**Функции**

То, что пишется в скобках у функции называется **аргументами или параметрами**.

Функция может получать параметры и возвращать значение

c = min(a, b)

Функция получила два параметра и вернула значение – минимальное из них.

Полученное значение можно не только присваивать переменным, но и делать с ним все, что можно делать со значением данного типа, например, использовать в арифметических выражениях или печатать на экране.

**print**(min(a, b) + min(c, d))

До этого мы использовали стандартные функции. Стандартные функции объявляются в библиотеках **модулях**. Есть модули

math – содержит математические функции

sys – содержит функции, которые позволяют работать с файлами

reg – позволяет анализировать и разбирать содержимое текстовых строк при помощи регулярных выражений

есть модули, которые позволяют рисовать что-либо на экране.

Богатство языка Python является его стандартные библиотеки, т.е. набор тех функций, которые доступны программисту в его программе. В каком-то смысле, чем больше в вашем языке каких-то функций, тем он более универсален и позволяет выполнять разные задачи.

Создание собственных функций облегчает программисту жизнь. Программист должен уметь свой код разбивать на функции. Это нужно, чтобы:

1. Если в программе есть действия, которые выполняются несколько раз с разными исходными данными, то вам нужно один раз реализовать алгоритм, а дальше многократно в любом месте программы можно использовать этот алгоритм.

2. При помощи функций код структурируется. Если у вас очень большая программа, то очень тяжело в ней разобраться, когда вся программа записана в одном файле. В этом случае лучше разбить программу на маленькие кусочки. Каждый кусочек будет являться отдельным алгоритмом, отдельной функцией. Функции будут вызывать друг друга. При написании программы вы можете написать маленький кусочек (1 функцию) отладить его, а потом перейти к следующему и т.д. В каждый момент времени вы работаете с небольшим кусочком.

3. Позволяет организовать коллективную работу над программой. Каждый человек пишет свою функцию и из них складывается программа. Из этих функций складывается программа.

Функция возвращает значение return. Возможны множественные точки возврата из функции

**Определение функции**

**def** min(a,b):

 **if** a<b:

 **return** a

 **else**:

 **return** b

Тело функции должно вернуть значение с помощью return. Как только функция попадает на этот оператор, то сразу происходит выход из функции. Поэтому функцию выше можно написать так

**def** min(a,b):

 **if** a<b:

 **return** a

 **return** b

Простейший вариант: функция, которая просто повторяет одни и те же действия, независимо ни от чего. У каждой функции есть заголовок (его обычно называют сигнатурой) и тело функции. Сигнатура описывает, как функцию вызывать, а тело описывает, что функция делает. Сигнатура содержит имя функции, а также аргументы (то есть параметры) которые передаются в функцию. Аргументов, в принципе, может и не быть. Записывается это так:

**def** <имя функции>([аргументы]...):

 <тело функции>

**Определение должно предшествовать вызову.**

**Вызов функции**

x = **int**(**input**())

y = **int**(**input**())

z = min(x, y)

**print**(min(3, 8))

Имена переменных при вызове функции никакого отношения к именам параметров не имеет. Вызывая функцию, вы можете не знать, как названы аргументы в определении функции. Вы должны знать их количество.

Преобразовать все маленькие латинские буквы в строке в верхний регистр.

**def** is\_low\_letter(c):

 **if** "a" <= c <= "z":

 **return** **True**

 **else**:

 **return** **False**

**def** to\_upper(c):

 **return** **chr**(**ord**(c) - 32)

s = **input**()

res=""

**for** c **in** s:

 **if** is\_low\_letter(c):

 res += to\_upper(c)

 **else**:

 res += c

**print**(res)

Бывают функции, которые значение не возвращают. Такие функции не содержат в теле return.

Распечатка списка

**def** print\_arr(a):

 **for** el **in** a:

 **print**(el, end=" ")

 **print**()

Из функции, которая не возвращает значение, можно выйти в любой момент, вызвав return без параметров. Если функция ничего не возвращает, то говорят, что она возвращает специальное значение **None**.