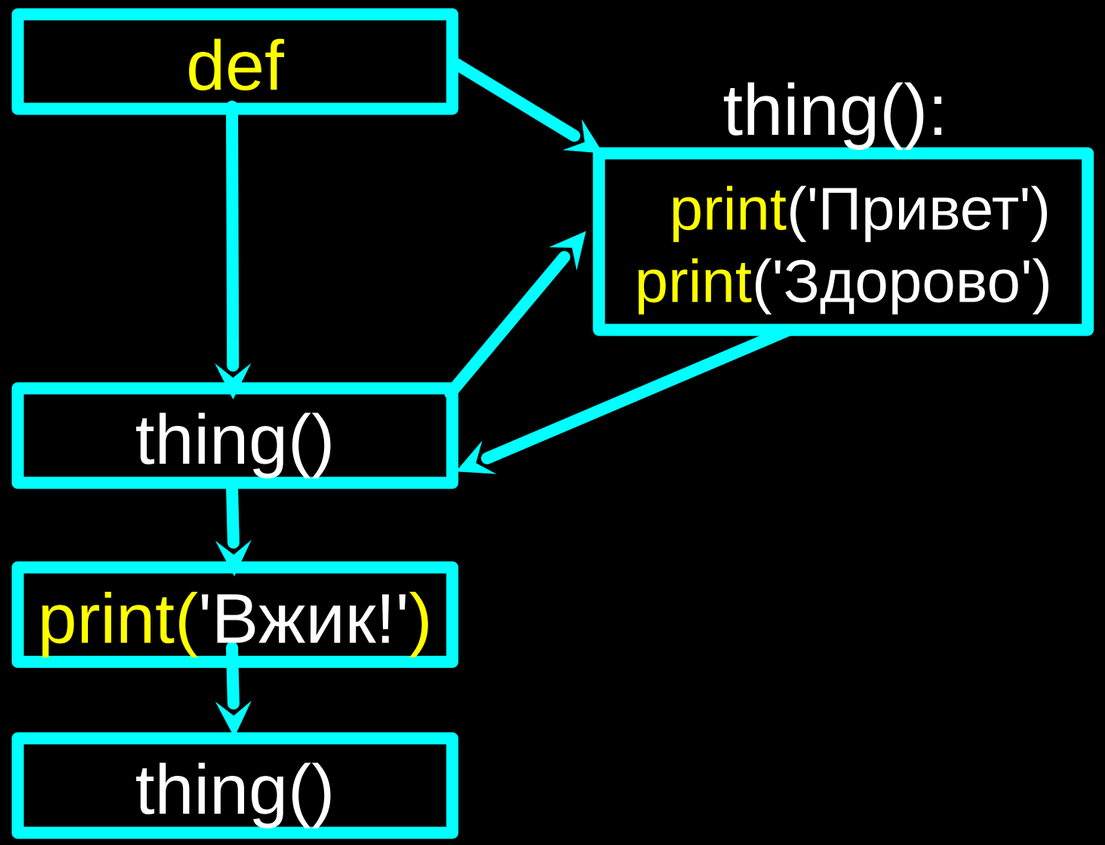
Функции

Глава 4

Пайтон для всех

www.py4e.com

Хранимые (и повторно используемые) Шаги



Программа:

```
def thing():

print('Привет')

print('Здорово')

thing()
print('Вжик!')
thing()
```

Результат:

Привет
Здорово
Вжик!
Привет
Здорово

Мы называем эти повторно используемые части «функциями»

Функции в Пайтон

- В Пайтон два вида функций:

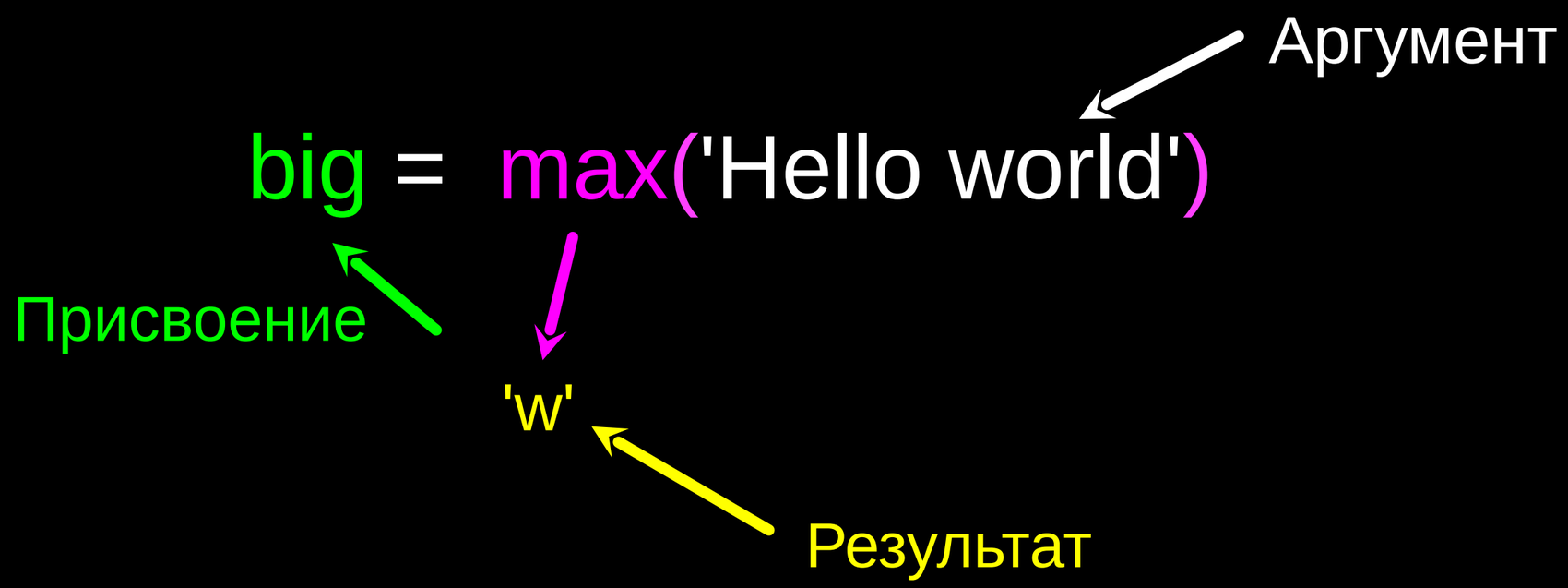
  - Встроенные функции, поставляемые как часть языка Пайтон: print(), input(), type(), float(), int() ...;

  - Функции, которые мы создаем сами и затем используем.

- Стоит воспринимать имена встроенных функций, как «новые» ключевые слова (не используйте их в качестве имен для переменных)

Определение функции

- Функция в Пайтон — это блок повторно используемого кода, который принимает аргумент(ы) в качестве входных данных, производит вычисления и затем возвращает результат(ы)
- Мы создаем функцию, используя специальное слово def
- Мы вызываем функцию, используя имя функции, круглые скобки и аргументы в выражении




```
>>> big = max('Hello world')
>>> print(big)
w
>>> tiny = min('Hello world')
>>> print(tiny)


>>>
```

Функция Max()

```
>>> big = max('Hello world')
>>> print(big)
w
```

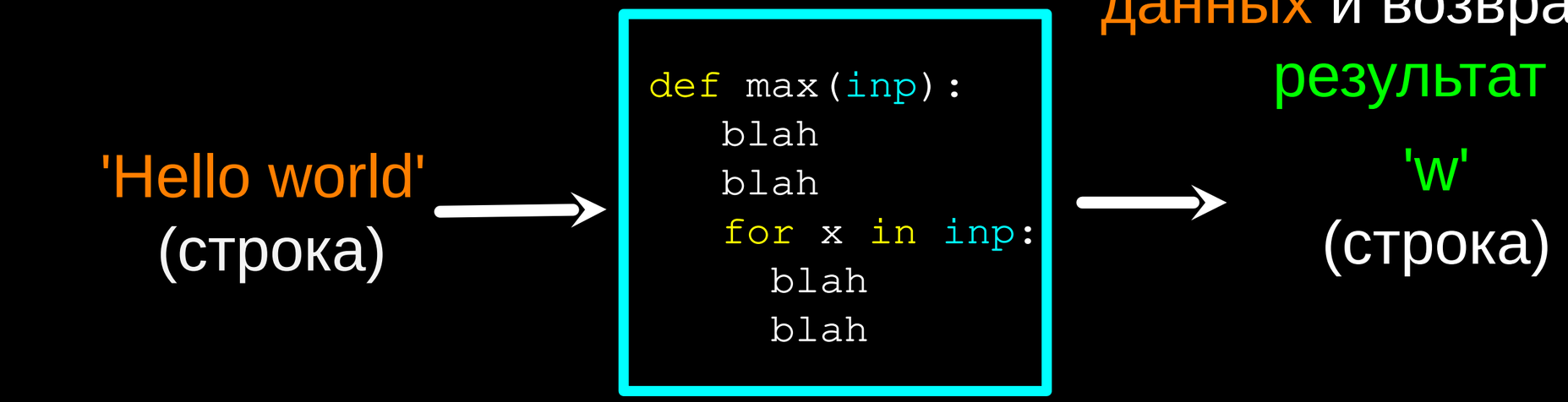
Функция — это некий сохраненный код, который мы используем. Функция получает набор входных данных и возвращает результат

Гвидо написал этот код

Функция Max()

```
>>> big = max('Hello world')
>>> print(big)
w
```

Функция — это некий сохраненный код, который мы используем. Функция получает набор входных данных и возвращает результат



Гвидо написал этот код

Преобразование типов

- Когда вы помещаете в выражение целое число и число с плавающей точкой, целое число преобразуется в число с плавающей точкой
- Вы можете устанавливать и изменять тип при помощи встроенных функций int() и float()

```
>>> print(float(99) / 100)
0.99
>>> i = 42
>>> type(i)
<class 'int'>
>>> f = float(i)
>>> print(f)
42.0
>>> type(f)
<class 'float'>
>>> print(1 + 2 * float(3) / 4 - 5)
-2.5
>>>
```

Преобразование строк

- Вы можете использовать функции int() и float() для преобразований между строками и числами
- В случае, если строка не содержит числовых символов, вы получите сообщение об ошибке

```
>>> sval = '123'
>>> type(sval)
<class 'str'>
>>> print(sval + 1)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str'
and 'int'
>>> ival = int(sval)
>>> type(ival)
<class 'int'>
>>> print(ival + 1)
124
>>> nsv = 'hello bob'
>>> niv = int(nsv)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int()
```

Создание собственных функций…

Создание собственных функций

- Чтобы создать новую функцию, используется ключевое слово def, за которым в скобочках следуют параметры, если необходимо.
- Тело функции требует отступа
- Следующая запись определяет функцию, но не выполняет команды, содержащиеся в ней:

```
def print_lyrics():
    print("Я лесоруб и все пучком.")
    print('Сплю ночью и работаю днем.')
```

print_lyrics():

```
print("Я лесоруб и все пучком.")
print('Сплю ночью и работаю днем.')
```

```
x = 5
print('Привет')

def print_lyrics():
    print("Я лесоруб и все пучком.")
    print('Сплю ночью и работаю днем.')

print('Эй!')
x = x + 2
print(x)
```

Привет
Эй!
7

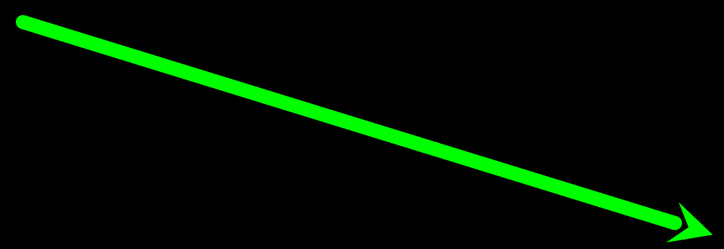
Определение и использование функции

- Как только мы определили/создали функцию, можем вызывать ее столько раз, сколько потребуется
- Это называется хранением и повторным использованием

```
x = 5
print('Привет')

def print_lyrics():
    print("Я лесоруб и все пучком.")
    print('Сплю ночью и работаю днем.')

print('Эй!')
print_lyrics()
x = x + 2
print(x)
```



```
Привет
Эй!
Я лесоруб и все пучком.
Сплю ночью и работаю днем.
7
```

Аргументы

- Аргумент — это значение, которое мы передаем функции в момент вызова в качестве входных данных
- Мы используем аргументы, чтобы при различных вызовах функция могла выполнить разные виды расчетов
- Аргументы помещаем внутрь скобок после имени функции

```
big = max('Привет, мир!')
```

Аргумент

Параметры

Параметр — это переменная, которую мы используем внутри определения функции. Это “дескриптор”, который позволяет коду внутри функции обращаться к аргументам при вызове функции.

```
>>> def greet(lang):
...     if lang == 'es':
...         print('Hola')
...     elif lang == 'fr':
...         print('Bonjour')
...     else:
...         print('Hello')
...
>>> greet('en')
Hello
>>> greet('es')
Hola
>>> greet('fr')
Bonjour
>>>
```

Возвращаемое значение

Функция принимает свои аргументы, производит некоторые вычисления и возвращает значение, которое будет использоваться в качестве значения вызова функции в вызывающем выражении. Для этого используется ключевое слово return.

```
def greet():
    return "Привет, "

print(greet(), "Аня")
print(greet(), "Михаил")
```

```
Привет, Аня
Привет, Михаил
```

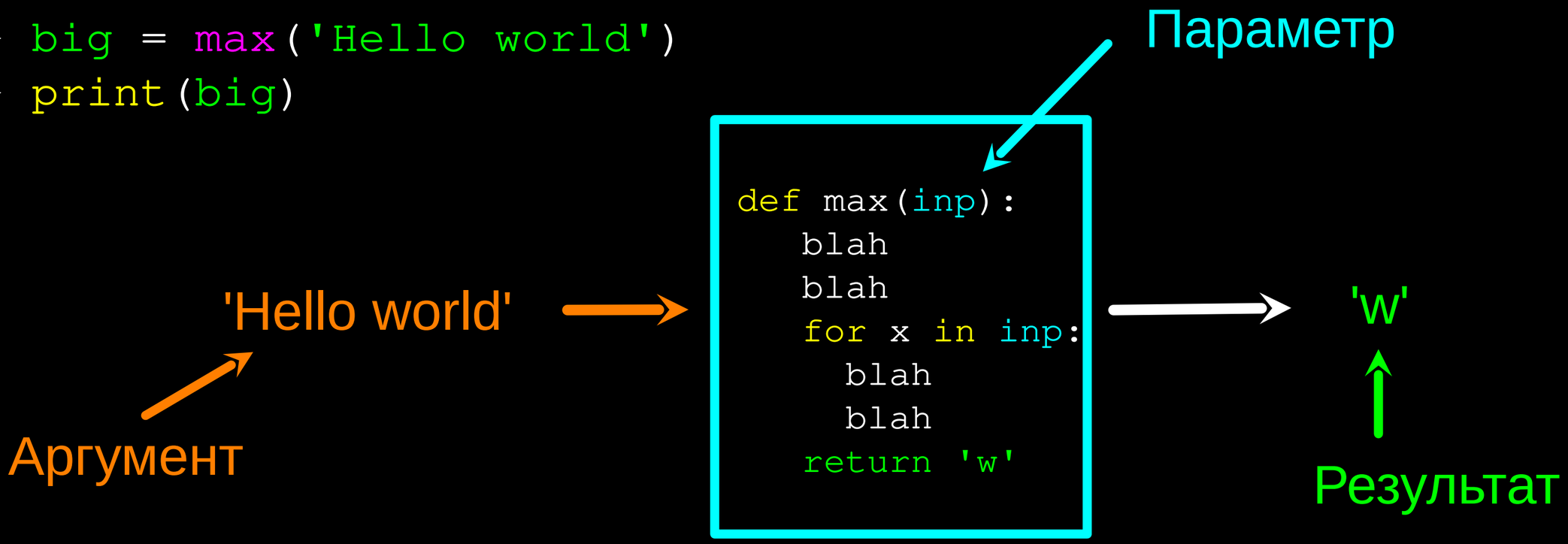
Возвращаемое значение

- «Плодотворная» функция дает результат (или возвращает значение)
- Оператор return завершает выполнение функции и возвращает “назад” результат выполнения функции

```
>>> def greet(lang):
...     if lang == 'es':
...         return 'Hola'
...     elif lang == 'fr':
...         return 'Bonjour'
...     else:
...         return 'Hello'
...
>>> print(greet('en'),'Glenn')
Hello Glenn
>>> print(greet('es'),'Sally')
Hola Sally
>>> print(greet('fr'),'Michael')
Bonjour Michael
>>>
```

Аргументы, Параметры и Результаты

```
>>> big = max('Hello world')
>>> print(big)
w
```



Множественные Параметры / Аргументы

- Мы можем указать больше одного параметра в определении функции
- Мы запросто можем добавить больше аргументов при вызове функции
- Мы сопоставляем количество и порядок аргументов и параметров

```
def addtwo(a, b):
    added = a + b
    return added

x = addtwo(3, 5)
print(x)
```

```
8
```

Пустые (неплодотворные) функции

- Если функция не возвращает значение, мы называем ее «пустой» функцией
- Функции, которые возвращают значение, называются «плодотворными»
- «Пустые» функции не являются «плодотворными»

Создавать или не создавать функцию...

- Организуйте свой код в виде “параграфов”: зафиксируйте мысль и дайте ей название
- Не повторяйтесь. Создайте функцию единожды, а затем повторно используйте по необходимости.
- Если код начинает быть длинным или слишком сложным, разбейте его на логические фрагменты и поместите их в функции
- Создайте библиотеку часто используемого кода. Возможно, вы захотите поделиться ей с друзьями…

Резюме

- Функции
- Встроенные функции
- Преобразование типов (int, float)
- Преобразование строк
- Параметры
- Аргументы
- Результаты (плодотворные функции)
- Пустые (неплодотворные) функции
- Зачем нам функции?

Задание

Перепишите программу расчета оплаты с полуторной ставкой за переработку и создайте функцию с именем computepay, которая принимает два параметра - (количество часов и ставка).

Введите количество часов: 45
Введите ставку: 10

Оплата: 475.0

475 = 40 * 10 + 5 * 15

Авторы / Благодарности

Исходная разработка: Чарльз Северанс, Школа Информации Мичиганского Университета.

Перевод выполнила Фомкина Виолетта.

… Добавьте сюда новых авторов и переводчиков

...