Konstanty a desetinná čísla

Jiří Zacpal

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

KMI/ZPP1 Základy programování v Pythonu 1

Konstanty

Příklad

- Začneme převodem slova napsaného malými písmeny na velká písmena:

```
string = 'python'
upper_case = ''

for i in range(len(string)):
    upper_case += chr(ord(string[i]) - 32)

print(upper_case)
```

- V programu se vyskytuje záhadná hodnota 32. Lepší je si ji pojmenovat:

```
string = 'python'
letters_distance = 32
upper_case = ''

for i in range(len(string)):
    upper_case += chr(ord(string[i]) - letters_distance)

print(upper_case)
```

Příklad

- Proměnná letters_distance je výjimečná v tom, že se po běhu programu nemění její hodnota. Takovým proměnným budeme říkat konstanty a budeme je psát velkými písmeny.

```
string = 'python‘
LETTERS_DISTANCE = 32
upper_case = ''

for i in range(len(string)):
    upper_case += chr(ord(string[i]) - LETTERS_DISTANCE)

print(upper_case)
```

- Ještě můžeme učinit jeden krok k zlepšení čitelnosti programu, kterým je doplnit výpočet hodnoty konstanty.

```
LETTERS_DISTANCE = ord('a') - ord('A')
```

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

- Zavedeme si nový binární aritmetický operátor dělení (/). Jeho výsledkem je vždy desetinné číslo:

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

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

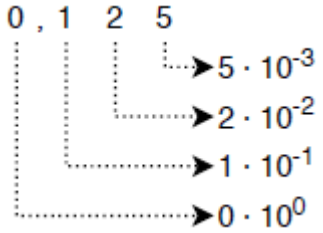
```
>>> 2**-3
0.125
```

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

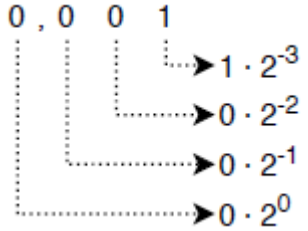
```
>>> 1e-2
0.01
>>> 1.23e2
123.0
>>> 100000000000000000000000000000000.0
1e+32
>>> 0.00001
1e-05
```

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. Jako například 1=1. 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
```

Úkol 1

- Pro dané přirozené číslo n vraťte součet:

$$\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n}$$

- Řešení:

```
n=10

soucet=0
for i in range(1,n+1):
    soucet+=1/i

print(f"Součet je {soucet:.4f}")
```

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"
```

Příklad

- Vytiskněte prvních n členů Fibonacciho posloupnosti. První člen je roven nule, druhý jedné a každý další je roven součtu dvou předchozích. Posloupnost tedy začíná 0; 1; 1; 2; 3; 5; 8:

```
n=input("Zadej n:")
if n.isdigit():
    n=int(n)
    f1=0
    print("1. Fibonacciho číslo: 0")
    f2=1
    print("2. Fibonacciho číslo: 1")
    for i in range(3,n+1):
        f3=f2+f1
        print(f"{i}. Fibonacciho číslo: {f3}")
        f1=f2
        f2=f3
else:
    print("Musíte zadat číslo!")
```

Příklad

- Pro dané přirozené číslo n > 1 vypočítejte přibližnou hodnotu zlatého řezu rovnou poměru a_n/a_{n-1}, kde a_i je i-tý člen Fibonnaciho posloupnosti.

```
n=input("Zadej n:")
if n.isdigit():
    n=int(n)
    f0=0
    f1=1
    for i in range(2,n+1):

        fn=f0+f1
        print(f"{i}. zlatý řez je: {fn/f1}")
        f0=f1
        f1=fn
else:
    print("Musíte zadat číslo!")
```

Příklad

- Je dána přesnost d, řekněme 10^{-10}. Předchozí program nám dává posloupnost stále se zlepšujících odhadu zlatého řezu. Upravte jej tak, aby výpočet skončil ve chvíli, kdy se sousední cleny odhadu zlatého rezu budou lišit o méně než d.

```
f0=0
f1=1
z1=1
z2=2
d=10e-10
while(abs(z2-z1)>d):
    fn=f0+f1
    zn=fn/f1
    f0=f1
    f1=fn
    z1=z2
    z2=zn
print(z2)
```

```
n=10

soucet=0
for i in range(1,n+1):
    soucet+=1/i

print(f"Součet je {soucet:.4f}")
n=10

soucet=0
for i in range(1,n+1):
    soucet+=1/i

print(f"Součet je {soucet:.4f}")
```

Úkol 2

- Spočítejte přibližně hodnotu čísla π , když víte, že čtvrtina je rovna:

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots$$

- Program bude brát přirozené číslo n a do výpočtu zahrnete jen n členů výpočtu.
- Řešení:

```
cislo=input("Zadej číslo n:")

n=int(cislo)
pi=0
for i in range(0,2*n):
    pi+=(-1)**i*1/(2*i+1)

pi*=4
print(f"Odhad pi je {pi:.4f}")
```

Úkol

1. Přibližně převeďte teplotu zadanou jako celé nebo desetinné číslo ve stupních Fahrenheita na stupně Celsia podle vztahu

$$C = \frac{5(f - 32)}{9}$$

- kde f je teplota ve stupních Fahrenheita a c ve stupních Celsia. Výslednou hodnotu uveďte jako desetinné číslo.

2. Použijte vztah zadaný v předchozím úkolu a napište program převádějící teplotu zadanou ve stupních Celsia na stupně Fahrenheita.