
Яндекс

Яндекс

Приоритет операций. Простейшие функции

Приоритет операций

Приоритет операций

от наивысшего (операция выполняется первой) до наинизшего:

1. Возведение в степень (**).
2. Унарный минус (-).
   Используется для получения, например, противоположного числа.
3. Умножение, деление (* / % //).
4. Сложение и вычитание (+ -).
5. Операции сравнения (<= < > >=)
6. Операции равенства (== !=)
7. Логические операции (not and or)
8. Операции присваивания (=)

Если используются операции с разными приоритетами, попробуйте добавить пробелы вокруг операций с самым низким приоритетом. Руководствуйтесь своими собственными суждениями, но никогда не используйте более одного пробела и всегда используйте одинаковое количество пробелов по обе стороны бинарной операции.

Простейшие функции

Простейшие функции

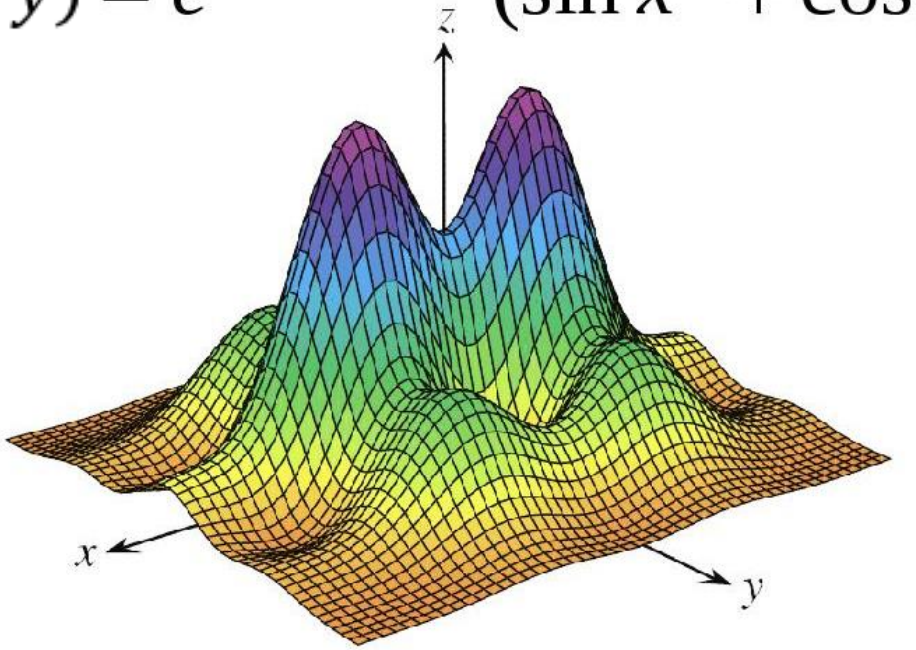
В математике функция из одного числа (или даже нескольких) делает другое.

Функция – это зависимость одного числа от другого или других.

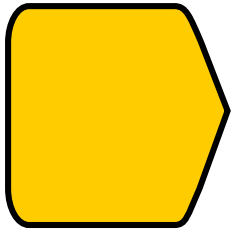
$$f(x) = x^2 - 5x + 8$$

$$f(x, y) = x^2 + y^2$$

$$f(x, y) = e^{-(x^2+y^2)/8}(\sin x^2 + \cos y^2)$$



Функции в программировании



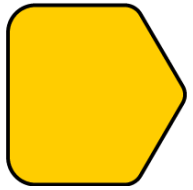
В программировании (и в Python в частности): функция – это сущность, которая из одного (или даже нескольких) значений делает другое. При этом она может ещё и выполнять какие-то действия.

Например, есть функция модуля y = |x|, аналогично в Python есть функция y = abs(x).

Но функции в Python необязательно принимают только числа.

Функции преобразования типов

Для того, чтобы вводить числа с клавиатуры и далее работать с ними, нам необходимо найти функцию, которая из строки делает число. И такие функции есть!



Тип данных целых чисел в Python называется int, дробных чисел — float.
Одноимённые функции принимают в качестве аргумента строку и возвращают число, если в этой строке было записано число (иначе выдают ошибку):

```
a = input()
b = int(a)
print(b + 1)
```

или

```
a = int(input())
print(a + 1)
```

или

```
print(int(input()) + 1)
```

Функции преобразования типов

С помощью функции float можно из вводимой строки получить вещественное число, а с помощью функции str преобразовать число в строку:

```
a = float(input()) -> 25.8
a = a + 1.05
print(a)            -> 26.85
print(str(a))       -> '26.85'
print(int(a))       -> 26
```

Функция **int** может быть применена и для получения целого числа из вещественного — в таком случае, дробная часть будет отброшена (без округления).

Округление

Для математического округления используется функция round(), её можно использовать с одним аргументом — округляемым числом, — тогда оно будет округлено до целого по правилам математики:

```
a = float(input()) -> 2.8
print(round(a))      -> 3
print(round(2.16))   -> 2
```

А можно записать второй аргумент — количество знаков после запятой, до которых надо округлить первый аргумент:

```
print(round(3.1415926, 2)) -> 3.14
print(round(2.71828, 4))   -> 2.7183
```

Функция len

Функция len

Длина строки — это количество символов в строке. Для определения длины строки используется стандартная функция Python len().

```
word = input()
length = len(word)
print('Вы ввели слово длиной', length, 'букв.')
```

В нашем примере данные в скобках должны быть строкой. Мы выбрали в качестве данных значение переменной word, которое пользователь до этого ввёл с клавиатуры. То есть значение переменной word выступает здесь в роли аргумента. А функция len выдаёт длину этой строки. Если пользователь ввёл, например, «привет», то в word оказывается равно «привет», и на место len(word) подставляется длина строки «привет», то есть 6.

Когда мы в программе используем функцию, это называется «вызов функции». Вызов функции устроен так: пишем имя функции — len, а затем в скобках те данные, которые мы передаём этой функции, чтобы она что-то с ними сделала. Такие данные называются аргументами.

Функция len

Обратите внимание: каждый раз, когда мы пишем имя переменной (кроме самого первого раза — в операции присваивания слева от знака =), вместо этого имени интерпретатор подставляет значение переменной.

```
word = input()
length = len(word)
print('Вы ввели слово длиной', length, 'букв.')
```

Точно так же на место вызова функции (то есть имени функции и её аргументов в скобках) подставляется результат её работы — это называется возвращаемое значение функции.

Таким образом, функция len возвращает длину своего аргумента. input — тоже функция (отсюда скобки), она не принимает никаких аргументов, зато считывает строку с клавиатуры и возвращает её.

print — тоже функция, она не возвращает никакого осмысленного значения, зато выводит свои аргументы на экран. Эта функция может принимать не один аргумент, а сколько угодно. Несколько аргументов одной функции следует разделять запятыми.

Функции min и max

Функции min и max

Для определения соответственно минимального или максимального значения в последовательности однотипных данных используются функции min() и max(). Аргументов у этих функций может быть любое количество, главное, чтобы они все были одного типа.

```
a = max(3, 8, -3, 12, 9)
b = min(3, 8, -3, 12, 9)
print(a, b) -> 12 -3
```

```
a = max('a', 'd', 'ee', 'A', 'aa')
b = min('a', 'd', 'ee', 'A', 'aa')
print(a, b)   -> ee A
```

А вот сравнить данные разных типов не получится:

```
a = max('a', 'd', 1)
print(a)
```

```
Traceback (most recent call last):
  File "1.py", line 1, in <module>
    a = max('a', 'd', 1)
TypeError: '>' not supported between instances of 'int' and 'str'
```

Обмен значениями переменных

Обмен значениями переменных

Давайте попробуем написать программу, которая поменяет местами содержимое переменных a и b. Пусть есть такой код:

```
a = 3
b = 5
...
...
print(a)
print(b)
```

Что надо вписать в пропущенные места, чтобы в a лежало 5, а в b лежало 3? При этом, числами 3 и 5 пользоваться нельзя.

Обмен значениями переменных

Как один из вариантов – можно использовать дополнительную переменную:

```
a = 3
b = 5
c = a
a = b
b = c
print(a)
print(b)
```

А теперь попробуйте написать вариант без дополнительной переменной, через сумму или разность двух чисел.

Обмен значениями переменных

Сложение:

```
a = 3
b = 5
a = a + b   # 8
b = a - b   # 8-5=3
a = a - b   # 8-3=5
print(a)
print(b)
```

Вычитание:

```
a = 3
b = 5
a = a - b   # -2
b = a + b   # -2+5=3
a = -a + b   # 2+3=5
print(a)
print(b)
```

Но нам с вами очень повезло, что мы изучаем язык Python, потому что он и поддерживает более простой вариант записи:

```
a = 3
b = 5
a, b = b, a
print(a)
print(b)
```

Форматирование строк

Форматирование строк

Чтобы вывести предварительно отформатированную строку, перед кавычками (любыми — одинарными, двойными или тройными) нужно поставить спецификатор — букву f или F. Тогда интерпретатор поймёт, что строку нужно выводить именно в таком виде, в котором она написана, за исключением значений, заключённых в фигурные скобки.

```
kind = 'черепаха'
name = 'Тони'
size = 20
print(f'В живом уголке живёт {kind} по имени {name}, а размер в см: {size}!')
# В живом уголке живёт черепаха по имени Тони, а размер в см: 20!
```

Допустим, у нас есть переменные, значения которых мы хотим подставить в некоторые места строки. Тогда в фигурных скобках в этих местах мы напишем имя переменной. При выводе вместо переменной подставится значение, а вся остальная строка сохранит свой вид неизменным.

Форматирование строк

Если нужно заключить часть строки в кавычки, нужно обозначить тип данных — строка — кавычками другого типа.

```
book = 'Приключения Буратино'
print(F'Самая известная черепаха описана в книге "{book}".')
# Самая известная черепаха описана в книге "Приключения Буратино".
```

Внутри фигурных скобок можно производить вычисления, например, так:

```
width = 15
height = 35
print(f'Площадь прямоугольника {width * height}.')
# Площадь прямоугольника 525.
```

Форматирование строк

Можно дополнить выводимую строку символами до заданной длины. Дополнять можно так, чтобы текст выравнивался по центру, по правому или по левому краю:

```
name = 'Тони'
print(f'Все имена одинаковой длины: {name:*^10}.')
# Все имена одинаковой длины: ***Тони***.

name = 'Тортилла'
print(f'Все имена одинаковой длины: {name:!<10}.')
# Все имена одинаковой длины: Тортилла!!.

name = 'Дженни'
print(f'Все имена одинаковой длины: {name:$>10}.')
# Все имена одинаковой длины: $$$$Дженни.
```

Если не указывать символ для выравнивания, то оно производится пробелами.

Форматирование строк

Целые числа можно выводить с ведущими нулями (после двоеточия указывается общее количество цифр, которое должно получиться в числе, включая нули):

```
cod = 985
number = 1234
print(f'Номер +7({cod}){number:08}')
# Номер +7(985)00001234
```

А вещественные числа с округлением до заданного количества знаков после запятой. Надо понимать, что само число при этом не меняется.

```
numerator = 11
denominator = 7
print(f'Значение дроби: {numerator / denominator:.3f}')
# Значение дроби: 1.571

part = 17
whole = 44
print(f'Процентное соотношение: {part / whole:.2%}')
# Процентное соотношение: 38.64%
```

Форматирование строк

Начиная с версии Python 3.12, в шаблонах f-строк можно использовать те же кавычки, что и снаружи. Например, вот такой код будет работать:

```
cat = 'Barsik'
print(f"{f"{cat}-cat" + f" {len(cat)} symbols"} in name")
# Barsik-cat 6 symbols in name
```


Яндекс