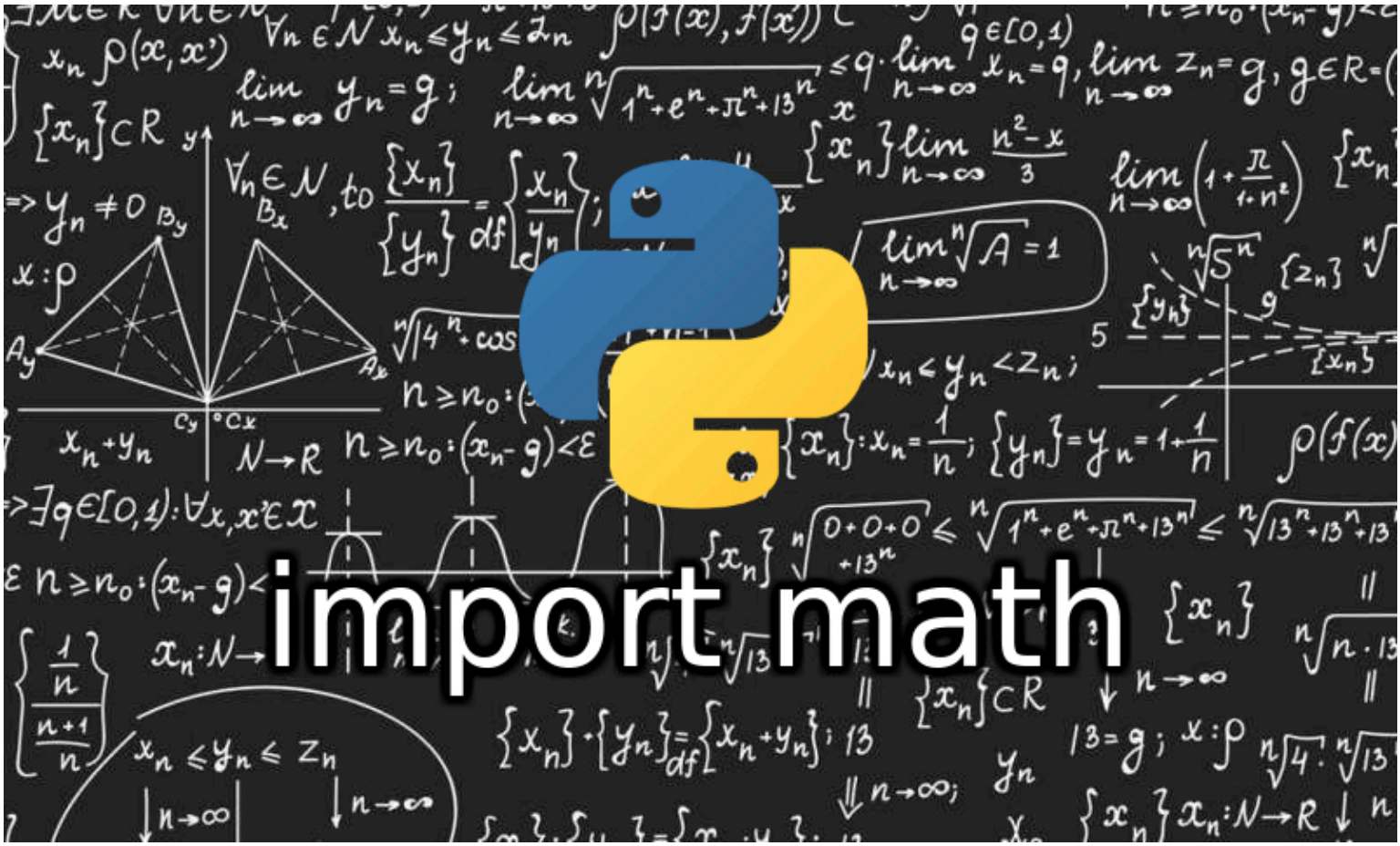
Модуль Math — математика в Python на примерах (Полный Обзор)



Библиотека **Math** в Python обеспечивает доступ к некоторым популярным математическим функциям и константам, которые можно использовать в коде для более сложных математических вычислений. Библиотека является встроенным модулем Python, поэтому никакой дополнительной установки через pip делать не нужно. В данной статье будут даны примеры часто используемых функций и констант библиотеки Math в Python.

Содержание статьи

- Специальные константы библиотеки math
- Число Пи из библиотеки math
- Число Эйлера из библиотеки math
- Экспонента и логарифм библиотеки math
- Функция экспоненты exp() в Python
- Функция логарифма log() в Python

Функция log10() в Python

Функция log2() в Python

Функция log(x, y) в Python

Функция log1p(x) в Python

Арифметические функции в Python

Тригонометрические функции в Python

Конвертация типов числа в Python

Специальные константы библиотеки math

В библиотеке Math в Python есть две важные математические константы.

Число Пи из библиотеки math

Первой важной математической константой является число Пи (π). Оно обозначает отношение длины окружности к диаметру, его значение 3,141592653589793. Чтобы получить к нему доступ, сначала импортируем библиотеку math следующим образом:

```python
import math
```

Затем можно получить доступ к константе, вызывая `pi`:

```python
math.pi
```

Вывод

```shell
3.141592653589793
```

Данную константу можно использовать для вычисления площади или длины окружности. Далее представлен пример простого кода, с помощью которого это можно сделать:

```python
import math

radius = 2
print('Площадь окружности с радиусом 2 равна:', math.pi * (radius ** 2))
```

Вывод

```shell
Площадь окружности с радиусом 2 равна: 12.566370614359172
```

Мы возвели радиус во вторую степень и умножили значение на число Пи, как и следовало сделать в соответствии с формулой πr^2.

Число Эйлера из библиотеки math

Число Эйлера (e) является основанием натурального логарифма. Оно также является частью **библиотеки Math** в Python. Получить доступ к числу можно следующим образом:

```python
math.e
```

Вывод

```shell
2.718281828459045
```

В следующем примере представлено, как можно использовать вышеуказанную константу:

```python
import math

print((math.e + 6 / 2) * 4.32)
```

Вывод

```shell

```

```
24.702977498943074
```

Экспонента и логарифм библиотеки math

В данном разделе рассмотрим функции библиотеки Math в Python, которые используются для нахождения экспоненты и логарифмов.

Функция экспоненты exp() в Python

Библиотека Math в Python поставляется с функцией `exp()`, которую можно использовать для вычисления значения e. К примеру, e^x — экспонента от x. Значение e равно `2.718281828459045`.

Метод может быть использован со следующим синтаксисом:

Python
```
math.exp(x)
```

Параметр x может быть положительным или отрицательным числом. Если x не число, метод возвращает ошибку. Рассмотрим пример использования данного метода:

Python
```
import math

# Инициализация значений
an_int = 6
a_neg_int = -8
a_float = 2.00

# Передача значений методу exp() и вывод
print(math.exp(an_int))
print(math.exp(a_neg_int))
print(math.exp(a_float))
```

Вывод

Shell
```
403.4287934927351
0.00033546262790251185
7.38905609893065
```

Мы объявили три переменные и присвоили им значения с различными числовыми типами данных. Мы передали значения методу `exp()` для вычисления их экспоненты.

Мы также можем применить данный метод для встроенных констант, что продемонстрировано ниже:

Python
```
import math

print(math.exp(math.e))

```

```
print(math.exp(math.pi))
```

Вывод

```
15.154262241479262
23.140692632779267
```

При передаче не числового значения методу будет сгенерирована ошибка TypeError, как показано далее:

```
import math

print(math.exp("20"))
```

Вывод

```
Traceback (most recent call last):
  File "C:/Users/admin/mathe.py", line 3, in <module>
    print (math.exp("20"))
TypeError: a float is required
```

Как видно из примера выше, генерируется ошибка `TypeError`.

Функция логарифма log() в Python

Функция `log()` возвращает логарифм определенного числа. Натуральный логарифм вычисляется относительно основания e. В следующем примере показано использование функции логарифма:

```
import math

print("math.log(10.43):", math.log(10.43))
print("math.log(20):", math.log(20))
print("math.log(math.pi):", math.log(math.pi))
```

В скрипте выше методу передаются числовые значения с различными типами данных. Также рассчитывается натуральный логарифм константы `pi`. Вывод следующий:

```
math.log(10.43): 2.344686269012681
math.log(20): 2.995732273553991
math.log(math.pi): 1.1447298858494002
```

Функция log10() в Python

Метод `log10()` возвращает логарифм по основанию 10 определенного числа. К примеру:

```python
import math

# Возвращает log10 числа 50
print("log10 числа 50 равен:", math.log10(50))
```

Вывод

```shell
log10 числа 50 равен: 1.6989700043360187
```

Функция log2() в Python

Функция `log2()` возвращает логарифм определенного числа по основанию 2. К примеру:

```python
import math

# Возвращает log2 числа 16
print("log2 числа 16 равен:", math.log2(16))
```

Вывод

```shell
log2 числа 16 равен: 4.0
```

Функция log(x, y) в Python

Функция `log(x, y)` возвращает логарифм числа x по основанию y. К примеру:

```python
import math

# Возвращает логарифм 3,4
print("Логарифм 3 по основанию 4 равен:", math.log(3, 4))
```

Вывод

```shell
Логарифм 3 по основанию 4 равен: 0.6309297535714574
```

Функция log1p(x) в Python

Функция `log1p(x)` рассчитывает логарифм(1+x), как представлено ниже:

Python

```python
import math

print("Значение логарифма(1+x) от 10 равно:", math.log1p(10))
```

Вывод

Shell

```
Значение логарифма(1+x) от 10 равно: 2.3978952727983707
```

Арифметические функции в Python

Арифметические функции используются для представления чисел в различных формах и осуществления над ними математических операций. Далее представлен перечень самых популярных арифметических функций:

`ceil()`: округление определенного числа вверх;

`fabs()`: возвращает модуль (абсолютное значение) указанного числа;

`floor()`: округление определенного числа вниз;

`gcd(a, b)`: получение наибольшего общего делителя чисел a и b;

`fsum(iterable)`: возвращает сумму всех элементов итерируемого объекта;

`expm1()`: возвращает (e^x)-1;

`exp(x)-1`: когда значение x слишком мало, вычисление `exp(x)-1` может привести к значительной потери в точности. `expm1(x)` может вернуть вывод с полной точностью.

В следующем примере показано использование перечисленных выше функций:

Python

```python
import math

num = -4.28
a = 14
b = 8

num_list = [10, 8.25, 75, 7.04, -86.23, -6.43, 8.4]
x = 1e-4 # Малое значение x

print('Число:', num)
print('Округление числа вниз:', math.floor(num))
print('Округление числа вверх:', math.ceil(num))
print('Модуль числа:', math.fabs(num))
print('Наибольший общий делитель a и b: ' + str(math.gcd(a, b)))
print('Сумма элементов списка: ' + str(math.fsum(num_list)))
print('e^x (при использовании функции exp()) равно:', math.exp(x)-1)
print('e^x (при использовании функции expml()) равно:', math.expm1(x))
```

Вывод

```Python
Число: -4.28
Округление числа вниз: -5
Округление числа вверх: -4
Модуль числа: 4.28
Наибольший общий делитель a и b: 2
Сумма элементов списка: 16.029999999999998
e^x (при использовании функции exp()) равно: 0.00010000050001667141
e^x (при использовании функции expm1()) равно: 0.00010000500016667084
```

К числу других математических функций относятся:

> `pow()`: принимает два вещественных аргумента, возводит первый аргумент в степень, значением которой является второй аргумент, после чего возвращает результат. К примеру, `pow(2, 2)` эквивалентно выражению `2 ** 2`;
>
> `sqrt()`: возвращает **квадратный корень** определенного числа.

Примеры данных методов представлены ниже:

Возведение в степень

```Python
math.pow(3, 4)
```

Вывод

```Shell
81.0
```

Квадратный корень

```Python
math.sqrt(81)
```

Вывод

```Shell
9.0
```

Тригонометрические функции в Python

Модуль **math** в Python поддерживает все тригонометрические функции. Самые популярные представлены ниже:

> `sin(a)`: Возвращает синус "a" в радианах;

`cos(a)`: Возвращает косинус "a" в радианах;

`tan(a)`: Возвращает тангенс "a" в радианах;

`asin(a)`: Возвращает инвертированный синус. Аналогичным образом работают "atan" и "acos";

`degrees(a)`: Конвертирует угол "a" из радиан в градусы;

`radians(a)`: Конвертирует угол "a" из градусов в радианы.

Рассмотрим следующий пример:

```python
import math

angle_In_Degrees = 62
angle_In_Radians = math.radians(angle_In_Degrees)

print('Значение угла:', angle_In_Radians)
print('sin(x) равен:', math.sin(angle_In_Radians))
print('tan(x) равен:', math.tan(angle_In_Radians))
print('cos(x) равен:', math.cos(angle_In_Radians))
```

Вывод

```shell
Значение угла: 1.0821041362364843
sin(x) равен: 0.8829475928589269
tan(x) равен: 1.8807264653463318
cos(x) равен: 0.46947156278589086
```

Обратите внимание, что вначале мы конвертировали значение угла из градусов в радианы для осуществления дальнейших операций.

Конвертация типов числа в Python

Python может конвертировать начальный тип числа в другой указанный тип. Данный процесс называется «преобразованием». Python может внутренне конвертировать число одного типа в другой, когда в выражении присутствуют смешанные значения. Такой случай продемонстрирован в следующем примере:

```python
3 + 5.1
```

Вывод

```shell
8.1
```

В вышеприведенном примере целое число 3 было преобразовано в вещественное число 3.0 с плавающей точкой. Результатом сложения также является число с плавающей точкой (или запятой).

Однако иногда вам необходимо явно привести число из одного типа в другой, чтобы удовлетворить требования параметра функции или оператора. Это можно сделать с помощью различных встроенных функций Python.

Например, чтобы преобразовать целое число в число с плавающей точкой, мы должны вызвать функцию `float()`, как показано ниже:

```python
a = 12
b = float(a)
print(b)
```

Вывод

```shell
12.0
```

Целое число типа `integer` было преобразовано в вещественное число типа `float`. `float` также можно конвертировать в `integer` следующим образом:

```python
a = 12.65
b = int(a)
print(b)
```

Вывод

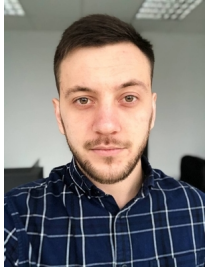
```shell
12
```

Вещественное число было преобразовано в целое через удаление дробной части и сохранение базового числа. Обратите внимание, что при конвертации значения в `int` подобным образом число будет усекаться, а не округляться вверх.

Заключение

Библиотека Math предоставляет функции и константы, которые можно использовать для выполнения арифметических и тригонометрических операций в Python. Библиотека изначально встроена в Python, поэтому дополнительную установку перед использованием делать не требуется. Для получения дополнительной информации можете просмотреть официальную документацию.

Vasile Buldumac



Являюсь администратором нескольких порталов по обучению языков программирования Python, Golang и Kotlin. В составе небольшой команды единомышленников, мы занимаемся популяризацией языков программирования на русскоязычную аудиторию. Большая часть статей была адаптирована нами на русский язык и распространяется бесплатно.

E-mail: vasile.buldumac@ati.utm.md

Образование

Universitatea Tehnică a Moldovei (*utm.md*)

2014 — 2018 Технический Университет Молдовы, ИТ-Инженер. Тема дипломной работы «*Автоматизация покупки и продажи криптовалюты используя технический анализ*»

2018 — 2020 Технический Университет Молдовы, Магистр, Магистерская диссертация «*Идентификация человека в киберпространстве по фотографии лица*»