# Тема 2. Числа в Python. Арифметические операции. Математические функции. Модуль math. Функции случайных чисел

## 2.1. Числовые типы данных

В Python **3 типа числовых данных**. Для удобства оформим их в виде таблицы.

| Название типа | Описание |
|---|---|
| int | Целые числа. Позволяет работать как с отрицательными, так и с положительными целыми числами. |
| float | Вещественные числа. Позволяет работать с дробными и целыми числами, как с положительными, так и с отрицательными. |
| complex | Комплексные числа. |

## 2.2. Арифметические операции

Чтобы облегчить понимание операторов предположим, что у нас есть 2 переменные с заданными значениями, **a = 5**, **b = 2** и рассмотрим следующую таблицу:

| Оператор | Описание | Пример |
|---|---|---|
| + | Сложение, аналогично обычному математическому сложению. | a + b, в результате получим 7 |
| - | Вычитание, аналогично обычному математическому вычитанию. | a - b, в результате получим 3 |
| * | Умножение, аналогично обычному математическому умножению. | a * b, в результате получим 10 |
| / | Деление, аналогично обычному математическому делению. | a / b, в результате получим дробное число 2.5 |
| % | Вычисление остатка от деления. | a % b, в результате получим целое число 1. В Python данная операция доступна и для вещественных чисел |
| ** | Возведение в степень. | a ** b, в результате получим 25 |
| // | Деление нацело, находит целую часть от деления. | a // b, в результате получим 2 |

Обратите внимание на то, что в тех случаях, где оператор состоит из двух символов, пробел между ними не ставится.

При выполнении операций над числами разного типа, **возвращается число более сложного типа**. Так, если у нас есть вещественные и целые числа в выражении, то в результате получим вещественный результат.

**Пример:**

```
a = 2
b = 3.5
c = a * b
print(c)
```

```
7.0
```

Если в результате получается вещественное число, то при его выводе по умолчанию выведется до **16 знаков в дробной части**.

**Пример:**

```
print(1/3)
```

```
0.3333333333333333
```

При необходимости вывести число **с другой точностью** используется **форматированный вывод**.

**Пример:**

```
a = 1/3
print('%.5f'%a)
```

```
0.33333
```

Если все-таки нужно таким образом вывести не переменную, а непосредственно результат выражения, то выражение нужно брать в скобки.

**Пример:**

```
print('%.5f'%(1/3))
```

```
0.33333
```

При выполнении операции над вещественными числами следует учитывать **ограничение точности вычислений**.

**Пример:**

```
print(0.3 - 0.1 - 0.1 - 0.1)
```

В результате получим **-2.7755575615628914e-17**. А должны получить **0.0**. Если необходимо выполнять операции с фиксированной точностью, следует использовать модуль **decimal**:

```
from decimal import Decimal

print(Decimal('0.3')-Decimal('0.1')-Decimal('0.1')-Decimal('0.1'))
```

```
0.0
```

**Сокращенные операторы**

В Python, наряду с обычным присваиванием присутствует комбинированная операция: присваивание с арифметическим действием. Для удобства приведем все операторы в виде таблицы:

| Оператор | Описание | Пример |
|---|---|---|
| += | Добавляет значение правого операнда переменной, находящейся слева и присваивает результат переменной слева. | b+=a, аналогично b=b+a |
| −= | Вычитает из переменной, находящейся слева значение операнда, находящегося справа и присваивает результат переменной слева. | b−=a, аналогично b=b−a |
| *= | Умножает переменную, находящуюся слева на значение операнда, находящего справа, и присваивает результат переменной слева. | b*=a, аналогично b=b*a |
| /= | Делит переменную, находящуюся слева на значение операнда, находящего справа, и присваивает результат переменной слева. | b/=a, аналогично b=b/a |
| %= | Находит остаток от деления переменной, находящейся слева на значение операнда, находящегося справа и присваивает результат переменной слева. | b%=a, аналогично b=b%a |
| **= | Возводит переменную, находящуюся слева в степень равную значению операнда, находящегося справа и результат записывает в переменную слева. | b**=a, аналогично b=b**a |
| //= | Находит целую часть от деления переменной, находящейся слева на значение операнда, находящегося справа и записывает результат в переменную слева. | b//=a, аналогично b=b//a |

**Приоритет выполнения операций**

При написании математических выражений стоит внимательно следить за приоритетом операций, иначе получится неправильный результат. Изменять приоритет операций можно при помощи круглых скобок.

**Перечислим операторы в порядке убывания приоритета:**

1. −a, +a,**a – унарный минус, унарный плюс, возведение в степень. Если унарный оператор расположен слева от оператора **, то возведение в степень имеет больший приоритет, а если справа то меньший.
Например, выражение -10**-2 эквивалентно -(10**(-2));
2. *, %, /, // – умножение, остаток от деления, деление, деление с округлением вниз;
3. +, - – сложение, вычитание;
4. <<, >> – двоичный сдвиг;
5. & – двоичное «**И**»;
6. ^ – двоичное «**исключающее ИЛИ**»;
7. | – двоичное «**ИЛИ**»;
8. =, +=, -=, *=, /=, //=, %=, **= – присваивания.

## 2.3. Математические функции

В Python, как и в других языках программирования, имеются специальные встроенные математические функции, а также модуль **math**, используемый в математических вычислениях.

**Встроенные математические функции и методы**

Первые 5 функций уже рассматривались ранее. Здесь они даны для полноты содержания.

- **int(***объект* [, *система счисления*]**)** – преобразует объект в целое число. В необязательном параметре можно указать систему счисления преобразуемого числа (по умолчанию 10).

  Примеры:

| Код | Вывод |
|---|---|
| `int(1.5)` | `1` |
| `int('25', 10)` | `25` |
| `int('0o25', 8)` | `21` |
| `int('0xA', 16)` | `10` |

- **float(***число или строка***)** – преобразует целое число или строку в вещественное значение.

| Код | Вывод |
|---|---|
| `float(1)` | `1.0` |
| `float('2')` | `2.0` |

- **bin(***число***)** – преобразует десятичное число в двоичное. В результате получаем строковое значение.

| Код | Вывод |
|---|---|
| `bin(15)` | `0b1111` |

- **oct(***число***)** – преобразует десятичное число в восьмеричное. В результате получаем строковое значение.

| Код | Вывод |
|---|---|
| `oct(15)` | `0o17` |

- **hex(***число***)** – преобразует десятичное число в шестнадцатеричное. В результате получаем строковое значение.

| Код | Вывод |
|---|---|
| `hex(15)` | `0xf` |

- **round(***число* [, *количество знаков после точки*]) – для чисел с дробной частью меньше 0.5 возвращает число, округленное до ближайшего меньшего целого. Для чисел с дробной частью больше 0.5 возвращает число, округленное до ближайшего большего целого. Если дробная часть равна 0.5, то округление происходит до ближайшего четного числа.

  **Примеры:**

| Код | Вывод |
|---|---|
| `round(0.4)` | `0` |
| `round(1.55)` | `2` |

| | |
|---|---|
| `round(2.5)` | `2` |
| `round(1.5)` | `2` |

**Во втором** необязательном **параметре** указывается желаемое количество знаков после запятой (целое неотрицательное число). Имеет смысл указывать только в дробных числах. В целых не будет никакого изменения.

| Код | Вывод |
|---|---|
| `round(0.2567, 3)` | `0.257` |
| `a = 12323`<br>`round(a, 1)` | `12323` |

Также дополнительным параметром может быть целое отрицательное число. Обнуляет все разряды до указанного номера (отчет слева направо). Работает только с целыми числами. При использовании с дробными числами обнуляет число.

| Код | Вывод |
|---|---|
| `a = 12323`<br>`round(a, -1)` | `12320` |
| `a = 12323`<br>`round(a, -2)` | `12300` |
| `a = 1.2323`<br>`round(a, -1)` | `0.0` |

- **abs**(*число*) – возвращает абсолютное значение числа (модуль).

| Код | Вывод |
|---|---|
| `abs(5)` | `5` |
| `abs(-5)` | `5` |

- **pow**(*число*, *степень*) – возводит число в степень.

| Код | Вывод |
|---|---|
| `pow(5, 2)` | `25` |

- **max**(*список чисел через запятую*) – находит максимальное значение из списка.

| Код | Вывод |
|---|---|
| `max(5, 7, 9, 2, 4)` | `9` |

- **min**(*список чисел через запятую*) – находит минимальное значение из списка.

| Код | Вывод |
|---|---|
| `min(5, 7, 9, 2, 4)` | `2` |

- **sum**(*последовательность*) – возвращает сумму значений элементов последовательности.

| Код | Вывод |
|---|---|
| `sum([1, 2, 3, 4, 5])` | `15` |

- **divmod**(*a*, *b*) – возвращает кортеж из двух значений (целая часть от деления **a** на **b** и остаток от деления **a** на **b**).

| Код | Вывод |
|---|---|
| `divmod(11, 2)` | `(5, 1)` |

- **is_integer() –** возвращает True, если заданное вещественное число не содержит дробной части.

| Код | Вывод |
|---|---|
| `(2.0).is_integer()` | `True` |
| `(2.5).is_integer()` | `False` |

## 2.4. Модуль math

Модуль **math** содержит дополнительные функции для работы с числами. Использование функций модуля подразумевает его предварительное подключение.

Подключение модуля осуществляется следующим образом:

```
import math
```

Для удобства основные функции представим в виде таблицы:

| Название | Описание |
|---|---|
| math.ceil(a) | Округление до ближайшего большего целого числа. |
| math.cmp(a,b) | -1 если a < b, 0 если a== b, или 1, если a > b |
| math.e | e = 2,718281... |
| math.exp(x) | $e^x$ |
| math.fabs(a) | Модуль числа (в отличие от встроенной функции abs, fabs возвращает всегда вещественное число). |
| math.factorial(a) | Факториал числа. |
| math.floor(a) | Округление вниз, до ближайшего меньшего целого. |
| math.fmod(a,b) | Остаток от деления a на b. В отличие от встроенной функции нахождения остатка, всегда возвращает вещественное число. |
| math.fsum(a1,a2,…,an) | Возвращает сумму чисел из списка. |
| math.log(a,[основание]) | Натуральный логарифм. Если указать доп. параметр основание, то вычисляет соответствующий логарифм. |
| math.log10(a) | Десятичный логарифм. |
| math.log2(a) | Логарифм по основанию 2. |
| math.max(a,b,c,…) | Находит максимальное среди чисел. |
| math.min(a,b,c,…) | Находит минимальное среди чисел. |
| math.pow(a,b) | Возведение в степень ($a^b$). |
| math.round(a) | Округление (требует уточнения). |
| math.sqrt(a) | Извлечение квадратного корня. |

Список всех методов модуля **math** можно получить одним из следующих способов:

### Способ 1 – список команд

```
import math
print(dir(math))
```

### Способ 2 – подробная справка

```
import math
print(help(math))
```

### Тригонометрические функции

Тригонометрические функции находятся в том же модуле **math**, но выведем их для удобства в отдельную таблицу.

| Функция | Описание |
|---|---|
| math.cos(X) | Косинус X (X указывается в радианах). |
| math.sin(X) | Синус X (X указывается в радианах). |
| math.acos(X) | Арккосинус X (в радианах). |
| math.asin(X) | Арксинус X (в радианах). |
| math.tan(X) | Тангенс X (X указывается в радианах). |
| math.atan(X) | Арктангенс X (в радианах). |
| math.degrees(X) | Конвертирует радианы в градусы. |
| math.radians(X) | Конвертирует градусы в радианы. |
| math.cosh(X) | Вычисляет гиперболический косинус. |
| math.sinh(X) | Вычисляет гиперболический синус. |
| math.tanh(X) | Вычисляет гиперболический тангенс. |
| math.acosh(X) | Вычисляет обратный гиперболический косинус. |
| math.asinh(X) | Вычисляет обратный гиперболический синус. |
| math.atanh(X) | Вычисляет обратный гиперболический тангенс. |
| math.pi | π = 3,1415926... |

## 2.5. Комплексные числа. Модуль cmath

Выше было сказано, что Python может работать с комплексными числами. Как известно, корень из отрицательного числа есть комплексное число.

Но если использовать для извлечения корня функцию **sqrt()** библиотеки **math**, комплексное число не получится.

```
import math
print(math.sqrt(-1))
```

В результате получим **ошибку**!

Для работы с комплексными числами используется специальный модуль **cmath.**

```
import cmath
print(cmath.sqrt(-1))
```

В результате получится комплексное число **1j**.

Остальные функции данного модуля предлагаем изучить самостоятельно.

## 2.6. Генерация случайных чисел. Модуль random

Случайные числа чаще всего используются в том случае, когда необходимо протестировать программу, перебирая различные числа, например, в симуляторах, играх.

Для работы со случайными числами необходимо импортировать модуль **random**:

```
import random
```

- **random.random() –** возвращает случайное вещественное число от 0 до 1.

| Код | Пример вывода |
|---|---|
| `import random`<br>`print(random.random())` | `0.5375882543375915` |

- **random.uniform(***начало*, *конец***)**– возвращает случайное число с плавающей точкой из диапазона, указанного в скобках. Начало и конец включительно.

| Код | Пример вывода |
|---|---|
| `import random`<br>`print(random.uniform(10, 20))` | `14.448751080168986` |

- **random.randint**(*начало*, *конец*) – возвращает целое случайное число из диапазона, указанного в скобках. Начало и конец включительно.

| Код | Пример вывода |
|---|---|
| `import random`<br>`print(random.randint(10, 20))` | `14` |

- **random.choince**(*последовательность*) – возвращает случайный элемент из любой последовательности (строки, списка, кортежа):

| Код | Пример вывода |
|---|---|
| `import random`<br>`print(random.choice('Python'))` | `y` |
| `print(random.choice([1, 2, 'a', 'b']))` | `2` |
| `print(random.choice([1, 2, 'a', 'b']))` | `a` |
| `print(random.choice((1, 3, 5, 'a')))` | `5` |

- **random.randrange**(*начало*, *конец, шаг*) – возвращает случайно выбранное число из последовательности. Начало включительно, конец не включительно.

| Код | Пример вывода |
|---|---|
| `import random`<br>`print(random.randrange(10, 20, 2))` | `12` |

В данном случае будут выбираться случайные числа, начиная от 10, с шагом в 2, то есть числа 10, 12, 14, 16, 18.

Мы рассмотрели основные методы. Справку по модулю **random** можно получить непосредственно в программе следующим образом:

```
import random
print(help('random'))
```