Алексей Васильев

Язык программирования Python

Киев 2019

Лекция 7. Числовые значения

- Основные числовые типы
- Числовые литералы
- Системы счисления
- Арифметические операторы
- Побитовые операторы
- Дроби
- Комплексные числа

Числовые типы

Числовые типы:

`int`	целые числа
`float`	действительные числа
`complex`	комплексные числа

Кроме числовых типов `int`, `float` и `complex`, можно использовать типы `Fraction` и `Decimal` для работы с рациональными дробями и целыми числами фиксированной точности

Литералы:

- в двоичной системе начинаются с префикса `0b` (или `0B`)
- в восьмеричной системе начинаются с префикса `0o` (или `0O`)
- в шестнадцатеричной системе начинаются с префикса `0x` (или `0X`)

В шестнадцатеричной системе: для обозначения чисел от `10` до `15` используются буквы от `A` до `F` (как большие, так и маленькие)

Целочисленные литералы, заданные в системах счисления, отличных от десятичной, интерпретируются как целые числа с соответствующим десятичным значением

Переменные, которым в качестве значения присваиваются двоичные, восьмеричные или шестнадцатеричные значения, могут использоваться в вычислениях как обычные переменные с целочисленным значением

Функции:

`bin()`	получить двоичный код
`oct()`	получить восьмеричный код
`hex()`	получить шестнадцатеричный код
`int()`	получить десятичное число

Если текстовое представление для числа содержит префикс системы счисления (`0b`, `0o` или `0x`), то соответствующее тестовое выражение можно передать аргументом функции `eval()`

В Python поддерживаются системы счисления с основанием от `2` до `36`. Если основание системы счисления больше `10`, в качестве "цифр" используются буквы от `A` (значение `10`) до `Z` (значение `35`). С помощью функции `int()` на основе текстового представления для числа в произвольной системе счисления (с основанием от `2` до `36`) можно получить число. Для систем счисления с основанием `2`, `8` или `16`, можно задавать литералы

Системы счисления

Программа (Nums.py)

```
print("Числа заданы в разных системах:")
# Число 19 в двоичной системе:
A=0b10011
print("A =",A)
# Число 93 в восьмеричной системе:
B=0o135
print("B =",B)
# Число 2603 в шестнадцатеричной системе:
C=0xA2B
print("C =",C)
print("C-A-B =",C-A-B)
# Обратное преобразование:
print("Обратное преобразование:")
A=int("10011",2)
print(bin(A),"=",A)
B=int("0o135",8)
print(oct(B),"=",B)
C=int("0xA2B",16)
print(hex(C),"=",C)
D=int("ABC",20)
print("ABC =",D)
```

```
Числа заданы в разных системах:
A = 19
B = 93
C = 2603
C-A-B = 2491
Обратное преобразование:
0b10011 = 19
0o135 = 93
0xa2b = 2603
ABC = 4232
```

Операторы

Арифметические операторы:

- Оператор сложения +. Результат выражения вида A+B с числовыми операндами A и B вычисляется как сумма значений этих операндов. Может использоваться как унарный оператор (например, выражение вида +A)
- Оператор вычитания –. Результат выражения вида A–B вычисляется как разность значений числовых операндов A и B. Может использоваться как унарный оператор (например, выражение вида –A)
- Оператор умножения *. Результат выражения вида A*B с числовыми операндами A и B вычисляется как произведение значений операндов A и B
- Оператор деления /. Результат выражения вида A/B вычисляется делением значения операнда A на значение операнда B
- Оператор целочисленного деления //. Результат выражения вида A//B вычисляется делением нацело значения операнда A на значение операнда B: вычисляется результат (обычного) деления значения операнда A на значение операнда B, а полученный результат округляется до ближайшего целочисленного значения (в направлении минус бесконечности)
- Оператор вычисления остатка от целочисленного деления % . Результатом выражения вида A%B является остаток от деления значения операнда A на значение операнда B: значение выражения вида A-(A//B)*B
- Оператор возведения в степень **. Результатом выражения вида A**B является число, которое получается возведением значения операнда A в степень, определяемую значением операнда B

Двоичные числа

Принцип бинарного кодирования чисел

Позиционное представление положительных чисел:

$\overline{a_n a_{n-1} \dots a_2 a_1 a_0} = 2^0 a_0 + 2^1 a_1 + 2^2 a_2 + \dots + 2^{n-1} a_{n-1} + 2^n a_n$ **параметры** $a_k = 0,1$**.**

Отрицательные числа:

Положительное число $x = \underbrace{0011010 \dots 011011}_{n}$

Инверсия кода $\sim x = \underbrace{1100101 \dots 100100}_{n}$

Сумма чисел $x + \sim x = \underbrace{1111111 \dots 111111}_{n}$

Сумма чисел $x + \sim x + 1 = 1\underbrace{0000000 \dots 000000}_{n}$

- **Чтобы получить код отрицательного числа, нужно взять код противоположного положительного числа, заменить в нем нули на единицы и наоборот, и прибавить к результату единицу**

- **Если код числа начинается с единицы, то это отрицательное число. Чтобы понять, какое это число, следует инвертировать код, прибавить к нему единицу, полученное значение перевести в десятичную систему и "дописать" знак "минус"**

Операторы

Побитовые операторы:

- Побитовое и `&`. Результат выражения вида `A&B` - число: код получается попарным сравнением значений битов в операндах `A` и `B`. Если оба бита равны `1`, в результате получаем единичный бит. Иначе в результате получаем нулевой бит
- Побитовое или `|`. Результат выражения вида `A|B` - число: код получается попарным сравнением значений битов в операндах `A` и `B`. Если хотя бы один бит равен `1`, в результате получаем единичный бит. Иначе в результате получаем нулевой бит
- Побитовое исключающее или `^`. Результат выражения вида `A^B` - число: код получается попарным сравнением значений битов в операндах `A` и `B`. Если значения битов разные, то в результате получаем единичный бит. Иначе в результате получаем нулевой бит
- Побитовый сдвиг влево `<<`. Результат выражения вида `A<<n` - число: код получается сдвигом влево кода числа `A` на количество позиций, определяемых значением операнда `n`. Младшие биты заполняются нулями. Старшие биты теряются
- Побитовый сдвиг вправо `>>`. Результат выражения вида `A>>n` - число: код получается сдвигом вправо кода числа `A` на количество позиций, определяемых значением операнда `n`. Старшие биты заполняются значением знакового бита. Младшие биты теряются
- Побитовое инвертирование `~`. Результат выражения вида `~A` - число: код получается из кода числа `A` заменой нулей на единицы и единиц на нули

Побитовые операции

Программа (BinOps.py)

```
# Переменные:
A=13
print("A =",A)
B=7
print("B =",B)
# Побитовые операции:
print("A&B =",A&B)
print("A|B =",A|B)
print("A^B =",A^B)
print("~A+1 =",~A+1)
# Побитовые сдвиги:
print("B>>1 =",B>>1)
print("B<<2 =",B<<2)
```

```
A = 13
B = 7
A&B = 5
A|B = 15
A^B = 10
~A+1 = -13
B>>1 = 3
B<<2 = 28
```

```
13 = 1101
 7 = 0111
```

```
& 1101
  0111
  0101 = 5
```

```
| 1101
  0111
  1111 = 15
```

```
^ 1101
  0111
  1010 = 10
```

```
A    000...001101
~A   111...110010
~A+1 111...110011
```

```
B    000...00111
B>>1 000...00011 = 3
```

```
B    000...00111
B>>2 000...11100 = 28
```

Дроби и числа

```
from fractions import Fraction
from decimal import Decimal
print("Дробные значения:")
A=Fraction(2,5)
print("A =",A)
B=Fraction(3,7)
print("B =",B)
C=A+B
print("A+B =",C)
print("Действительные числа:")
X=2/5
print("X =",X)
D=X+B
print("X+B =",D)
print("Числа заданной точности:")
A=Decimal('1.01')
print("A =",A)
B=Decimal('2.02')
print("B =",B)
C=A+B
print("A+B =",C)
print("1.01+2.02 =",1.01+2.02)
```

- **Реализация дробей:**
  тип `Fraction`
  из модуля `fractions`

- **Числа фиксированной точности:**
  тип `Decimal`
  из модуля `decimal`

Программа (FracDec.py)

```
Дробные значения:
A = 2/5
B = 3/7
A+B = 29/35
Действительные числа:
X = 0.4
X+B = 0.8285714285714285
Числа заданной точности:
A = 1.01
B = 2.02
A+B = 3.03
1.01+2.02 = 3.0300000000000002
```

Комплексные числа

Мнимая единица: $i^2 = -1$
Комплексное число: $z = x + iy$
Комплексно сопряженное: $z^* = x - iy$
Модуль: $|z| = \sqrt{zz^*} = x^2 + y^2$
Тригонометрическая форма: $z = |z| \exp(i\phi)$
Аргумент: $\cos(\phi) = \frac{x}{|z|}$ **и** $\sin(\phi) = \frac{y}{|z|}$
Формула Эйлера: $\exp(i\phi) = \cos(\phi) + i\sin(\phi)$

- **Мнимая часть определяется с помощью суффикса `j` или `J`**

Например: `3+4j`

- **Функция `complex`:**

Например: `complex(3,4)`

```
# Комплексные числа:
A=3+4j
print("A =",A)
print("Re(A) =",A.real)
print("Im(A) =",A.imag)
print("|A| =",abs(A))
B=-1+1j
print("B =",B)
C=complex(0,1)
print("C =",C)
# Арифметические операции:
print("A+B =",A+B)
print("A*C =",A*C)
print("A/B =",A/B)
```

Программа (Complex.py)

```
A = (3+4j)
Re(A) = 3.0
Im(A) = 4.0
|A| = 5.0
B = (-1+1j)
C = 1j
A+B = (2+5j)
A*C = (-4+3j)
A/B = (0.5-3.5j)
```

Задание - 1

Напишите программу, в которой пользователь вводит основание для системы счисления (2, 8 или 16) и число (в десятичной системе), а программа отображает это число в соответствующей системе счисления

Задание - 2

Напишите программу, в которой пользователь вводит целое число и номер бита, значение которого нужно определить в программе

Домашнее задание

[1] Напишите программу, в которой пользователь вводит два комплексных числа, а программа вычисляет сумму, разность, произведение и частное этих чисел. Среди полученных значений необходимо определить то, у которого наибольший и наименьший модуль

[2] Напишите программу, в которой пользователь вводит две рациональные дроби, а программа вычисляет сумму, произведение, разность и частное этих дробей, среди полученных значений находит наибольшее и наименьшее, и отображает результат вычислений