Самое важное

Оператор def
Оператор def — это оператор создания функций. Именно он указывает Python на то, что код, идущий далее (с отступом) является частью функции.

Синтаксис
def <название функции>(<параметр1>, <параметр2> и тд):
    <тело функции>

Внимание: тело функции (код, который будет выполняться при вызове функции) расположено с отступом, именно отступ даст понять Python, где заканчивается функция:

def foo(x):
  print(x) — этот код будет выполнен внутри функции.
print(123) — этот код будет выполнен вне зависимости от функции.

Параметров у функции может быть сколько угодно, все их нужно перечислить через запятую.

При вызове функции все эти параметры будут созданы как обычные переменные.

То есть когда пишут:
def foo(x):
  print(x)

foo(5) — Python видит это как <запуск функции foo с параметром x=5> и первым делом, ещё до запуска кода из тела функции, он вызовет операцию x = 5, а затем уже пойдёт выполнение кода print(x).

Вывод: переменные снаружи и параметры — это разные объекты, которые никак не связаны.

Если пишут:
def foo(x):
  print(x)

x = 10
foo(5) — функция всё равно создаст внутри себя свою собственную переменную ‘x’ и присвоит ей значение 5, которое мы указали при вызове. В итоге функция выдаст в консоль сообщение ‘5’, а не ‘10’.

Но если мы напишем print(x), то в консоли увидим ‘10’, то есть создание ‘x’ внутри функции никак не повлияло на ‘x’, который мы создали снаружи.

Об этом важно помнить и желательно не создавать переменные и параметры с одинаковыми названиями, иначе это может привести к путанице.

Вложенный вызов функции

На самом деле этот приём уже использовался в примерах, которые рассматривали выше.

def foo(x):
  print(x) — это, по сути, вложенный вызов функции, просто это стандартная функция, которая уже была создана до нас.

Однако это значит, что внутри функции можно вызывать и любую другую функцию, в том числе созданную программистом.

```
def foo(x):
  bar()


def bar():
  print('Hello')
```

При желании можно даже передавать параметр текущей функции во вложенную:

```
def foo(x):
  bar(x)


def bar(y): — старайтесь использовать разные названия для параметров.
  print(y + 2)
```

Не допускай следующих ошибок!

Не забывайте передавать все необходимые данные внутрь функции.
Представьте, что функция — это отдельный «модуль», который не знает о существовании переменных «снаружи».

```
x = input()

def math_test():
   if x > 5:
      x = 0
   else:
      x = 1
```

```
print(x)
```

Такая операция вызовет ошибку!