Мова програмування Python

Числа з фіксованою точністю. Клас Decimal

Особливості використання чисел з фіксованою точністю. Клас Decimal

Числа з *фіксованою точністю* – це числа типу Decimal, які при обчисленнях використовують фіксовану кількість знаків після коми. Тип Decimal – це спеціально розроблений клас (починаючи з версій Python 2.4).
Поняття “фіксована точність” означає, що з допомогою таких чисел можна зберігати значення яке буде завжди зберігати визначену кількість знаків після коми.
Наприклад, потрібно зберігати числа строго з кількістю 6 знаків після коми.
Клас Decimal реалізований в модулі decimal. Щоб використовувати можливості класу Decimal потрібно виконати команду

```
from decimal import Decimal
```

Як з допомогою класу Decimal задати потрібну фіксовану точність? Приклади

Щоб створити об’єкт класу Decimal використовується конструктор цього класу. Конструктор отримує рядок з числом, в якому вказується задана точність, наприклад

```
Decimal('0.002') # фіксована точність 3 знаки після коми
Decimal('0.23') # фіксована точність 2 знаки після коми
Decimal('0.00001') # фіксована точність 5 знаків після коми
```

Приклад необхідності застосування класу Decimal у програмах на Python

У прикладі продемонстровано необхідність написання програм з використанням класу Decimal для випадків, коли точність обчислення є вкрай важлива.

```
# Числа з фіксованою точністю. Клас Decimal - переваги застосування
# підключити клас Decimal з модуля decimal
from decimal import Decimal

# звичайне обчислення, існує похибка
a = 0.2+0.2+0.2-0.4 # a = 0.20000000000000007 - похибка (???)
print('a = ', a)

# обчислення з допомогою класу Decimal
b = Decimal('0.2')+Decimal('0.2')+Decimal('0.2')-Decimal('0.4')
print('b = ', b) # b = 0.2 - точний результат
```

```
a = 0.20000000000000007
b = 0.2
```

Спочатку створюється об’єкт (змінна) з іменем **a**, в яку записується сума

```
0.2+0.2+0.2-0.4
```

Потім значення цієї змінної виводиться на екран. Як видно з результату, результат обчислення змінної **a** містить похибку. Це пов’язане з тим, що пам’ять, що виділяється для чисел дійсного типу, є обмежена. Іншими словами, кількість бітів у представленні дійсних чисел є недостатньою. Потім створюється об’єкт з іменем **b**, в який записується сума з використанням класу Decimal

Після виведення значення **b** на екран, видно що значення змінної **b** представлене точно без похибки.

```
b = Decimal('0.2')+Decimal('0.2')+Decimal('0.2')-Decimal('0.4')
```

Приклад використання класу Decimal та функції str()

У попередньому прикладі конструктор класу Decimal отримував рядок з числом

```
Decimal('0.2')
```

в якому визначалось точність (1 знак після коми) та значення числа 0.2.
Можлива ситуація, коли потрібно передати безпосередньо число а не рядок. У цьому випадку зручно використовувати функцію str(), як показано нижче

```
# обчислення з допомогою класу Decimal
b = Decimal(str(0.2))+Decimal(str(0.2))+Decimal(str(0.2))-Decimal(str(0.4))
print('b = ', b) # b = 0.2
```

Функція str() отримує число і переводить його в рядок

```
x = str(0.2) # x = '0.2'
```

Використання фіксованої точності для чисел з різною точністю представлень. Приклад

Можлива ситуація, коли у виразі, що містить клас Decimal, представлено числа з різною точністю представлень. У цьому випадку точність результату автоматично встановлюється рівною точності числа з найбільшою точністю представлення.
Наприклад. При додаванні трьох чисел

```
c = Decimal('0.1')+Decimal('0.001')+Decimal(str(0.01)) # c = 0.111
```

автоматично встановлюється точність 3 знаки після коми, оскільки конструктор

```
Decimal('0.001')
```

визначає число 0.001 з найбільшою точністю представлення.

Створення об’єктів класу Decimal з дійсних чисел. Приклад

Для випадків, коли є в наявності дійсне число, можна створити об’єкт класу Decimal.
У цьому випадку використовується метод from_float() класу Decimal.

```
# Створення об'єкту типу Decimal, не завжди працює
# Випадок 1. Спотворена точність
# x1 =   2.47999999999999982236431605997495353221893310546875
x1 = Decimal.from_float(2.48)
print('x1 =',x1)

# Випадок 2. Фіксована точність
x2 = Decimal.from_float(2.5) # x2 = 2.5 - правильна точність
print('x2 =', x2)
```

Результат виконання вищенаведеного коду

```
x1 = 2.47999999999999982236431605997495353221893310546875
x2 = 2.5
```

Як видно з результату, не завжди вдається отримати фіксовану точність при використанні методу from_float().

Глобальне налаштування точності. Приклад

Бувають випадки, коли точність в програмі потрібно задати для усіх операцій поточного потоку керування. Це може бути, наприклад, представлення грошових сум з врахуванням копійок (2 знаки після коми).

```
# Глобальне задавання точності
# підключити клас Decimal
from decimal import Decimal

a = Decimal(5)/Decimal(13) # точність не задана
print('a = ', a) # a =   0.384615384615384615384615384615

# задавання точності 6 знаків після коми
import decimal
decimal.getcontext().prec=6
b = Decimal(5)/Decimal(13)
print('b = ', b) # b = 0.384615

c = Decimal(6)/Decimal(13)
print('c = ', c) # c = 0.461538
```

У вищенаведеному прикладі глобальна точність для класу Decimal задається з допомогою функції getcontext(), яка повертає об’єкт контексту в цьому модулі. Точність задається у поточному потоці керування.

Результат виконання програми

```
a = 0.384615384615384615384615384615
b = 0.384615
c = 0.461538
```