Лекция 9

Функции

Введение. Понятие функции

В этой части мы изучим функции — составные инструкции, которые могут принимать данные ввода, выполнять указания и возвращать данные вывода.

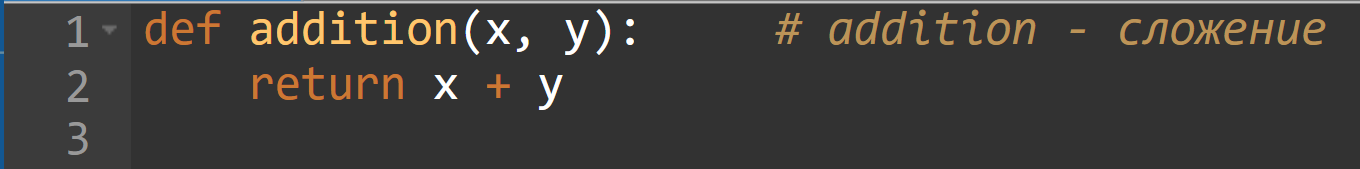
Тело функции — набор инструкций, которые поочередно исполняют в процессе вызова.

Функции помогают разбить нашу программу на более мелкие части. По мере того, как наша программа становится все больше и больше, функции делают ее более организованной и управляемой.

Также создание функций следует принципу DRY (Don’t Repeat Yourself) – “не повторяйся”. Этот принцип требует, чтобы часто используемые части информации встречались в вашем коде один, и только один раз.

Синтаксис

Любые функции начинаются с инструкции def (от английского define — «определять»). За def в Python идет название функции, после — набор аргументов. Двоеточие в конце открывает блок тела функции, которое располагают с новой строки.

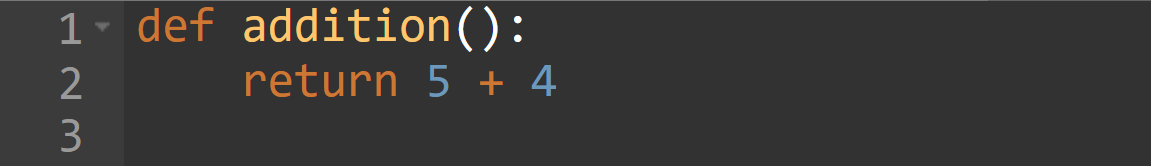
Давайте напишем простую функцию.

Данная функция принимает на вход два параметра x и y.

Инструкция return говорит, что нужно вернуть значение.

В Python каждая последующая инструкция отделяется новой строкой. Часть кода, входящая в определённую инструкцию, отделяется табуляцией.

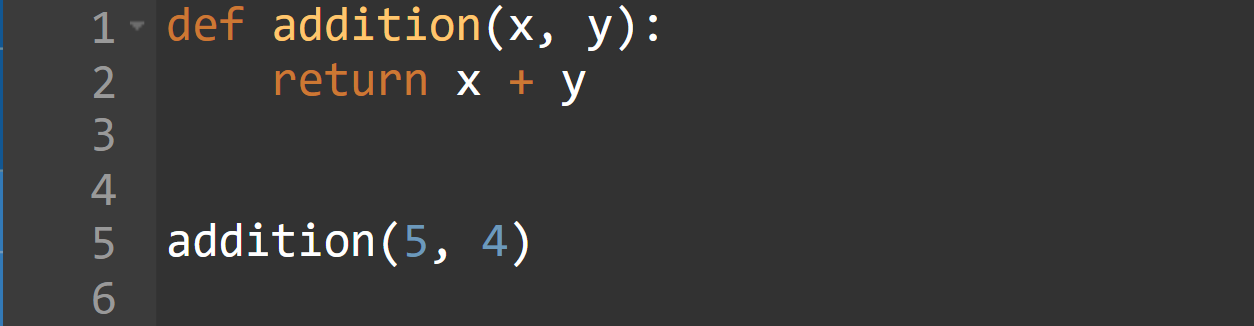
Также мы можем не передавать никаких значений функции или. Для этого нужно оставить круглые скобки пустыми.



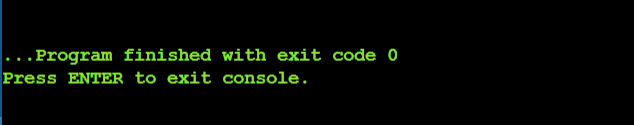
Итак, мы можем передавать в функцию бесконечное число аргументов или же не передавать ничего.

Вызвать функцию — значит передать ей входные данные, необходимые для выполнения и возвращения результата. Когда вы передаете функции входные данные, это называется передача параметра функции.

Давайте вызовем нашу функцию.

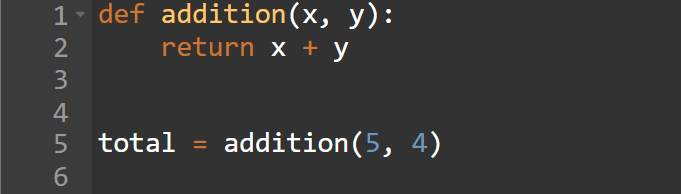


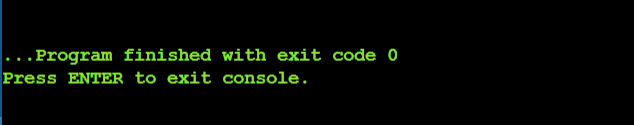
Вот что вывела наша программа:



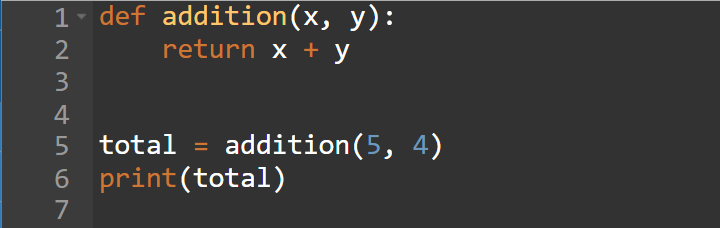
Почему же консоль ничего не вывела? Дело в том, что мы никуда не сохранили наше возвращаемое значение. Функция выполнилась, но ничего не запомнила.

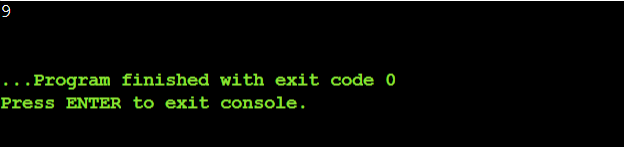
Давайте сохраним значение в переменную.



Консоль вывела:  


Казалось бы, значение сохранилось в переменную, но консоль по-прежнему ничего не выводит. Всё потому что значение нужно вывести, чтобы его увидеть.

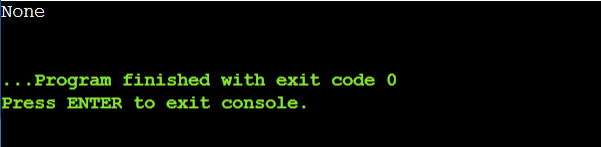




Вот теперь наша программа выводит число 9. Функция успешно сработала!

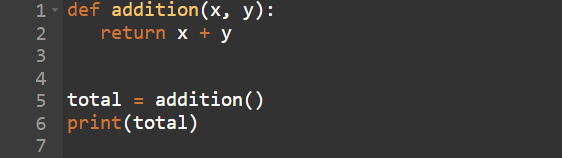
Функция может и не заканчиваться инструкцией return, при этом она вернет значение None.



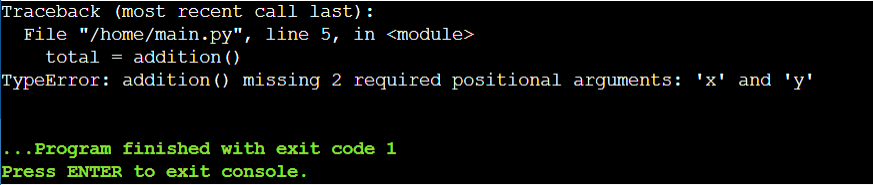


Что и требовалось доказать! Консоль вывела None.

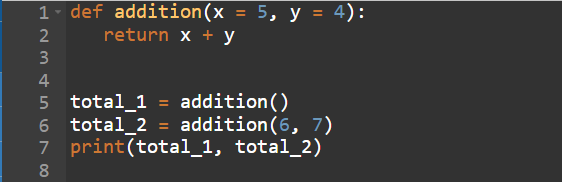
Как вы заметили, при вызове функции с параметрами мы чётко передавали ей нужные значения. Что же будет, если мы этого не сделаем?



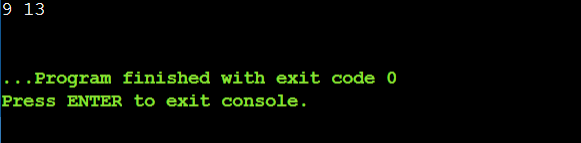
Как видите, консоль выдала ошибку.



Такие параметры называются обязательными. Если не ввести никаких значений, программа будет выдавать ошибку. Её можно исправить, используя опциональные параметры.



На данном этапе программа вывела 9 и 13.

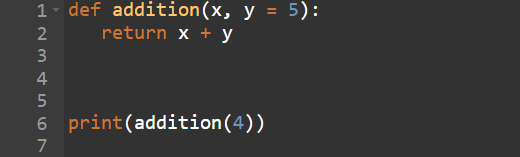


При первом вызове мы не передали в функцию никаких значений, соответственно, x приняло значение 5, y же стал равен 4.

Но во втором случае мы передали конкретные значения, и функция использовала именно их.

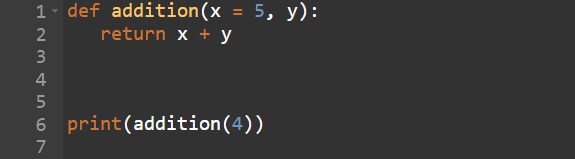
Если функция имеет оба типа параметров, то опциональные параметры должны стоять в конце, так как они считываются справа налево.

Рассмотрим примеры.

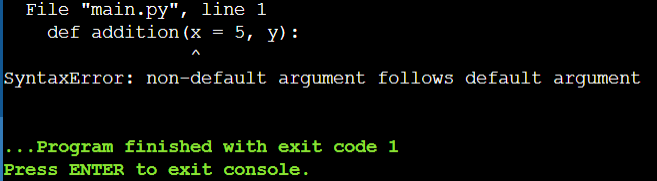


В данном случае всё просто – программа выведет число 9, так как параметр y по умолчанию будет равен 5.

Но что же будет, если мы укажем параметры наоборот?



Казалось бы, программа выведет тоже самое значение, но нет! Поскольку компилятор считывает параметры справа налево, он не сможет задать все значения аргументам и выдаст ошибку.



Правила записи идентификаторов

В заключение научимся правильно задавать имена функциям.

1. Нельзя использовать ключевые слова для определения функции (ключевые слова — это зарезервированные слова в Python).
2. В записи идентификаторов можно использовать только латинские буквы (a-z) в нижнем и верхнем регистре, цифры (0–9) и символ подчеркивания (\_). Но помните, что Python чувствителен к регистру, то есть названия addition и Addition - будут разными идентификаторами.
3. Идентификатор не может начинаться с цифры (1addition - не подходит).
4. Лучше всего определять названия функций с маленькой буквы. Если же оно состоит из нескольких слов, то есть два принятых варианта, так называемые верблюжья нотация (camel case) и змеиная нотация (snake case).

Camel case. Первое слово пишется со строчной буквы, следующие — с заглавной, разделителей между составными частями нет. Торчащие посреди итогового названия заглавные буквы напомнили кому-то горбы верблюда — так возникло название нотации.



Snake case. Слова разделяются символами подчёркивания — они как бы ползут по строке, в результате получается длинное, как змея, название.

