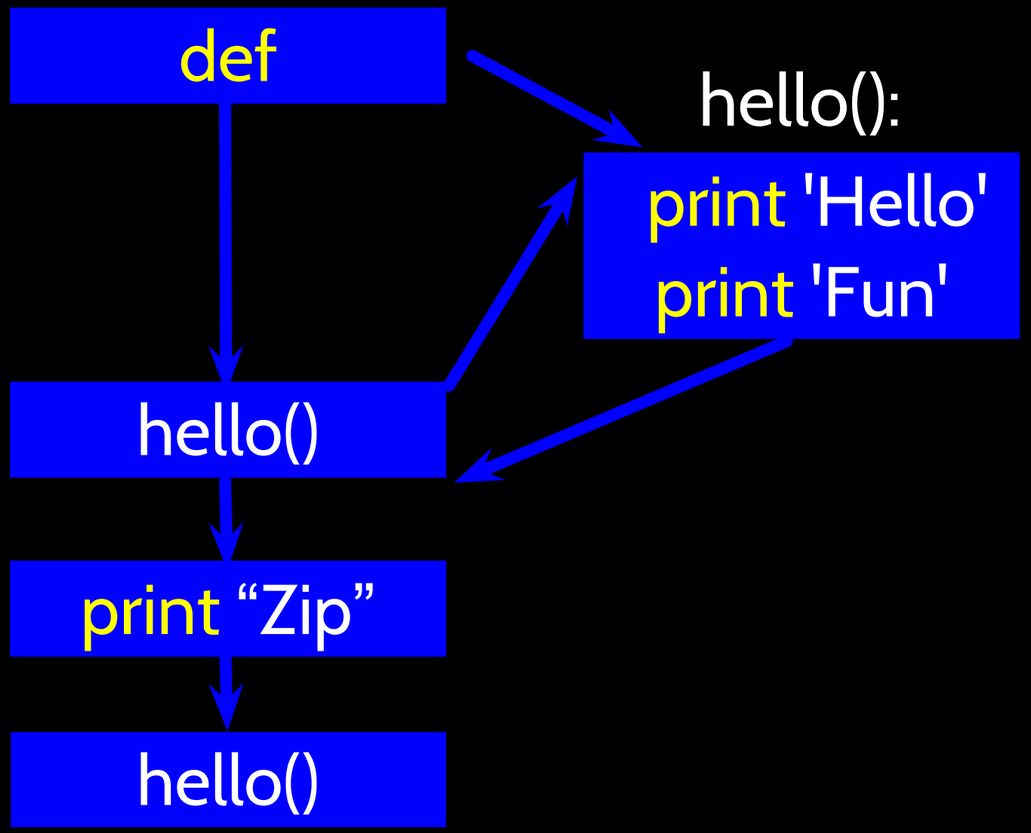
Функции

Глава 4

Python for Informatics: Exploring Information
www.pythonlearn.com

Сохраненные (и многократно используемые) шаги



hello():

```
print 'Hello'
print 'Fun'
```

Программа:

```
def thing():
    print 'Hello'
    print 'Fun'

thing()
print 'Zip'
thing()
```

Результат:

```
Hello
Fun
Zip
Hello
Fun
```

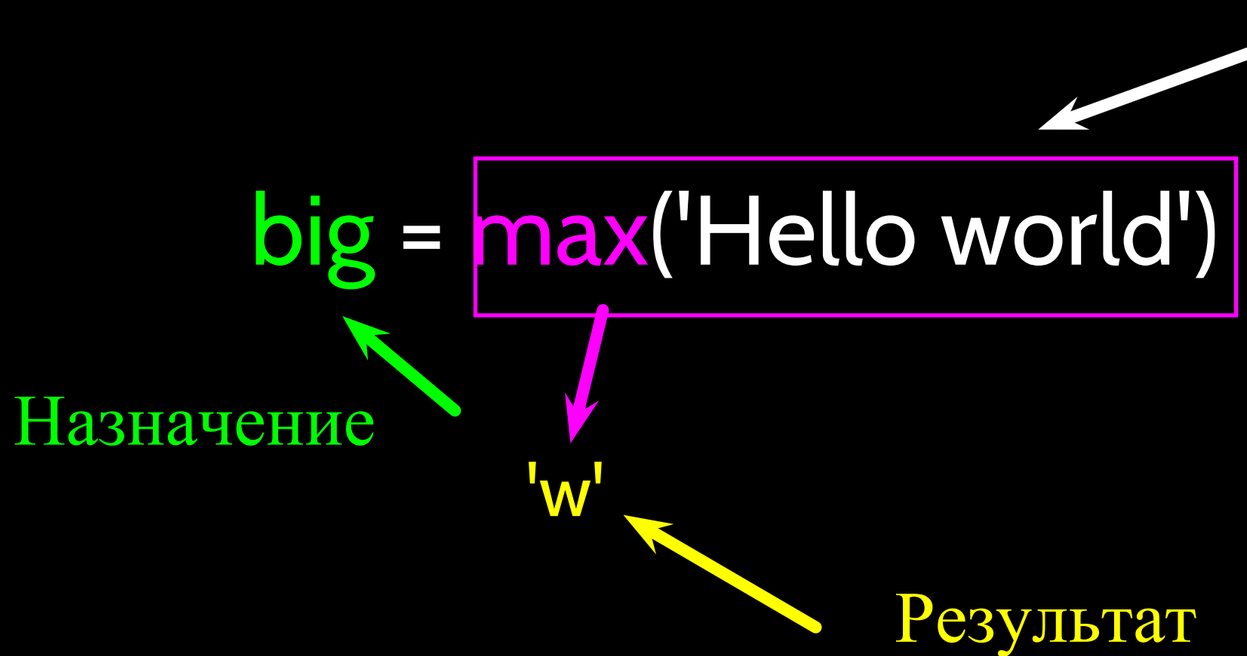
Многократно используемые части кода мы называем “функциями”

Функции Python

- Python содержит два вида функций.
  - Встроенные функции, которые предоставляются в рамках языка Python: raw_input(), type(), float(), int() ...
  - Функции, которые мы задаем самостоятельно и затем используем
- С названиями встроенных функций мы обращаемся так же, как и с зарезервированными словами (т.е. мы не используем их в названиях переменных)

Определение функций

- В языке Python под функцией понимается определенный код многократного использования, принимающий некоторые аргументы на входе, выполняющий определенные вычисления и выдающий результат или результаты на выходе
- Мы задаем функцию с помощью зарезервированного слова def
- Мы вызываем/обращаемся к функции с помощью ее названия, скобок и аргументов




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

>>>
```

Функция max

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

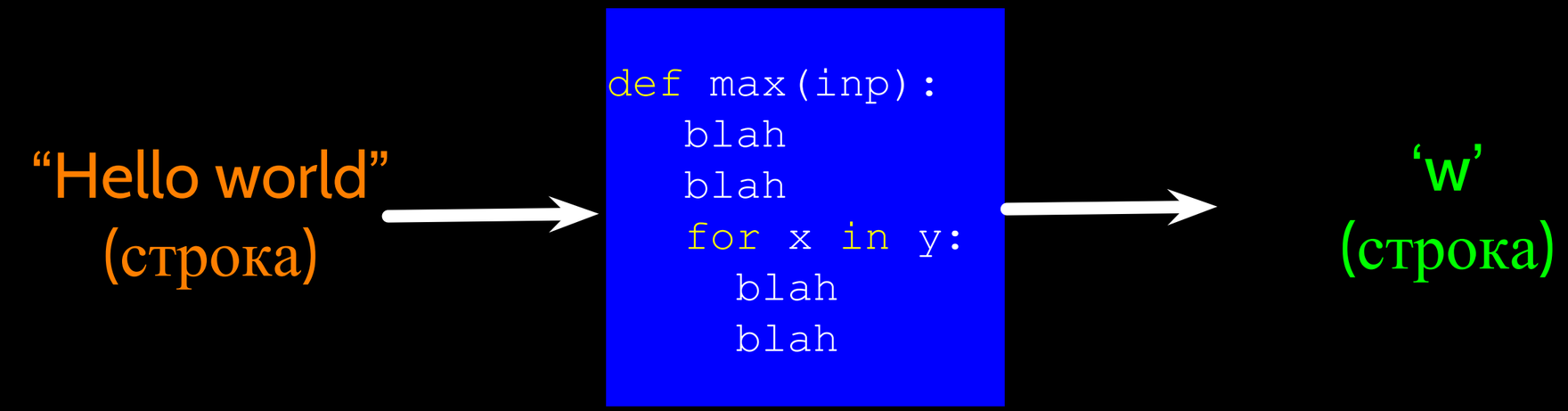
Функция - это используемый нами сохраненный код. Функция принимает входные данные и производит результат.

Этот код написал г-н Гвидо

Функция max

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

Функция - это используемый нами сохраненный код. Функция принимает входные данные и производит результат.



Этот код написал г-н Гвидо

Преобразование типа

- При использовании целого числа и числа с плавающей точкой в одном выражении, целое число косвенно преобразуется в значение с плавающей точкой
- Это управляется с помощью встроенных функций, таких как int() и float()

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

Преобразование строк

- Функции int() и float() можно также использовать для преобразования строк в числа
- Если строка не содержит числовых значений, появится ошибка

```
>>> sval = '123'
>>> type(sval)
<type '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)
<type '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 личных функций

- Мы создаем новую функцию с помощью зарезервированного слова def и при необходимости указываем параметры в скобках
- Тело функции выделяется отступами
- Следующий код задает функцию, но *не* выполняет тело функции

```
def print_lyrics():
    print "I'm a lumberjack, and I'm okay."
    print 'I sleep all night and I work all day.'
```

print_lyrics():

```
print "I'm a lumberjack, and I'm okay."
print 'I sleep all night and I work all day.'
```

```
x = 5
print 'Hello'

def print_lyrics():
    print "I'm a lumberjack, and I'm okay."
    print 'I sleep all night and I work all day.'

print 'Yo'
x = x + 2
print x
```

```
Hello
Yo
7
```

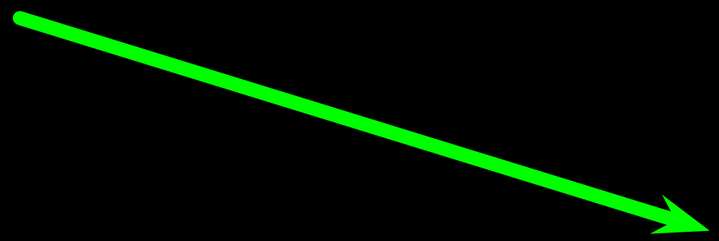
Определения и использование

- Задав функцию, мы можем вызвать (или обратиться) к ней любое количество раз
- Это пример сохранения и многократного использования

```
x = 5
print 'Hello'

def print_lyrics():
    print "I'm a lumberjack, and I'm okay."
    print 'I sleep all night and I work all day.'

print 'Yo'
print_lyrics()
x = x + 2
print x
```



```
Hello
Yo
I'm a lumberjack, and I'm okay.
I sleep all night and I work all day.
7
```

Аргументы

- Аргументом является значение, которое мы передаем в качестве входного параметра при вызове функции
- Аргументы используются для выполнения функции с различными значениями
- Аргументы указываются в скобках после названия функции

```
big = max('Hello world')
```

Аргумент

Параметры

Параметр - это переменная, которую мы используем в определении функции. Это своего рода "ручка", предоставляющая коду в функции доступ к аргументам для вызова конкретной 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
>>>
```

Возвращаемые значения

Зачастую функция принимает на входе аргументы, выполняет расчеты и возвращает значение, которое используется как значение при вызове функции в выражении вызова. Для этого используется ключевое слово return.

```
def greet():
    return "Hello"

print greet(), "Glenn"
print greet(), "Sally"
```

```
Hello Glenn
Hello Sally
```

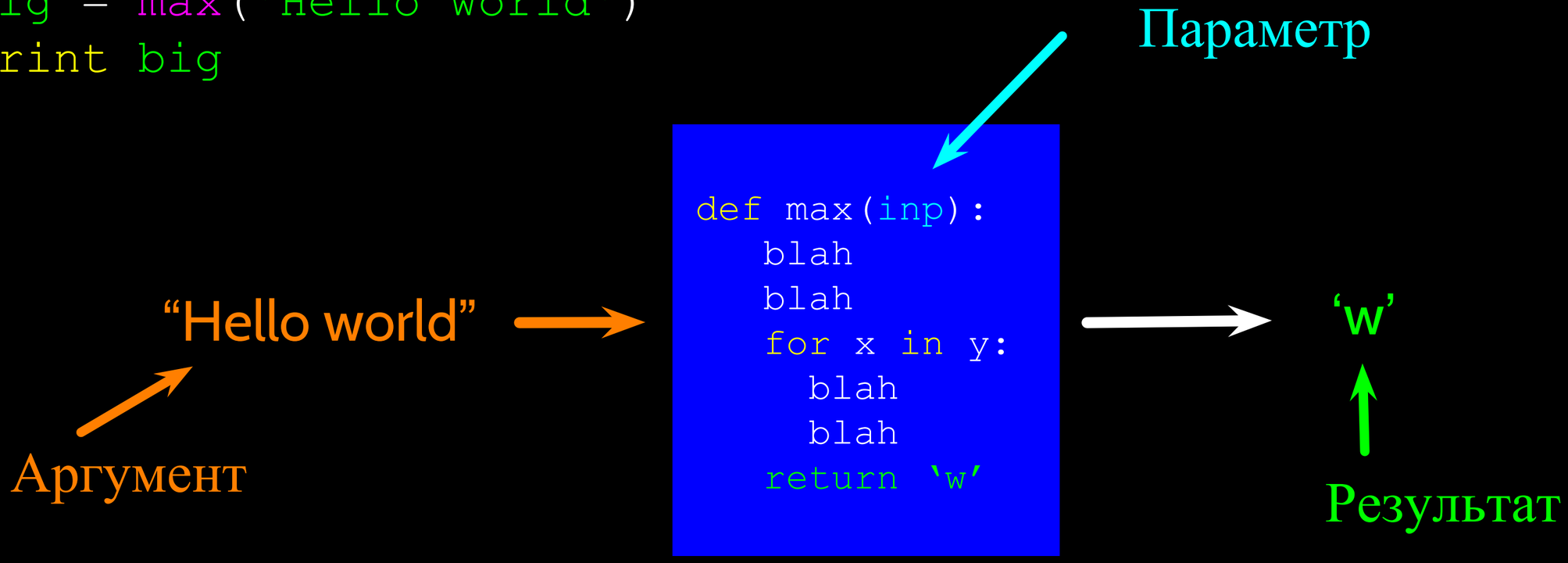
Возвращаемое значение

- “Результативной” является функция, производящая результат (или возвращаемое значение)
- Инструкция return заканчивает исполнение функции и “возвращает” результат 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
```



Несколько параметров / аргументов

- В определении функции можно задать несколько параметров
- Для этого просто добавляем дополнительные аргументы при вызове функции
- Число и порядок аргументов должны совпадать

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

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

Функции типа void (нерезультативные)

- Если функция не производит результат, ее называют функцией типа void
- Функции, производящие результат, являются “результативными”
- Функции типа void являются “нерезультативными”

Использовать функцию или нет...

- Разделите код по “абзацам” - закончите определенную мысль и “назовите ее”
- Избегайте повторов - создайте рабочий код один раз и многократно используйте его
- Если какая-то часть кода стала слишком длинной или сложной, разбейте ее на логические секции и поместите эти секции в функции
- Создайте библиотеку наиболее употребляемых кодов. При желании поделитесь этой библиотекой с друзьями...

Упражнение

Измените код расчета заработной платы с учетом того, что ставка за сверхурочные часы в полтора раза выше обычной ставки, и создайте функцию под названием computepay, которая принимает два параметра (часы и ставка).

Введите часы: 45
Введите ставку: 10
Зарплата: 475.0

475 = 40 * 10 + 5 * 15

Обзор

- Функции
- Встроенные функции
  - Преобразование типа (int, float)
- Аргументы
- Параметры

Благодарность / Содействие

Первоначальная разработка: Чарльз Северанс, Школа информации Мичиганского университета

Здесь впишите дополнительных авторов и переводчиков...

...