



<https://dfedorov.spb.ru>

>> far = 80

>> far

80

>> 5/9 * (far - 32)

26.666

>> far = 70

>> 5/9 * (far - 32)

far_to_cels (80)



$$T_C = 5/9 * (T_F - 32)$$



Собственные функции

Имя функции
(придумывает программист)

Аргумент функции (градусы по Фаренгейту)

Возвращаемое значение в градусах по Цельсию

Аргумент функции
(градусы по Фаренгейту)

far_to_cels(80)

80

Функция возвращает значение в градусах по Цельсию.
Аргумент – значение градусов по Фаренгейту (может быть *int* или *float*)

26.666

Собственные функции

Осмысленное имя
функции
(выбирает
программист)

Параметры (через
запятую, если их
несколько)

```
def far_to_cels(far):
```

```
    return (far - 32) * 5/9
```

Tab ← →

Возвращаемое значение

В момент передачи аргумента **80** функции **far_to_cels** происходит связывание **far** со значением **80**.

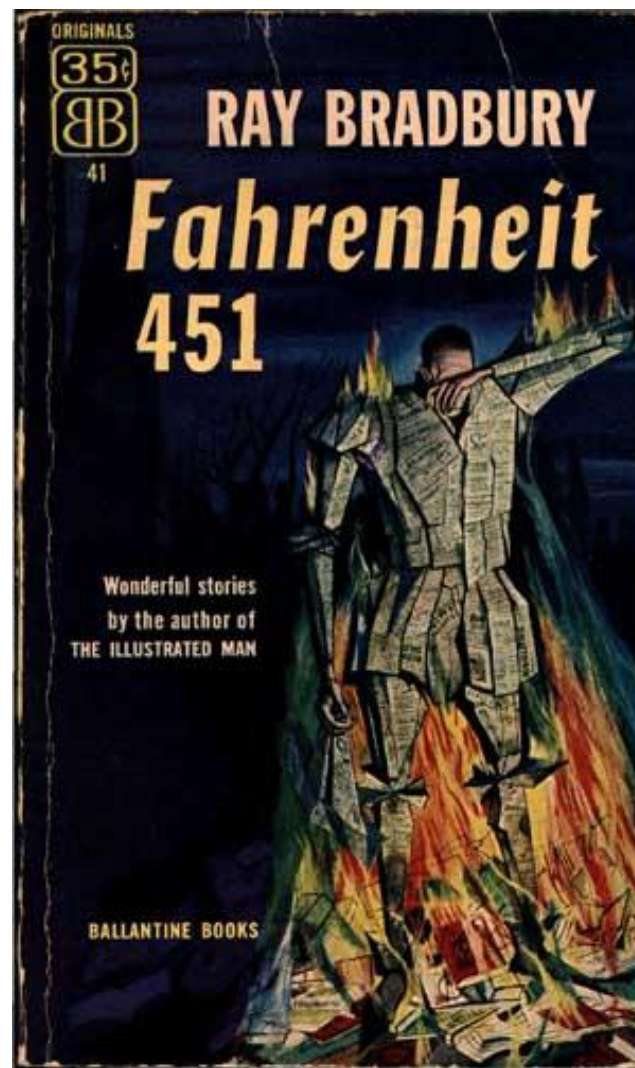
```
>>> def far_to_cels(far):  
    return (far-32) * 5/9
```

```
>>> far_to_cels(80)  
26.666666666666666668
```

```
>>>
```

Упражнение

У американского писателя-фантаста Рэя Бредбери есть роман "451 градус по Фаренгейту".
Какой температуре по шкале Цельсия соответствует указанное в названии значение?





**Функции должны
делать одну вещь и
делать ее хорошо и
делать только ее.**

Роберт С. Мартин,
консультант и автор в области
разработки ПО



Область видимости переменных

Почему полезно
создавать функции?


```
>>> a = 4
```

```
>>> b = 7
```

```
>>> fun()
```

```
>>> a
```

```
4
```

```
>>> b
```

```
7
```

```
def fun():
```

```
    # внутри функции
```

```
    # создаются
```

```
    # локальные переменные
```

```
    # с именами a и b
```

```
    a = 6
```

```
    b = 9
```

```
    return None
```

Область памяти внутри функции называется фреймом

```
>>> a = 4
```

```
>>> b = 7
```

```
>>> fun()
```

```
>>> a
```

```
6
```

```
>>> b
```

```
7
```

```
def fun():
```

```
    # внутри функции
```

```
    # создаются переменные и
```

```
    # a объявляется глобальной
```

```
    global a
```

```
    a = 6
```

```
    b = 9
```

```
    return None
```

Связь между именами функций и переменными

```
>>> def g():  
    return 1
```

Теперь **g** связано с функцией, возвращающей значение 1

```
>>> g()
```

```
1
```

```
>>> g = 2
```

```
>>> g
```

Теперь **g** связано с целочисленным значением 2

```
>>> def g(h, i):  
    return h + i
```

```
>>> g(1, 2)
```

```
3
```

```
>>> def g():  
    return 1
```

```
>>> g()
```

1

```
>>> g = 2
```

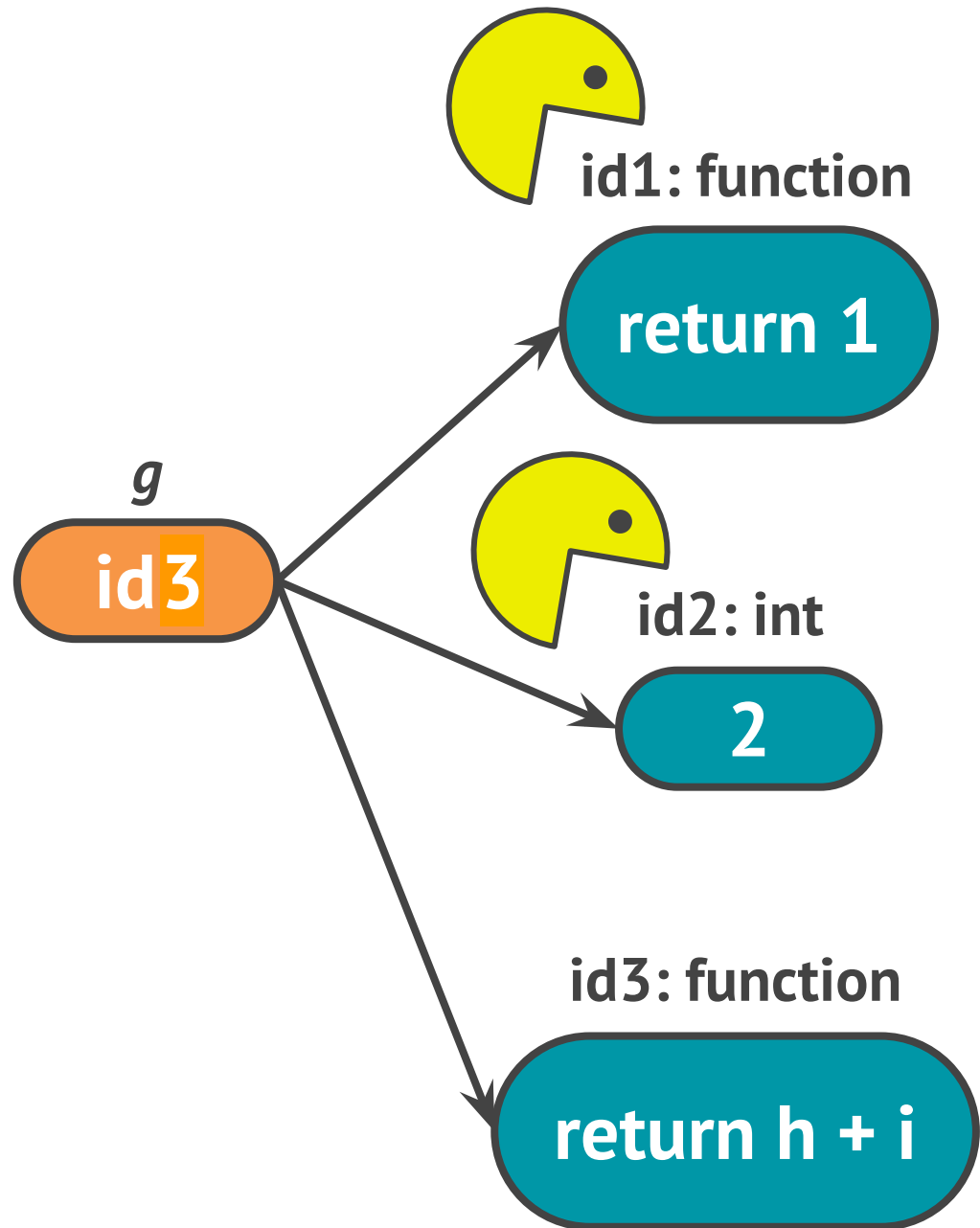
```
>>> g
```

2

```
>>> def g(h, i):  
    return h + i
```

```
>>> g(1, 2)
```

3



Почему полезно создавать **ЧИСТЫЕ**
функции в Python?

Ответ: проще производить вычисления

чистая функция



```
>>> def far_to_cels(far):  
    return (far-32) * 5/9
```

```
>>> far_to_cels(80)  
26.666666666666666668
```

```
>>>
```

нечистая функция

```
>>> def far_to_cels(far):  
    print(far) ●  
    return (far-32) * 5/9
```

```
>>> far_to_cels(80)  
26.666666666666666668
```

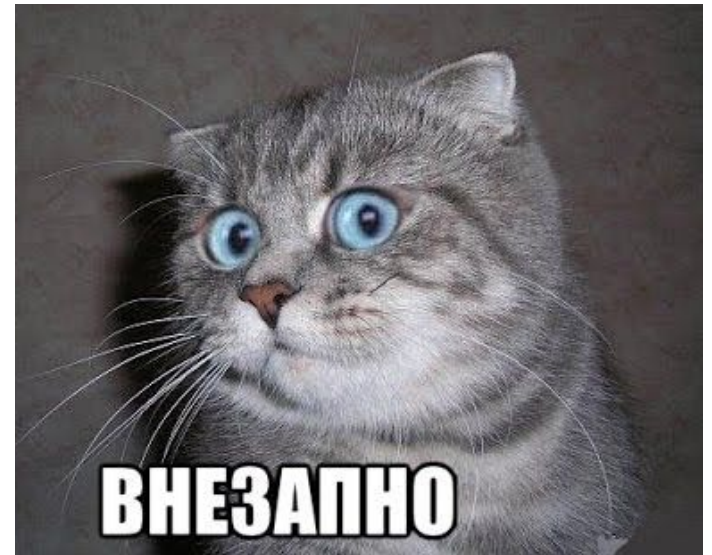
```
>>>
```


Функция в Python всегда что-то возвращает и по умолчанию это объект `None`.

```
def func(x):  
    y = x + 5
```

```
# значение функции от 3  
print(func(3) + 1)
```

Если в функции не
укажем **return**, то
Python подставит
return None





<https://dfedorov.spb.ru>