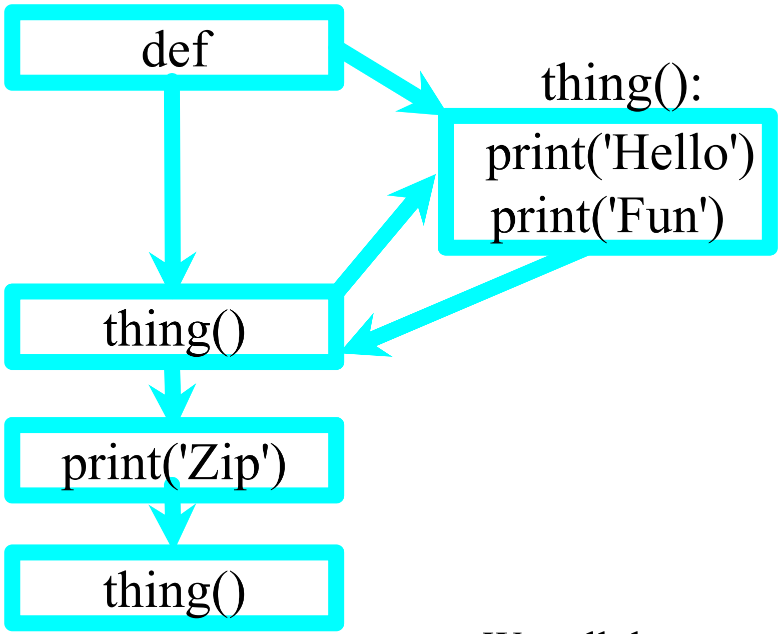
ITA6017
Python Programming

Dr. Arun Pandian J

Stored (and reused) Steps



Program:

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

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

Output:

```
Hello
Fun
Zip
Hello
Fun
```

We call these reusable pieces of code “functions”

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

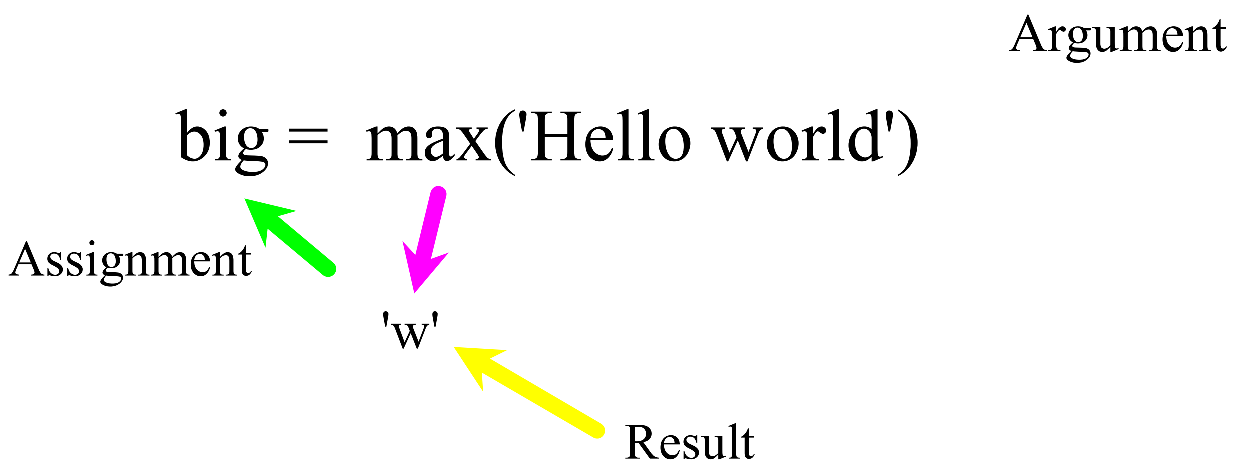
We treat function names as “new” reserved words
(i.e., we avoid them as variable names)

Function Definition

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

We define a function using the def reserved word

We call/invoke the function by using the function name, parentheses, and arguments in an expression



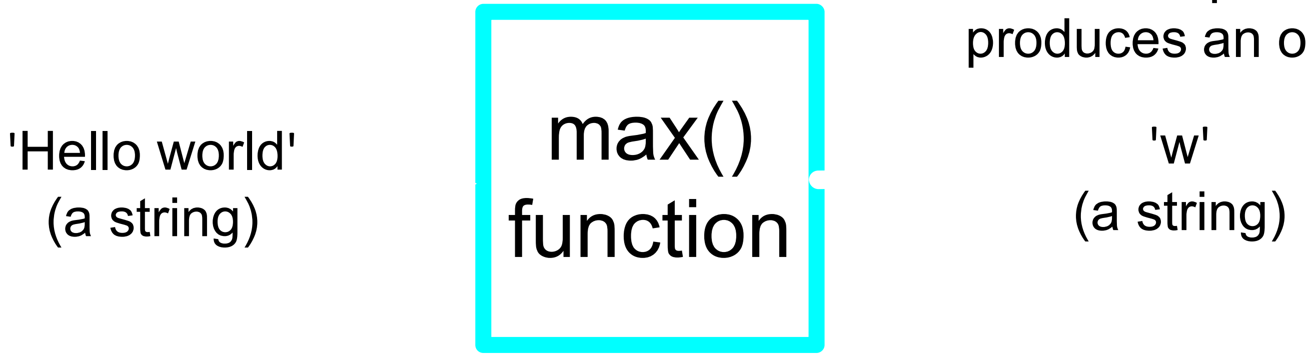
```
>>> big = max('Hello world')
>>> print(big)
w
>>> tiny = min('Hello world')
>>> print(tiny)

>>>
```

Max Function

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

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



Guido wrote this code

Max Function

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

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

'Hello world'
(a string)

```
def max(inp):
   blah
   blah
   for x in inp:
     blah
     blah
```

'w'
(a string)

Guido wrote this code

Type Conversions

When you put 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)
<class 'int'>
>>> f = float(i)
>>> print(f)
42.0
>>> type(f)
<class 'float'>
>>> print(1 + 2 * float(3) / 4 – 5)
-2.5
>>>
```

String Conversions

You can also use int() and float() to convert between strings and integers

You will get an error if the string does not contain numeric characters

```
>>> 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()
```

Functions of Our Own…

Building our Own Functions

We create a new function using the def keyword followed by optional parameters in parentheses

We 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 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
```

Definitions and Uses

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

This is the store and reuse pattern

```
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
```