Рассмотрим пример. В отдельный файл с именем `myprog.py` поместим следующий код:

```
a = 3 # глобальная переменная
print('глобальная переменная a = ', a)
y = 8 # глобальная переменная
print('глобальная переменная y = ', y)

def func():
    print('func: глобальная переменная a = ', a)
    y = 5 # локальная переменная
    print('func: локальная переменная y = ', y)

func() # вызываем функцию func
print('??? y = ', y) # отобразится глобальная переменная
```

Обращаю внимание, что у функции `print` могут быть несколько аргументов, заданных через запятую. В одинарные кавычки помещается строка[14].

После выполнения программы получим следующий результат:

```
>>>
======= RESTART: C:/Python35-32/myprog.py =========
глобальная переменная a = 3
глобальная переменная y = 8
func: глобальная переменная a = 3
func: локальная переменная y = 5
??? y = 8
>>>
```

Внутри функции мы смогли обратиться к глобальной переменной `a` и вывести ее значение на экран. Далее внутри функции создается локальная переменная `y`, причем ее имя совпадает с именем глобальной переменной – в этом случае при обращении к `y` выводится содержимое локальной переменной, а глобальная остается неизменной.
Как быть, если мы хотим изменить содержимое глобальной переменной внутри функции? Ниже показан пример такого изменения с использованием ключевого слова `global`[15]:

```
x = 50 # глобальная переменная
def func():
    global x # указываем, что x - глобальная переменная
    print('x равно', x)
    x = 2    # изменяем глобальную переменную
    print('Заменяем глобальное значение x на', x)

func()
print('Значение x составляет', x)
```

[14] О строках подробно поговорим в следующей главе
[15] В Python «явное лучше неявного».