

# Functions

## Lecture 5

# len Function

```
>>> fruit = 'banana'
>>> x = len(fruit)
>>> print(x)
6
```

Function:  
some stored code

'banana'  
(a string)



len()  
function



6  
(a number)

# len Function

```
>>> fruit = 'banana'
>>> x = len(fruit)
>>> print(x)
6
```

A function is some stored code that we use. A function takes some input and produces an output.

'banana'  
(a string)



```
def len(par):
    blah
    blah
    do sth to par
    Return 6
```

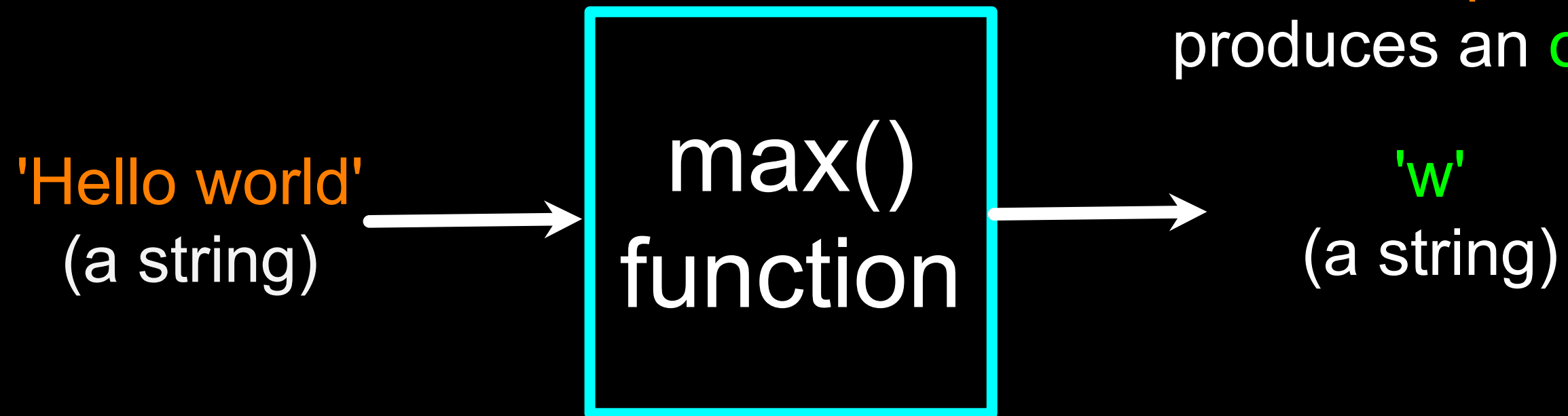


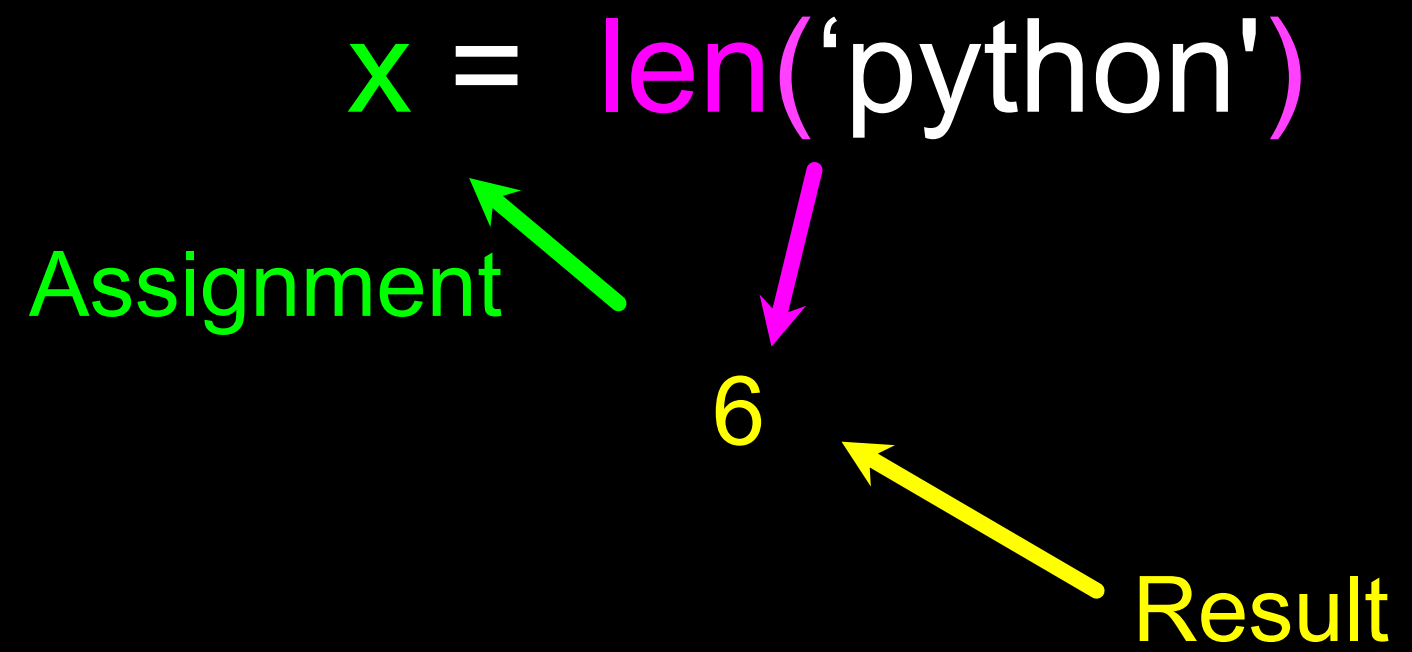
6  
(a number)

# Max Function

A function is some stored code that we use. A function takes some input and produces an output.

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





```
>>> x = len('python')
>>> print(x)
6
```

# Python Functions

- There are two kinds of functions in Python.
  - **Built-in functions** that are provided as part of Python - print(), input(), type(), float(), int() ...
  - **Functions that we define ourselves** and then use

Functions of Our Own...

# Building our Own Functions

keyword      Function name      Optional parameters

↓           ↓           ↓

```
def myfunction():  
    print("This is my first function.")  
    print('A test to see if it works.')
```

myfunction():

```
print("This is my first function.")  
print('A test to see if it works.')
```



Define the function  
but **does not execute** the  
body of the function

```
x = 5  
print( 'Hello' )
```

```
def myfunction():  
    print("This is my first function.")  
    print('A test to see if it works.')
```

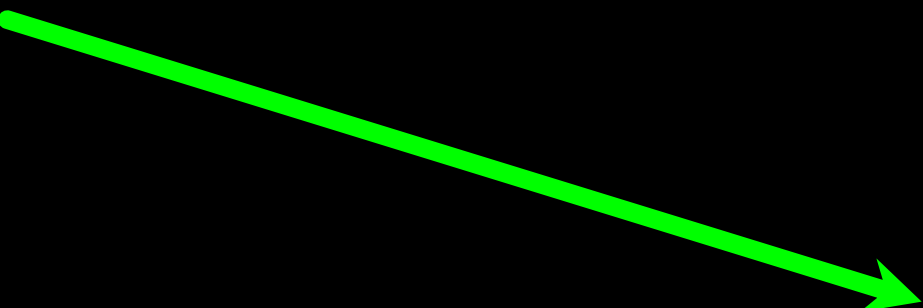
```
print( 'Bye' )  
x = x + 2  
print(x)
```

Output:  
Hello  
Bye  
7

```
x = 5  
print( 'Hello' )
```

```
def myfunction():  
    print("This is my first function.")  
    print('A test to see if it works.')
```

```
print( 'Bye' )  
myfunction()  
x = x + 2  
print(x)
```



Hello  
Bye  
This is my first function.  
A test to see if it works.  
7

# Functions That Return Values

keyword      Function name      Optional parameters      returns values and exits

↓                   ↓                   ↓

```
def test_return():  
    print("This is my first function.")  
    print('A test to see if it works.')  
    Return 6
```

```
x = test_return()  
print(x)
```

This is my first function.  
A test to see if it works.  
6

# Functions That Return Values

```
def short_word():  
    wd = input('Type a word: ')  
    if len(wd) > 4:  
        return len(wd)  
    else:  
        print('The word is too short!')
```

```
res = short_word()  
print('The result: ', res)
```

Type a word: python  
6

# Arguments

- Where? in parentheses after the **name**
- What? a value we pass into the **function** as its **input**
- Why? do different work when we call it at **different** times


```
l = len('Hello world')
```



Argument

# Functions That Take Parameters

parameter : variable used in the function definition



```
def print_len(s):  
    length = len(s)  
    print('The word has', length, 'characters')  
    return len(s)
```

```
res = print_len("python")
```



Argument

The word has 6 characters

# Parameters

A **parameter**: a variable which we use **in** the function **definition**.

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

# Arguments, Parameters, and Results

```
>>> length = len('python')  
>>> print(length)  
6
```

Argument  
→  
'python'



```
def len(par):  
    blah  
    blah  
    do sth to par  
    return 6
```

Parameter  
↙



6  
↑  
Result



# Multiple Parameters / Arguments

- We can define more than one **parameter** in the **function definition**
- We simply add more **arguments** when we call the **function**
- We match the number and order of arguments and parameters

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

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

8

# To function or not to function...

- **Organize** your code into “paragraphs”
- **Don't repeat** yourself
- If something gets too long or complex, **break** it up into logical chunks and put those chunks in functions