**Числа**

Язык **Python 3** поддерживает следующие числовые тины:

* **int**- целые числа. Размер числа ограничен лишь объемом оперативной памяти;
* **float**- вещественные числа;
* **comp1ex**- комплексные числа.

Операции над числами разных типов возвращают число, имеющее более сложный тип из типов, участвующих в операции. Целые числа имеют самый простой тип, далее идут вещественные числа и самый сложный тип- комплексные числа. Таким образом, если в операции участвуют целое и вещественное числа, то целое число будет автоматически преобразовано в вещественное число, а затем произведена операция над вещественными числами. Результатом этой операции будет вещественное число.

Создать объект целочисленного типа можно обычным способом:



Кроме того, можно указать число в двоичной, восьмеричной или шестнадцатеричной форме. Такие числа будут автоматически преобразованы в десятичные целые числа. Двоичные числа начинаются с комбинации символов 0b (или 0B) и содержат цифры 0 или 1:



Восьмеричные числа начинаются с нуля и следующей за ним латинской буквы о (регистр не имеет значения) и содержат цифры от 0 до 7:



Шестнадцатеричные числа начинаются с комбинации символов 0x (или 0X) и могут содержать цифры от 0 до 9 и буквы от А до F (регистр букв не имеет значения):



Вещественное число может содержать точку и (или) быть представлено в экспоненциальной форме с буквой Е (регистр не имеет значения):



Комплексные числа записываются в формате:

**<Вещественная часть>+<Мнимая часть>J**

Примеры комплексных чисел:



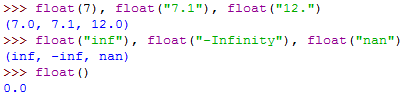
**6.1 Встроенные функции для работы с числами**

Для работы с числами предназначены следующие встроенные функции:

* **int ( [<Объект> [, <Система счисления>]])** - преобразует объект в целое число. Во втором параметре можно указать систему счисления (значение по умолчанию 10). Пример:



* **float ( [<Число или строка>])** - преобразует целое число или строку в вещественное число:



* **bin ( <Число>)** - преобразует десятичное число в двоичное. Возвращает строковое представление числа. Пример:



* **oct (<Число>)** - преобразует десятичное число в восьмеричное. Возвращает строковое представление числа. Пример:

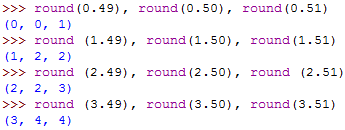


* **hex (<Число>)** - преобразует десятичное число в шестнадцатеричное. Возвращает строковое представление числа. Пример:



* **round** ( <Число> [, <Количество знаков после точки>] ) -возвращает число, округленное до ближайшего меньшего· целого для чисел ·с дробной частью меньше 0.5, или значение, округленное до ближайшего большего целого для чисел с дробной частью больше 0.5. Если дробная часть равна 0.5, то округление производится до ближайшего четного числа.

Пример:



Во втором параметре можно указать количество знаков в дробной части. Если параметр не указан, то используется значение 0:



* **abs (<Число>)** - возвращает абсолютное значение:



* **pow ( <Число>, <Степень> [, <Остаток от деления>] )** - возводит <**Число**> в <**Степень**>:



Если указан третий параметр, то возвращается остаток от деления:



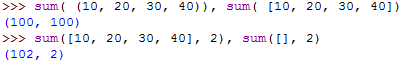
* **max (<Список чисел через запятую>)** - максимальное значение из списка:



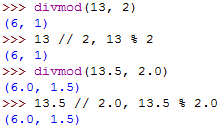
* **min (<Список чисел через запятую>)** -минимальное значение из списка:



* **sum (<Последовательность> [, <Начальное значение>] )** - возвращает сумму значений элементов последовательности (например, списка, кортежа) плюс <**Начальное значение**>. Если второй параметр не указан, то значение параметра равно 0. Если последовательность пустая, то возвращается значение второго параметра. Примеры:



* **divmod(x, у)** – возвращает кортеж из двух значений (х //у, х% у):



**6.2 Модуль math. Математические функции**

Модуль **math** предоставляет дополнительные функции для работы с числами, а также стандартные константы. Прежде чем использовать модуль, необходимо подключить его с помощью инструкции:

**import math**

**ПРИМЕЧАНИЕ**

Для работы с комплексными числами необходимо использовать модуль **cmath**.

Модуль **math** предоставляет следующие стандартные константы:

* **pi** - возвращает число π:



* е - возвращает значение константы е:



Перечислим основные функции для работы с числами:

* **sin(), cos(), tan()** — стандартные тригонометрические функции (синус, косинус, тангенс). Значение указывается в радианах;
* **asin(), acos(), atan()** — обратные тригонометрические функции (арксинус, арккосинус, арктангенс). Значение возвращается в радианах;
* **degrees()** — преобразует радианы в градусы:



* **radians()** — преобразует градусы в радианы:



* **exp()** — экспонента;
* **log (<Число> [, <База>])** — логарифм по заданной базе. Если база не указана, вычисляется натуральный логарифм (по базе е);
* **log10() —** десятичный логарифм;
* **log2() —** логарифм по базе 2;
* **sqrt() —** квадратный корень:



* **ceil()** — значение, округленное до ближайшего большего целого:



* **floor**() — значение, округленное до ближайшего меньшего целого:



* **pow(<Число>, <Степень>)** — возводит <**Число**> в <**Степень**>:



* **fabs**() — абсолютное значение:



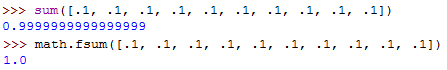
* **fmod**() — остаток от деления:



* **factorial**() — факториал числа:



* **fsum**(<список чисел>) — возвращает точную сумму чисел из заданного списка:



***Примечание***

В этом разделе мы рассмотрели только основные функции. Чтобы получить полный список функций, обращайтесь к документации по модулю **math**.

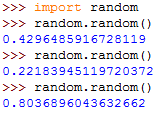
**6.3 Модуль *random*.** **Генерация случайных чисел**

Модуль **random** позволяет генерировать случайные числа. Прежде чем использовать модуль, необходимо подключить его с помощью инструкции:

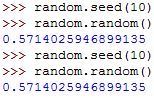
**import random**

Перечислим основные его функции:

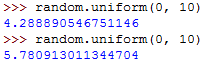
* **random**() — возвращает псевдослучайное число от 0.0 до 1.0:



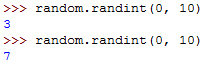
* **seed([<Параметр>] [, version=2])** — настраивает генератор случайных чисел на новую последовательность. Если первый параметр не указан, в качестве базы для случайных чисел будет использовано системное время. Если значение первого параметра будет одинаковым, то генерируется одинаковое число:



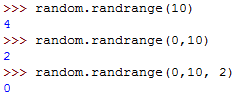
* **uniform(<начало>, <конец>)** — возвращает псевдослучайное вещественное число в диапазоне от < **начало** > до <**конец**>:



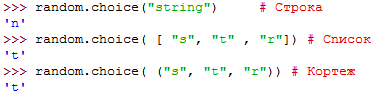
* **randint (<Начало>, <Конец>)** — возвращает псевдослучайное целое число в диапазоне от <**Начало**> до <**Конец**>:



* **randrange ([<Начало>, ]<Конец>[, <Шаг>])** — возвращает случайный элемент из числовой последовательности. Параметры аналогичны параметрам функции **range()**. Можно сказать, что функция **randrange** «за кадром» создает диапазон, из которого и будут выбираться возвращаемые случайные числа:

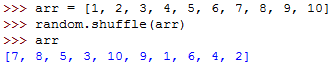


* **choice(<последовательность>)** — возвращает случайный элемент из заданной последовательности (строки, списка, кортежа):

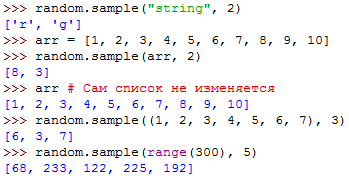


* **shuffle(<Сnисок> [, <Число от 0.0 до 1.0> ])** - перемешивает элементы списка случайным образом. Функция перемешивает сам список и ничего не возвращает. Если второй параметр не указан, то используется значение, возвращаемое функцией **random()**.

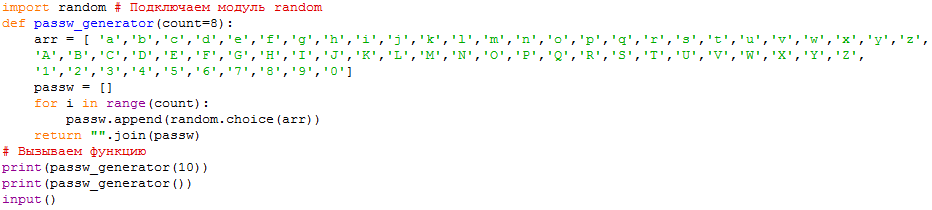
Пример:



* **sаmрlе(<Последовательность>, <Количество элементов>)** - возвращает список из указанного количества элементов. В этот список попадут элементы из последовательности, выбранные случайным образом. В качестве последовательности можно указать любые объекты, поддерживающие итерации. Примеры:



Для примера создадим генератор паролей произвольной длины. Для этого добавляем в список **arr** все разрешенные символы, а далее в цикле получаем случайный элемент с помощью функции **choice()**. По умолчанию будет выдаваться пароль из 8 символов.



Результат работы программы:

