

# КАКАЯ ОШИБКА, ЕСЛИ...

...в коде пропущено двоеточие, отступ, закрывающаяся скобка или закрывающиеся кавычки?



**SYNTAX ERROR**

# КАКАЯ ОШИБКА, ЕСЛИ...

...в коде используется переменная, название и значение которой не были ранее объявлены в программе?



**NAME ERROR**



# КАКАЯ ОШИБКА, ЕСЛИ...

...программа работает слишком долго и, возможно, попала в бесконечный цикл?

**TL (TIME LIMIT)**



# КАКАЯ ОШИБКА, ЕСЛИ...

...результат вашей программы не соответствует эталонному результату, заложенному в EduApp?

**WA (WRONG ANSWER)**





# РЕБУС



**time limit**

# ЦЕЛИ ЗАНЯТИЯ

- Вспомнить, что такое функция
- Обсудить, как функции делают нашу жизнь проще
- Научиться создавать свои функции





# ФУНКЦИЯ

**Функция** – это законченный фрагмент кода, который выполняет какую-то небольшую задачу, и который можно вызвать из основной программы любое количество раз



# ПРИМЕРЫ ФУНКЦИЙ

print()

int()

map()

input()

float()

round()

str()



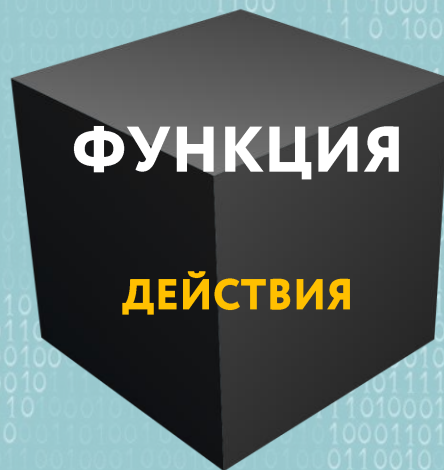
# ФУНКЦИИ ПОМОГАЮТ...

- Не копировать один и тот же код много раз
- Создавать код под конкретные задачи
- Делать код более читаемым и понятным
- Разделять обязанности в команде при работе над большим проектом
- Использовать код, написанный другими



# ПРИНЦИП РАБОТЫ

**АРГУМЕНТЫ**



**ВОЗВРАЩАЕМОЕ  
ЗНАЧЕНИЕ**





# ПРИНЦИП РАБОТЫ



# ПРИМЕР ФУНКЦИИ





# СОЗДАНИЕ ФУНКЦИИ

ключевое слово для создания функции → `def`

название функции → `power`

параметры в скобках через запятую → `(num, pow):`

двоеточие → `:`

ключевое слово для возврата значения → `return`

отступ → `result = 1`  
`for i in range(pow):`  
`result *= num`

тело функции → `result`

возвращаемое значение → `result`

```
def power(num, pow):  
    result = 1  
    for i in range(pow):  
        result *= num  
    return result
```



# ВЫЗОВ ФУНКЦИИ

```
def power(num, pow):  
    result = 1  
    for i in range(pow):  
        result *= num  
    return result
```

сначала определение,  
потом вызов

`power(4, 2)`

} функция  
не выполняется,  
а только  
сохраняется  
в памяти

количество  
аргументов  
должно  
совпадать  
с количеством  
параметров



# ВЫЗОВ ФУНКЦИИ

```
def power(num, pow):  
    result = 1  
    for i in range(pow):  
        result *= num  
    return result
```

```
power(4, 2)
```



The diagram consists of two arrows. A yellow arrow originates from the number '4' in the function call 'power(4, 2)' and points diagonally upwards and to the right to the parameter 'num' in the function definition 'def power(num, pow)'. An orange arrow originates from the number '2' in the function call and points diagonally upwards and to the right to the parameter 'pow' in the function definition.

# ВЫЗОВ ФУНКЦИИ

```
def power(4, 2):  
    result = 1  
    for i in range(2):  
        result *= 4  
    return result  
  
power(4, 2)
```



# ВЫЗОВ ФУНКЦИИ

```
def power(4, 2):  
    result = 1  
    for i in range(2):  
        result *= 4  
    return result
```

power(4, 2)



# ВЫЗОВ ФУНКЦИИ

```
def power(4, 2):  
    result = 1  
    for i in range(2):  
        result *= 4  
    return result
```

16





# ВЫЗОВ ФУНКЦИИ

```
def power(num, pow):  
    result = 1  
    for i in range(pow):  
        result *= num  
    return result  
  
square = power(4, 2)
```

# ВЫЗОВ ФУНКЦИИ

```
def power(num, pow):  
    result = 1  
    for i in range(pow):  
        result *= num  
    return result
```

↙  
square = power(4, 2)



# ВЫЗОВ ФУНКЦИИ


```
def power(num, pow):  
    result = 1  
    for i in range(pow):  
        result *= num  
    return result
```

square = 16



# ВЫЗОВ ФУНКЦИИ


```
def power(num, pow):  
    result = 1  
    for i in range(pow):  
        result *= num  
    return result  
  
square = 16  
print(square)
```





# ВЫЗОВ ФУНКЦИИ

```
def power(num, pow):  
    result = 1  
    for i in range(pow):  
        result *= num  
    return result  
  
print(power(4, 2))
```



# ВЫЗОВ ФУНКЦИИ

```
def power(num, pow):  
    result = 1  
    for i in range(pow):  
        result *= num  
    return result
```

  
print(16)



# ВЫЗОВ ФУНКЦИИ

сначала определение,  
потом вызовы

```
def power(num, pow):  
    result = 1  
    for i in range(pow):  
        result *= num  
    return result
```

функция  
не выполняется,  
а только  
сохраняется  
в памяти

```
square = power(4, 2)  
print(square)  
print(power(square, 2))
```

количество  
аргументов  
должно  
совпадать  
с количеством  
параметров

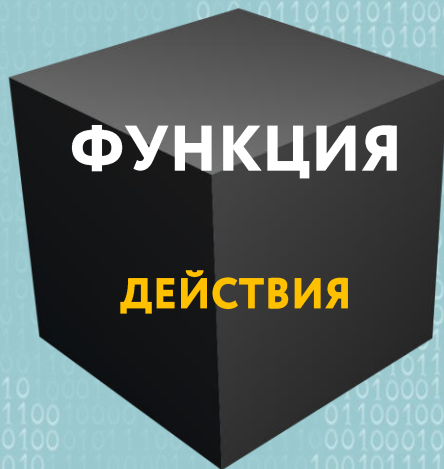


**ТРИ ВОПРОСА  
НА ЗАСЫПКУ!**



# ВОПРОС №1

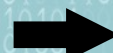
**АРГУМЕНТЫ**



**ВОЗВРАЩАЕМОЕ  
ЗНАЧЕНИЕ**

**МОЖНО ЛИ НЕ ПОДАВАТЬ  
В ФУНКЦИЮ АРГУМЕНТЫ?**

# ВОПРОС №1

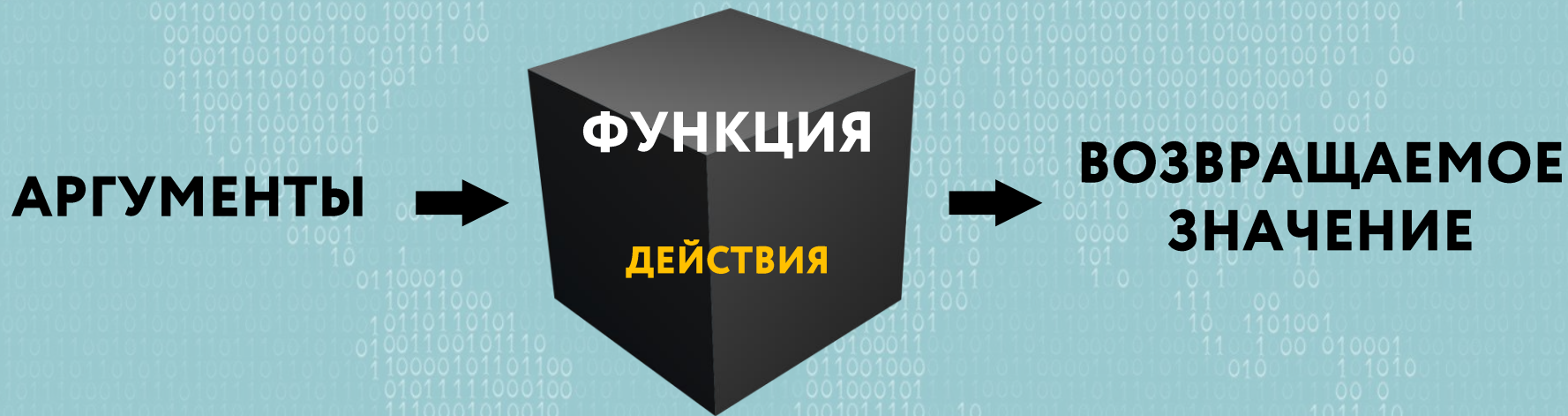


**ВОЗВРАЩАЕМОЕ  
ЗНАЧЕНИЕ**

**ДА!**



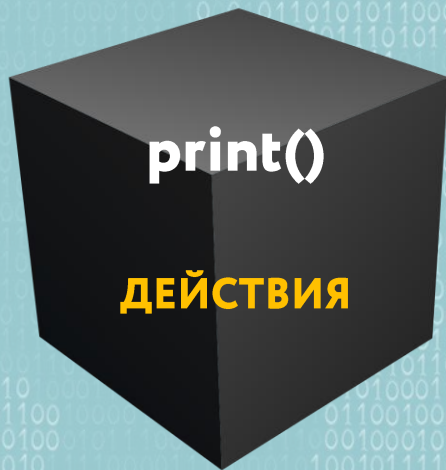
# ВОПРОС №2



**МОЖНО ЛИ НЕ ВОЗВРАЩАТЬ  
ИЗ ФУНКЦИИ **ЗНАЧЕНИЕ**?**

# ВОПРОС №2

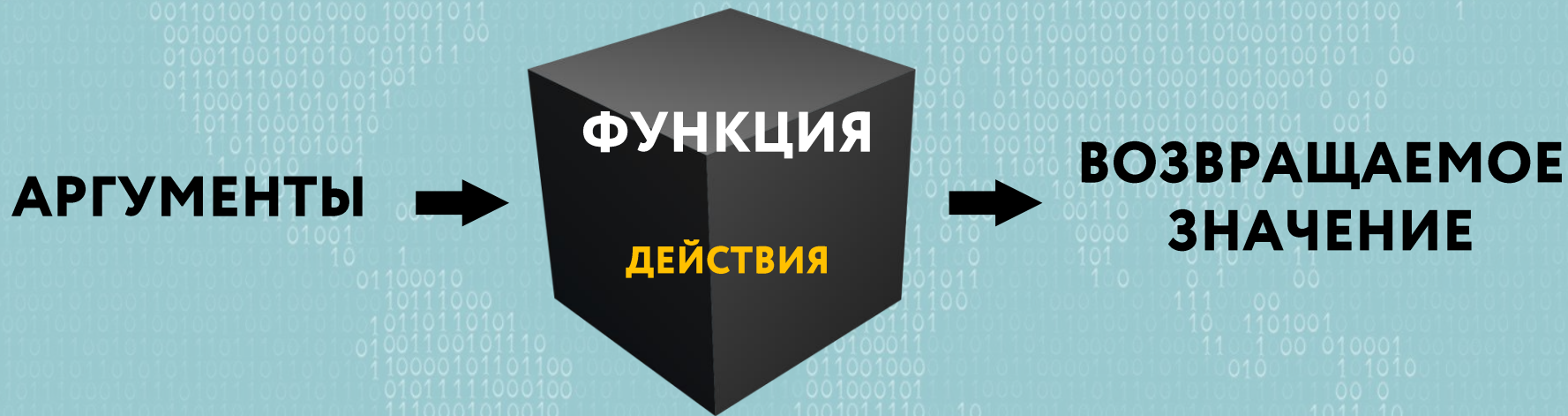
**АРГУМЕНТЫ**



**ДА!**

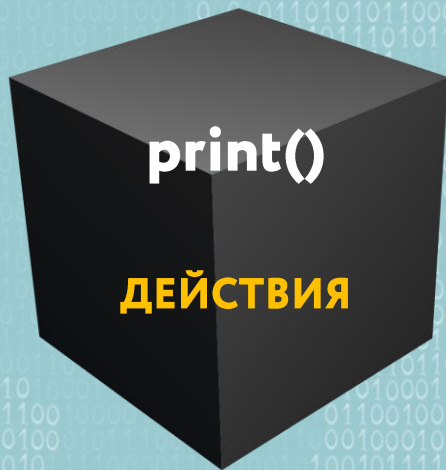


# ВОПРОС №3



**МОЖНО ЛИ ОБОЙТИСЬ  
И БЕЗ АРГУМЕНТОВ, И БЕЗ ВОЗВРАТА ЗНАЧЕНИЯ?**

# ВОПРОС №3



**ДА!**



# ОБЛАСТИ ВИДИМОСТИ

```
def func():  
    k = x + 2
```

```
x = 10
```

```
func()
```

```
print(k)
```

**ЧТО БУДЕТ НАПЕЧАТАНО НА ЭКРАНЕ?**

1. 12

2. Name Error: name 'x' is not defined

3. Name Error: name 'k' is not defined

# ОБЛАСТИ ВИДИМОСТИ

```
def func():  
    k = x + 2
```

```
x = 10  
func()  
print(k)
```

Name Error: name 'k' is not defined





# ОБЛАСТИ ВИДИМОСТИ

```
def func():  
    k = x + 2
```

локальная  
переменная

область  
видимости  
функции

глобальная  
переменная

```
x = 10  
func()  
print(k)
```

глобальная  
область  
видимости



# ИТОГИ ЗАНЯТИЯ

- Обсудили, зачем нужны функции
- Научились создавать свои собственные функции
- Узнали про области видимости и локальные переменные
- Узнали, что такое декомпозиция

**Вопросы?**