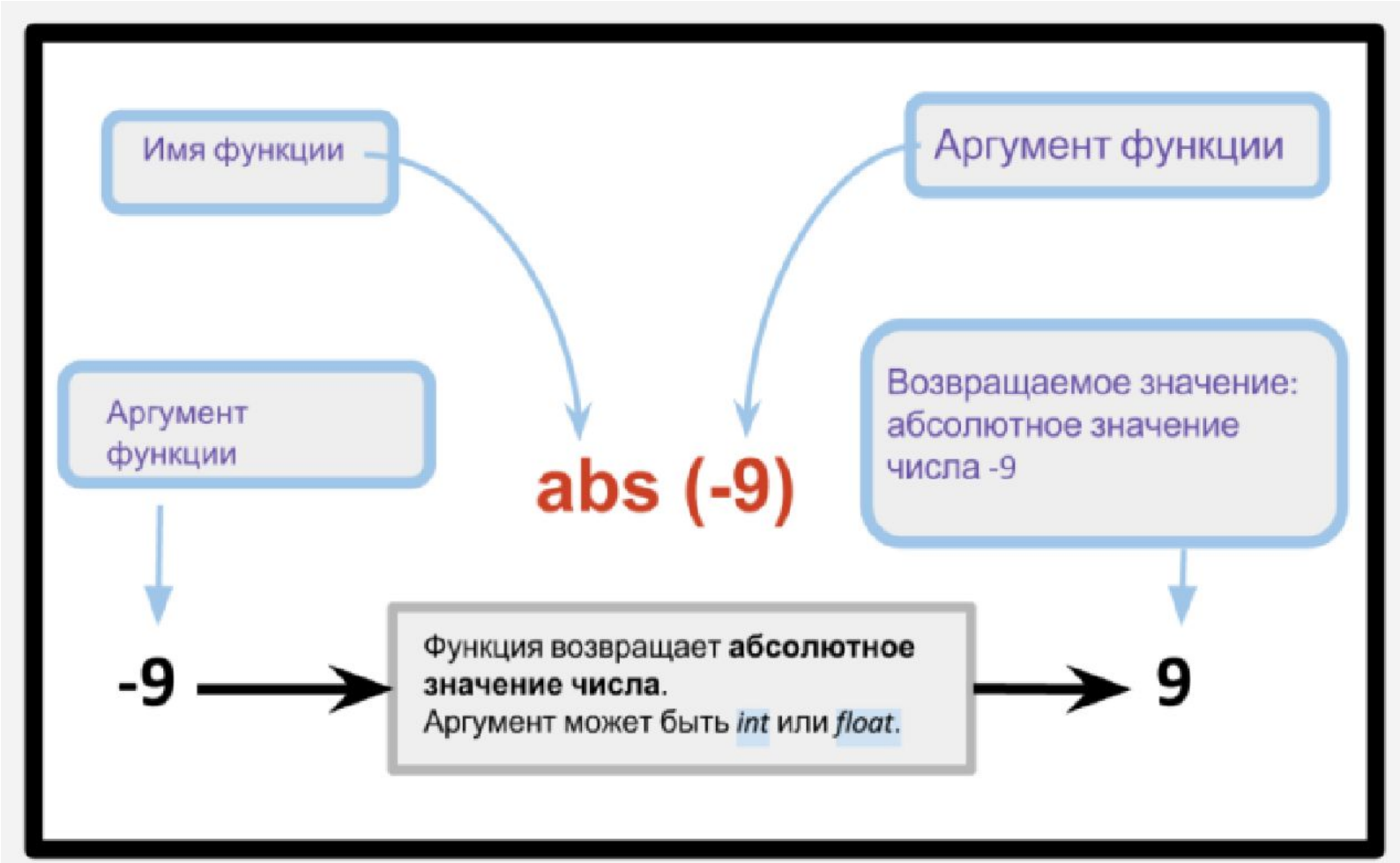
Введение в Python

Лекция 2: Функции

Тема: Функции



Пример вызова функции `abs()` с аргументом -9 имеет вид:

```
>>> abs(-9)
9
>>> d = 1
>>> n = 3
>>> abs(d - n)
2
>>> abs(-9) + abs(5.6)
14.6
>>>
```

Некоторые математические функции в Python

`pow(x, y)` возвращает значение `x` в степени `y`. Эквивалентно записи `x**y`.

```
>>> pow(4,5)
1024
>>>
```

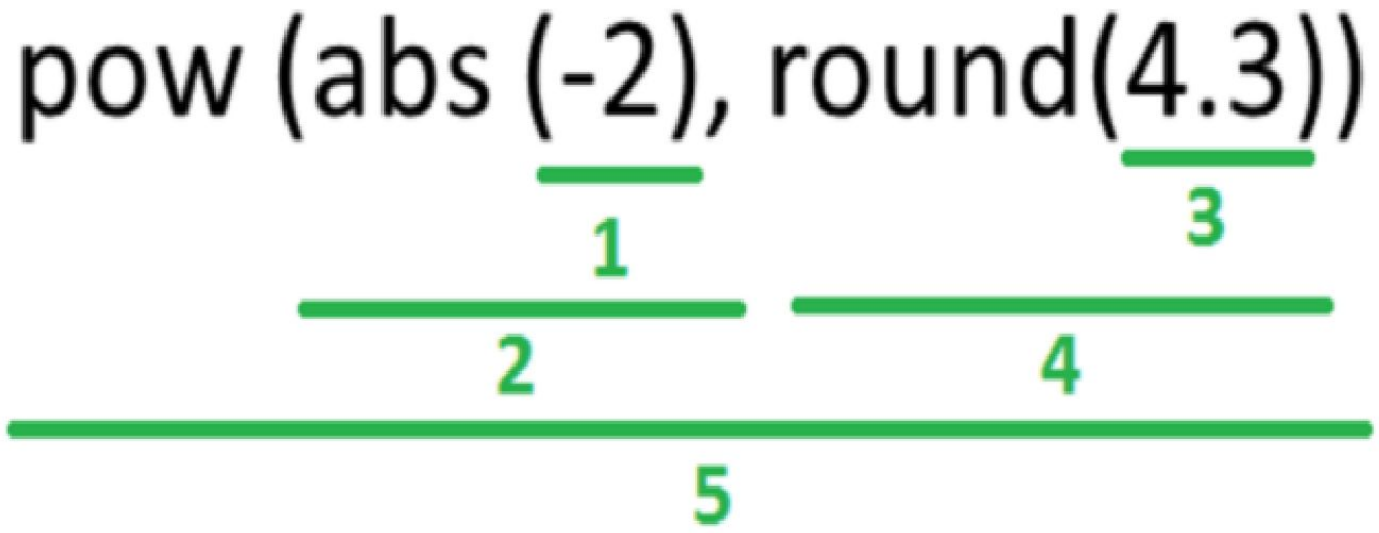
Функция round (number)

`round(number)` возвращает число с плавающей точкой, округленное до 0 цифр после запятой (по умолчанию). Может быть вызвана с двумя аргументами:

`round(number[, ndigits])`, где `ndigits` – число знаков после запятой.

```
>>> round(4.56666)
5
>>> round(4.56666, 3)
4.567
>>>
```

Помимо составления сложных математических выражений Python позволяет передавать результаты вызова функций в качестве аргументов других функций без использования дополнительных переменных:



Функции преобразования числовых объектов:

`int()` возвращает целочисленный объект, построенный из числа или строки, или 0, если аргументы не переданы.

`float()` возвращает число с плавающей точкой, построенное из числа или строки.

Рассмотрим примеры:

```
>>> int(5.6)
5
>>> int()
0
>>> float(5)
5.0
>>> float()
0.0
>>>
```

В Python документация для функции может быть вызвана с помощью функции `help()`, на вход которой передается имя функции:

```
>>> help(abs)
Help on built-in function abs in module
builtins:

abs(x, /)

Return the absolute value of the argument.

>>>
```

Создание собственной функции (процедуры)

формула перевода градусов по шкале Фаренгейта (T_F) в градусы по шкале Цельсия (T_C):

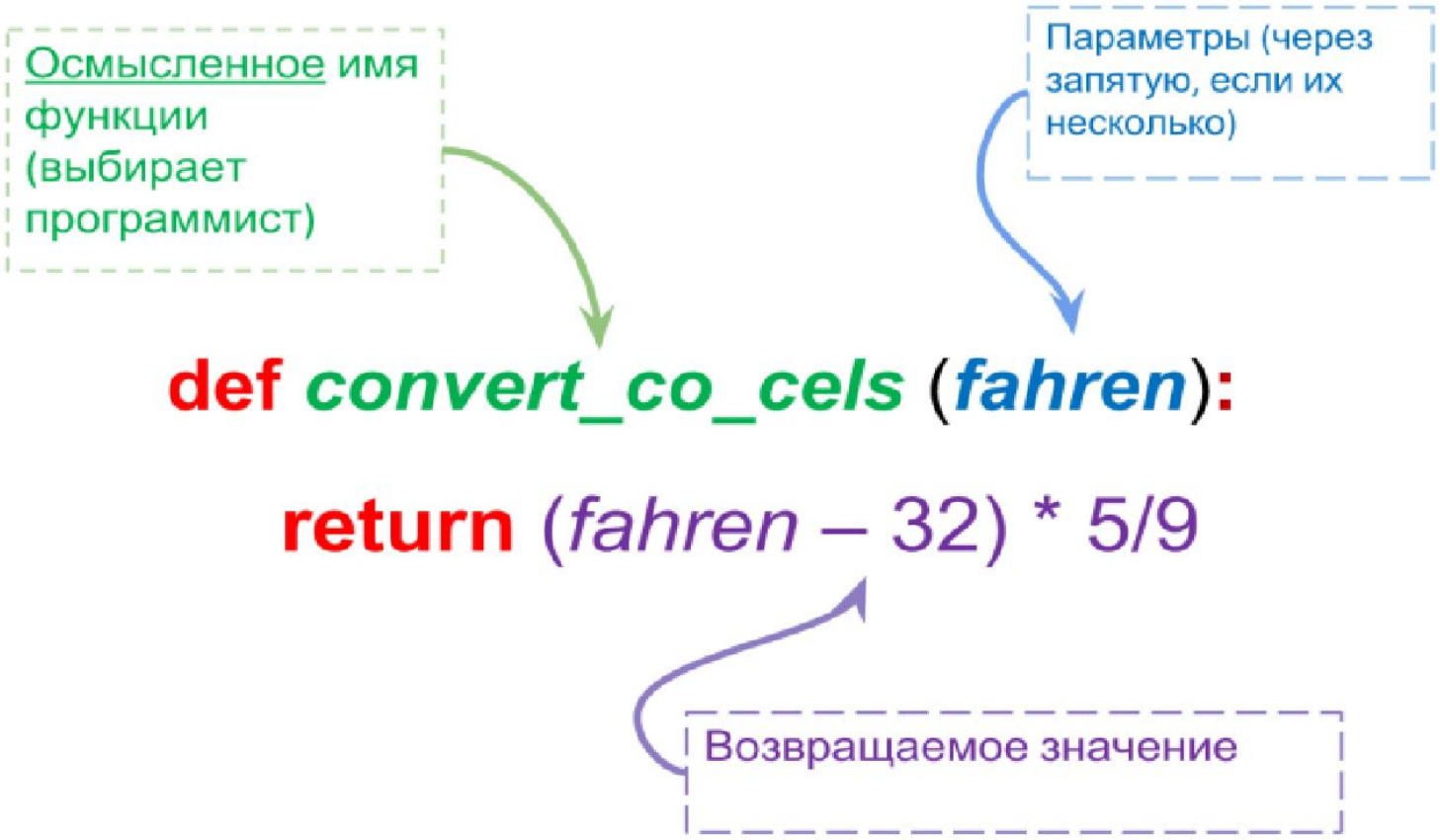
```
TC = 5/9 * (TF - 32)
```

Произведем несколько вычислений, где переменная `deg_f` будет содержать значение в градусах по Фаренгейту:

```
>>> deg_f = 80
>>> deg_f
80
>>> 5/9 * (deg_f - 32)
26.666
>>> deg_f = 70
>>> 5/9 * (deg_f - 32)
```

Создадим собственную функцию, переводящую градусы по Фаренгейту в градусы по Цельсию:

Пусть функция с именем `convert_co_cels` создана, тогда ее вызов для значения 80 будет иметь вид: `convert_co_cels(80).`



- `Ключевое слово def` для Python означает, что дальше идет описание функции. После `def` указывается имя функции `convert_co_cels`, затем в скобках указывается параметр, которому будет присваиваться значение при вызове функции.
- Параметры функции – обычные переменные, которыми функция пользуется для внутренних вычислений. Переменные, объявленные внутри функции, называются *локальными* и не видны вне функции.
- После символа «:» начинается тело функции. В интерактивном режиме Python самостоятельно поставит отступ от края экрана, тем самым обозначив, где начинается тело функции.
- Выражение, стоящее после ключевого слова `return` будет возвращаться в качестве результата вызова функции.

В интерактивном режиме создание функции имеет следующий вид

```
>>> def convert_co_cels(fahren):
        return (fahren-32)*5/9
>>> convert_co_cels(451)
232.77777777777777
>>> convert_co_cels(300)
148.88888888888889
>>>
```

Программы в отдельном файле

```
a=5
print(a)
print(a+5)
```

В меню выберем Save As и сохраним файл в произвольную директорию, указав имя `myprog1.py`.

Чтобы выполнить программу в меню редактора выберем

Run -> Run Module (или <F5>). Результат работы программы отобразится в интерактивном режиме:

```
>>>
========== RESTART: C:/Python35-32/myprog1.py ==========
5
10
>>>
```

Как создавать функции в отдельном файле и вызывать их:

```
def f(x):
    x = 2 * x
    return x
    print(f(4)) # комментарии игнорируются Python
    print(f(56))
```

Запустим программу с помощью F5 и увидим, что в интерактивном режиме отобразился результат

```
======== RESTART: C:/Python35-32/myprog.py ==========
8
112
>>>
```

Область видимости переменных. Глобальная переменная

Переменная является **локальной** (видна только внутри функции), если значение ей присваивается внутри функций, в ином случае – **переменная глобальная**, т.е. видна (к ней можно обратиться) во всей программе, в том числе и внутри функции.

- **Глобальная переменная** — если ей присвоено значение в основной программе (вне процедуры).
- **Локальная переменная** (внутренняя) известна только на уровне процедуры, обратиться к ней из основной программы и из других процедур нельзя.

Пример. В отдельный файл поместим 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
```

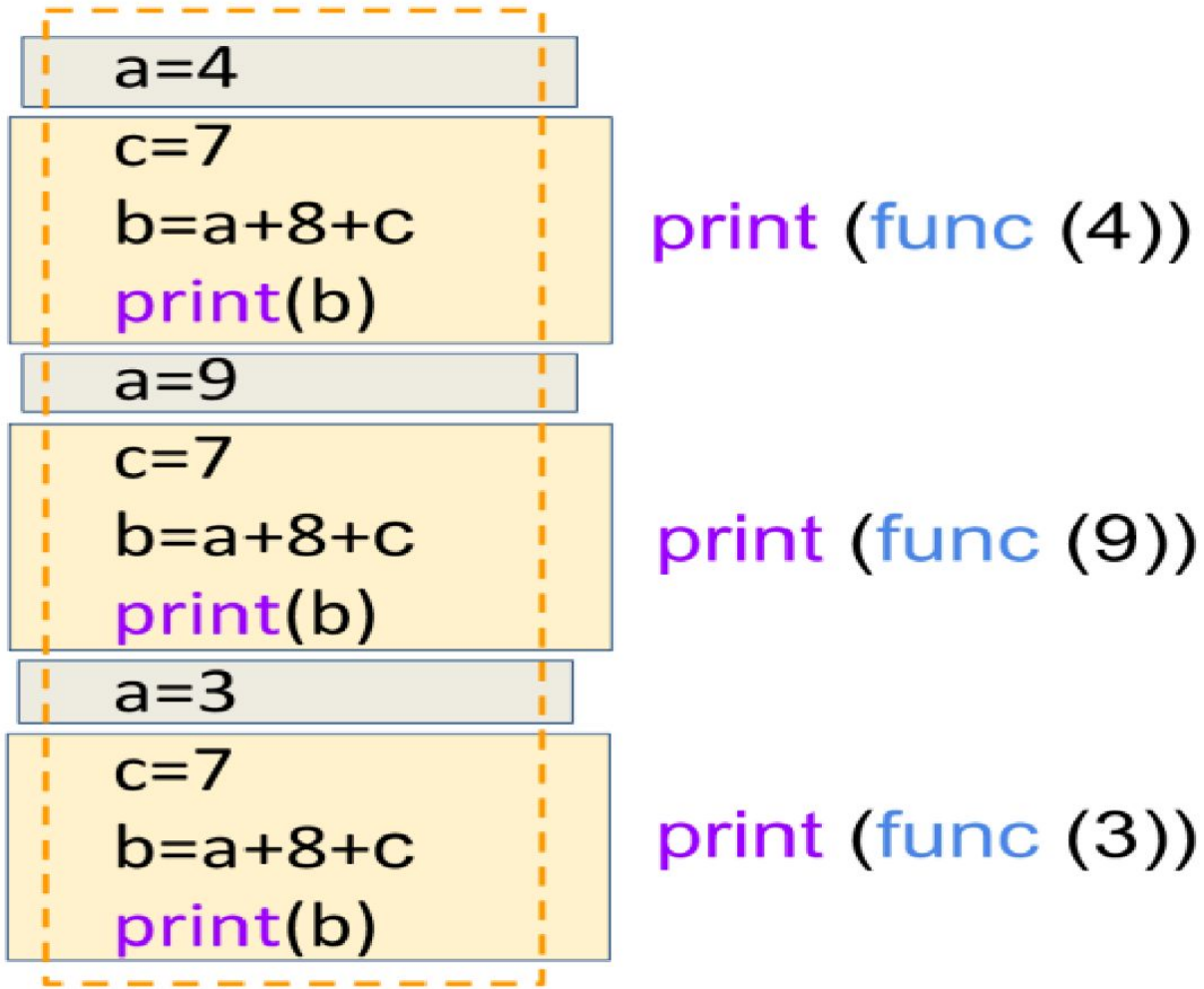
Как быть, если мы хотим изменить содержимое глобальной переменной внутри функции?

```
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)
```

Часто функции используются для сокращения кода программы, например, объявив функцию вида:

```
def func(x):
    c = 7
    return x + 8 + c
```

Следующий код может быть заменен на три вызова функции с различными аргументами:



Случай, когда функция ничего не принимает на вход и ничего не возвращает (не используется ключевое слово `return`):

```
def print_hello():
    print('Привет')
    print('Hello')
    print('Hi')
```