné číslo

  1.7976931348623157e+308

- a nejmenší kladné desetinné číslo

  5e-324.

Rozsah velikosti desetinný čísel

- Existence největšího desetinného čísla vede k tomu, že pokud by operace měla vrátit číslo vetší než největší možné, tak se vrátí nekonečno nebo dojde k chybě.

```
>>> 1e308 * 2
inf
>>> 2.0 ** 10000
OverflowError: (34, 'Result too large‘)
```

- Celá čísla horní omezení velikosti teoreticky nemají.

```
>>> 2 ** 10000
199506311688075838 4...
```

- Pokud by výsledek byl blíže nule než nejmenší kladné desetinné číslo, propadne se tento výsledek na nulu.

```
>>> 5e-324 / 2
0.0
>>> (-5e-324) / 2
-0.0
```

- Poznamenejme, že máme zápornou a kladnou nulu.

Hodnota nan

- Při počítání s nekonečnem můžeme narazit na zvláštní hodnotu nan.

```
>>> inf = 1e+308 * 2
>>> inf / inf
nan
```

- Hodnota nan je zkratka za Not A Number a je hodnotou neurčitých výrazů. Označme si hodnotu nan.

```
>>> nan = inf / inf
```

- Výsledky operací, kde aspoň jeden z operandu je nan, jsou opět nan.

```
>>> nan + 1
nan
>>> nan * nan
nan
```

- Hodnota nan se nerovná ničemu dokonce ani sama sobe.

```
>>> nan == 1
False
>>> nan == nan
False
```

- Přibližné výpočty s desetinnými čísly mají za důsledek to, že pokud bude sčítanec a o mnoho řádu vetší než sčítanec b, pak muže být součet a a b roven a.

```
>>> 10e20 + 1 == 10e20
True
```

Zlomky

Zlomky

- Pro práci se zlomky slouží knihovna fraction, je potřeba si ji tedy připojit příkazem:

```
from fractions import Fraction
```

- Zlomek vytvoříme:

```
Fraction(čitatel,jmenovatel)
```

- Příklad:

```
>>> z1=Fraction(1,3)
>>> z1
Fraction(1, 3)
>>> print(z1)
1/3
>>>
```

Práce se zlomky

- Se zlomky můžeme provádět základní aritmetické operace:

```
>>> z1 = Fraction(1,3)
>>> z2= Fraction(1,3)
>>> z3=z1+z2
>>> print(z3)
2/3
```

- Při operacích se zlomky dojde automaticky k převedení na základní tvar:

```
>>> z1=Fraction(1,4)
>>> z2=Fraction(1,4)
>>> z3=z1+z2
>>> print(z3)
1/2
```

„Přesná“ desetinná čísla

Decimal

- Pro práci se desetinnými čísly slouží knihovna decimal, je potřeba si ji tedy připojit příkazem:

```
from decimal import Decimal
```

- Desetinné číslo vytvoříme:

```
Decimal(číslo)
```

  - `číslo` může být zadáno jako celé číslo, desetinné číslo nebo řetězec

Příklad

- Decimal můžeme vytvořit pomocí desetinného čísla:

```
>>> c1=Decimal(0.1)
>>> print(c1)
0.1000000000000000055511151231257827021181583404541015625
```

- Decimal můžeme vytvořit pomocí celého čísla:

```
>>> c1=Decimal(1)/Decimal(10)
>>> print(c1)
0.1
>>> c1
```

- Decimal můžeme vytvořit pomocí řetězce :

```
Decimal('0.1')
>>> c1=Decimal('0.1')
>>> print(c1)
0.1
```

Příklad

- Počítání s Decimal je přesné:

```
>>> soucet=Decimal('0.1')+Decimal('0.1')+Decimal('0.1')
>>> print(soucet)
0.3
```

- Můžeme také nastavit počet desetinných míst (výchozí počet je 28):

```
>>> Decimal(1)/Decimal(7)
Decimal('0.14285714285714285714285714285714')
>>> getcontext().prec=2
>>> Decimal(1)/Decimal(7)
Decimal('0.14')
```

Převod mezi datovými typy

Implicitní přetypování

- Automatický převod jednoho datového typu v druhý.
- Dochází k převodu na „širší“ datový typ.
- Příklady:

```
>>> print (4+6)
10
>>> print (4+6.0)
10.0
>>> print(6/3)
2.0
>>> print(6//3)
2
>>> print(4*3)
12
>>> print(4+"2")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Explicitní přetypování

- Mezi datovými typy je možné provádět převody.
- V jazyce Python se pro převody používají funkce.
- Pro převod na celé číslo se používá funkce **int()**, pro převod na desetinné číslo se používá funkce **float()** a pro převod na řetězec se používá funkce **str()**.
- Příklady:

```
int(4.2) # prevod desetinného čísla na celé číslo 4
int("42") # prevod retezce čísla na celé číslo 42
float(42) # prevod celého čísla na desetinné číslo 42.0
float("42") # prevod retezce na desetinné číslo 42.0
str(4.2) # prevod desetinného čísla na retezec "4.2"
str(42) # prevod celého čísla na retezec "42"
```

Komplexní čísla

Komplexní čísla

- Komplexní čísla se zapisují jako součet reálné a imaginární části, přičemž imaginární část je označena písmenem j:

```
reální + imaginárníj
```

- Příklad:
- `>>> k1=4+2j`
- `>>> k2=2+5j`
- `>>> print(k1+k2)`
- `(6+7j)`
- `>>> print(k1*k2)`
- `(-2+24j)`

Modul math

Modul math

- Poskytuje velké množství matematických funkcí.
- Pro práci s tímto modulem si ho musíme připojit:

```
import math
```

- Funkce:
  - gonimoterické
    - např. `math.sin()`
  - odmocnina
    - `math.sqrt()`
  - logaritmické
    - `math.log()`
  - mnoho dalších
- Konstanty:
  - `math.pi`
  - `math.e`