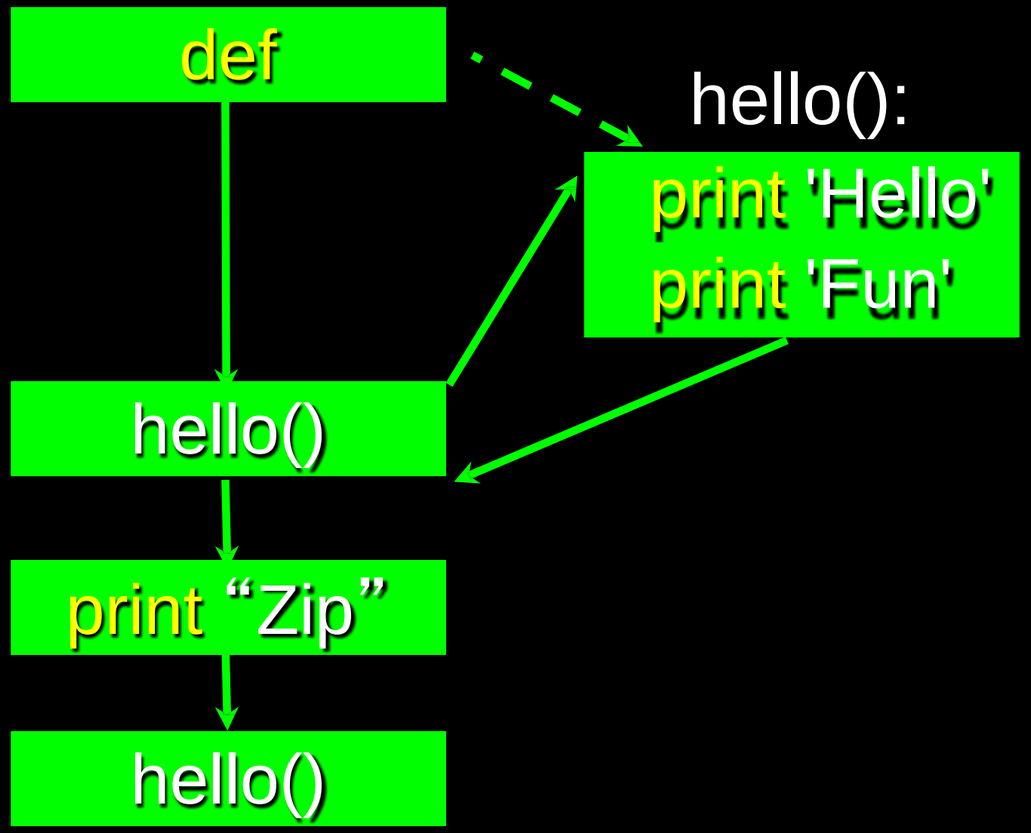
Python - Functions

Stored (and reused) Steps



hello():

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

Program:

```
def hello():
    print 'Hello’
    print 'Fun’

hello()
print 'Zip’
hello()
```

Output:

```
Hello
Fun
Zip
Hello
Fun
```

These reusable pieces of code are called “functions”.

Python Functions

-- There are two kinds of functions in Python

-- Built-in functions, which are provided by Python - raw_input(), type(), float(), int() ...

-- Functions, which are user-defined

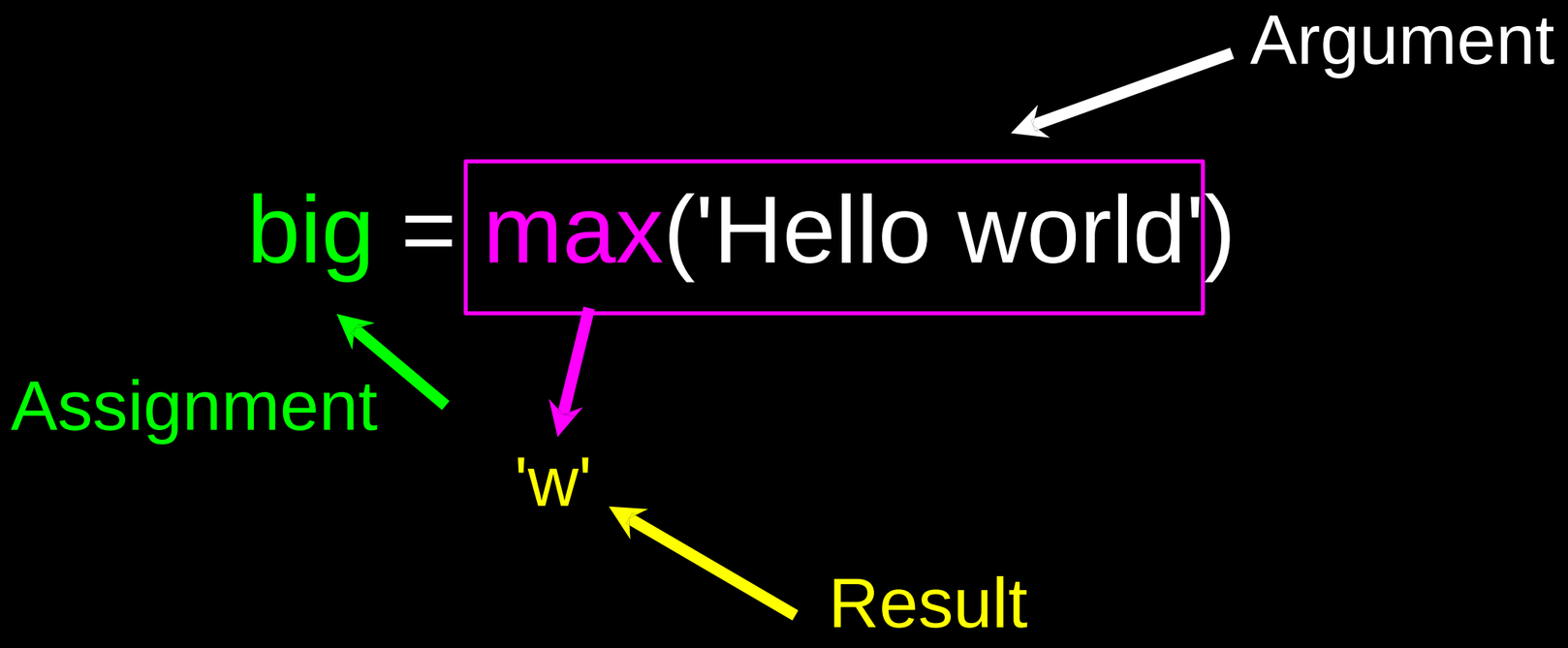
-- Built-in function names treated as “new” reserved words

Function Definition

-- In Python, a function is reusable code, which takes arguments(s) as input does some computation and then returns a result(s)

-- Define a function using the def reserved word

-- Call/invoke the function by using the function name, parenthesis and arguments in an expression

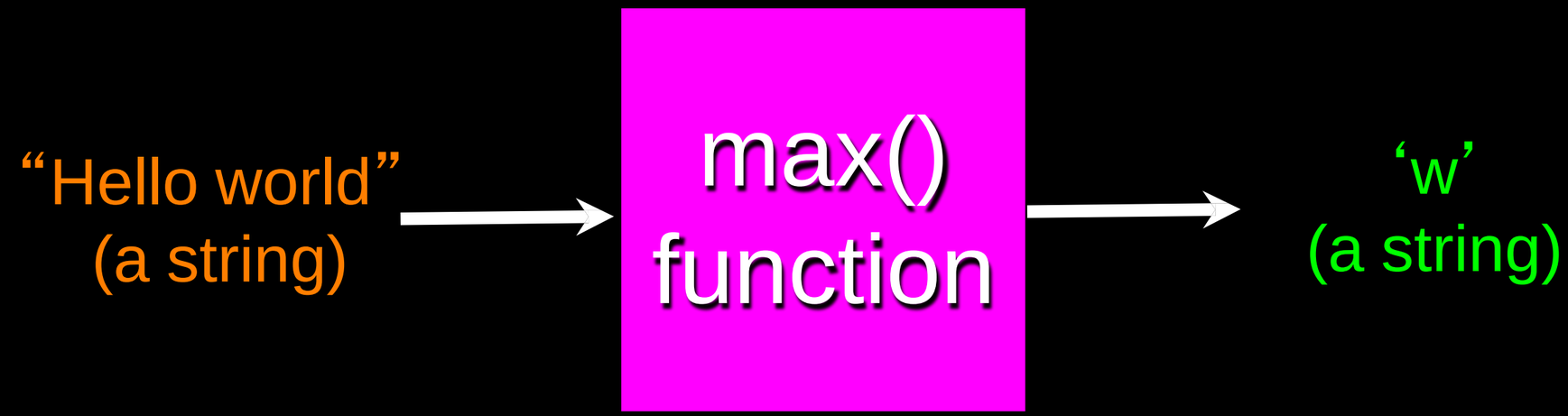



```
>>> big = max('Hello world')
>>> print big
w
>>> tiny = min('Hello world')
>>> print tiny
 
```

Max Function

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

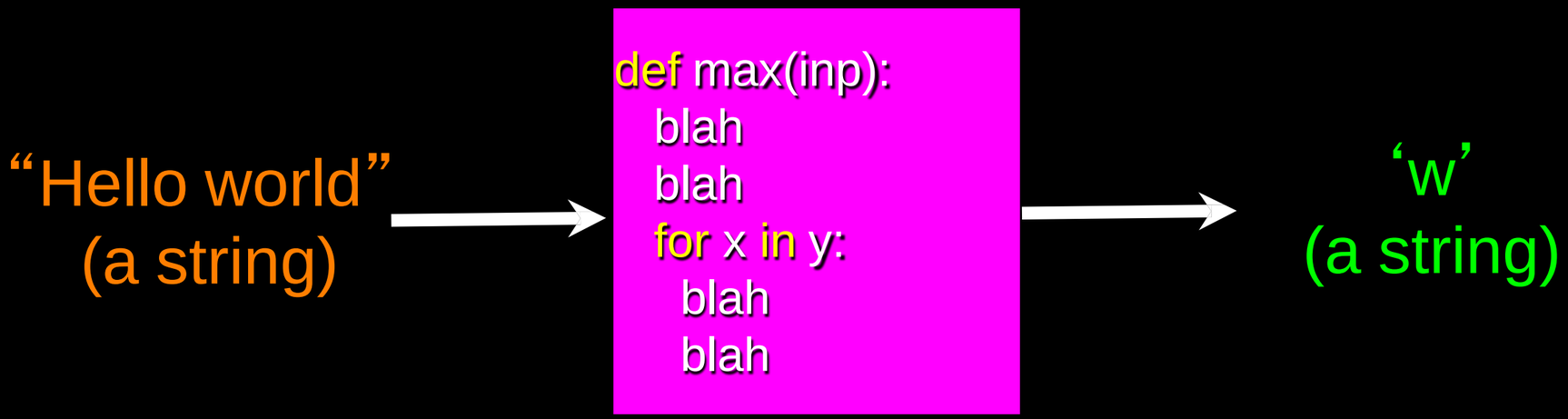
A function is reusable code, which takes some input and produces an output



Max Function

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

A function is reusable code, which takes some input and produces an output



Type Conversions

-- In an integer and floating point in an expression, the integer is implicitly converted to a float

-- You can control this with the built in functions int() and 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
>>>
```

String Conversions

-- Use int() and float() to convert between strings and integers

-- An error will occur if the string does not contain numeric characters

```
>>> 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()
```

Building our Own Functions

-- To create a new function, the def keyword is used followed by optional parameters in parenthesis

-- Indent the body of the function

-- This defines the function but *does not* execute the body of the function

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

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

print_lyrics():

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

```
Hello
Yo
7
```

Definitions and Uses

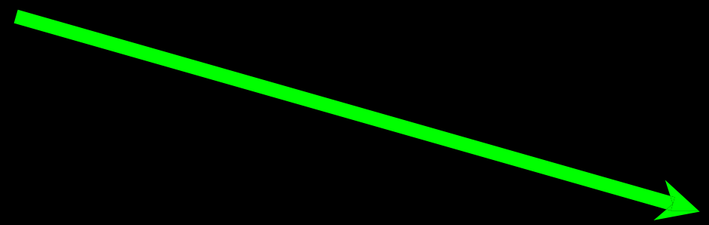
-- Once a function is defined, we can call (or invoke) it as many times

-- This is the store and reuse pattern

```
x = 5
print 'Hello'

def print_lyrics():
    print "I'm a student at ECSU, 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 student at ECSU, and I'm okay.  I sleep all night and I work all day.
7
```

Arguments

-- An argument is a value passed to the function as its input when the function is called

-- Arguments direct the function to do different kinds of work when called different times

-- Arguments are placed in parenthesis after the name of the function

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

Argument

Parameters

-- A parameter is a variable, which is used in the function definition as a “handle” to allow the function to access the arguments for a particular function invocation

```
>>> 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
>>>
```

Return Values

-- Often a function will take arguments, do some computation and return a value to be used by the that function called it; this is supported by the return keyword

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

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

```
Hello Glenn
Hello Sally
```

Return Value

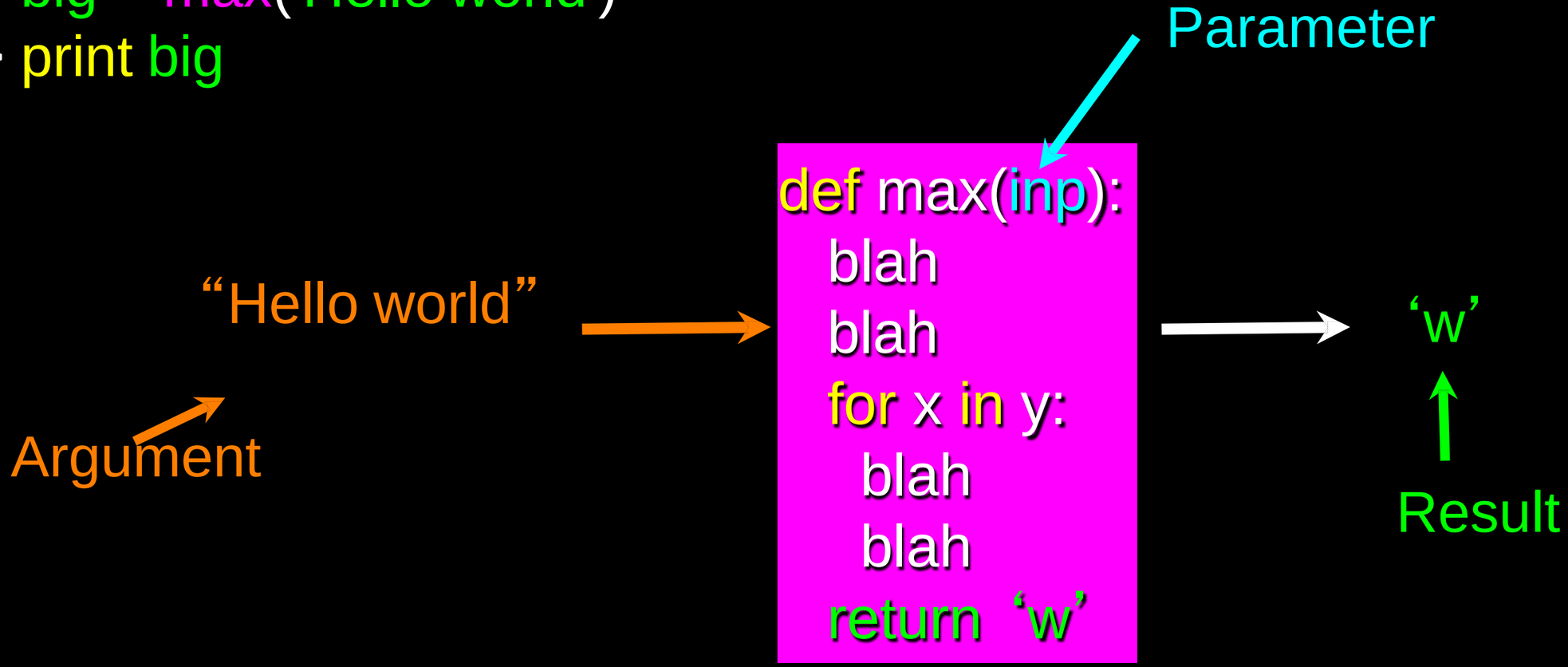
-- A “fruitful” function is one that produces a result (or return value)

-- The return statement ends the function execution and “sends back” the result of the function

```
>>> 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
>>>
```

Arguments, Parameters, and Results

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



Multiple Parameters / Arguments

-- More than one parameter in the function definition can be defined

-- To do this, simply add more arguments when the function is called

-- The number and order of arguments and parameters are matched

```
def addtwo(a, b):
    added = a + b
    return added
x = addtwo(3, 5)
print x
```

Void (non-fruitful) Functions

-- If a function does not return a value, it is called a "void" function

-- Functions, which return values are called "fruitful" functions

-- Void functions are "not fruitful"

To function or not to function...

Organize your code into “paragraphs” - capture a complete thought and “name it”

Don’t repeat yourself - make it work once and then reuse it

If something gets too long or complex, break up logical chunks and put those chunks in functions

Make a library of common stuff that you do over and over - perhaps share this with your friends...

Exercise

Rewrite your pay computation with time-and-a-half for overtime and create a function called computepay which takes two parameters (hours and rate).

Enter Hours: 45
Enter Rate: 5.15
Pay: 281.00

281.00= 40 * 5.15 + 5 * 15