Functions

Functions

- A set of statements (lines of code) that can be run repeatedly

- Goals: Learning Python by Lutz and Ascher
  - Code reuse
  - Procedural decomposition

Top-Down Design

- Break problem into subproblems
- Print HIHO in block letters

3. print H
4. print I
5. print H
6. print O

- Write a *function* to solve each subproblem

```
def printH():
        print "* *"
        print "***"
        print "* *"
        print

def printI():
        print "***"
        print " * "
        print "***"
        print

def printO():
        print " * "
        print "* *"
        print " * "
        print

printH()
printI()
printH()
printO()
```

Function Calls

- We've seen a few:
  - my_num = input("Enter number: ")
  - my_string = raw_input("Enter string: ")

- Syntax: *function_name(parameters)*
- Other examples:
  - *int("7")* - converts the string "7" to an integer
  - *str(9)* - converts the integer 9 to a string
  - *float(2)* - converts the integer 2 to a float(2.0)
    - can be used to force floating point division: float(5)/2 = 2.5!

Modules

- A module groups together several functions
- math is a common module
- *import math* allows you to use the math functions
- dot operator allows you to call math functions
  - syntax: *module_name.function(parameters)*

```
import math
math.floor(9.5)
math.ceil(9.5)
str(math.floor(9.4)) #function call as parameter
```

Function Definition

- **Step 1**: Think about what your function will do, the input it requires, the result it will produce, and the side-effects it might have
  - printH
    - the function will display the letter H in star characters
    - it does not require input
    - it will not produce a result
    - the side-effect will be output displayed on the screen

Function Definition

- Syntax:

```
def function_name(parameters):
    statements
```

- function_name can be anything - follow the rules for variable names
- parameters are the input
- statements are the code that gets executed
- statements ***MUST*** be indented (all by the same number of spaces/tabs)

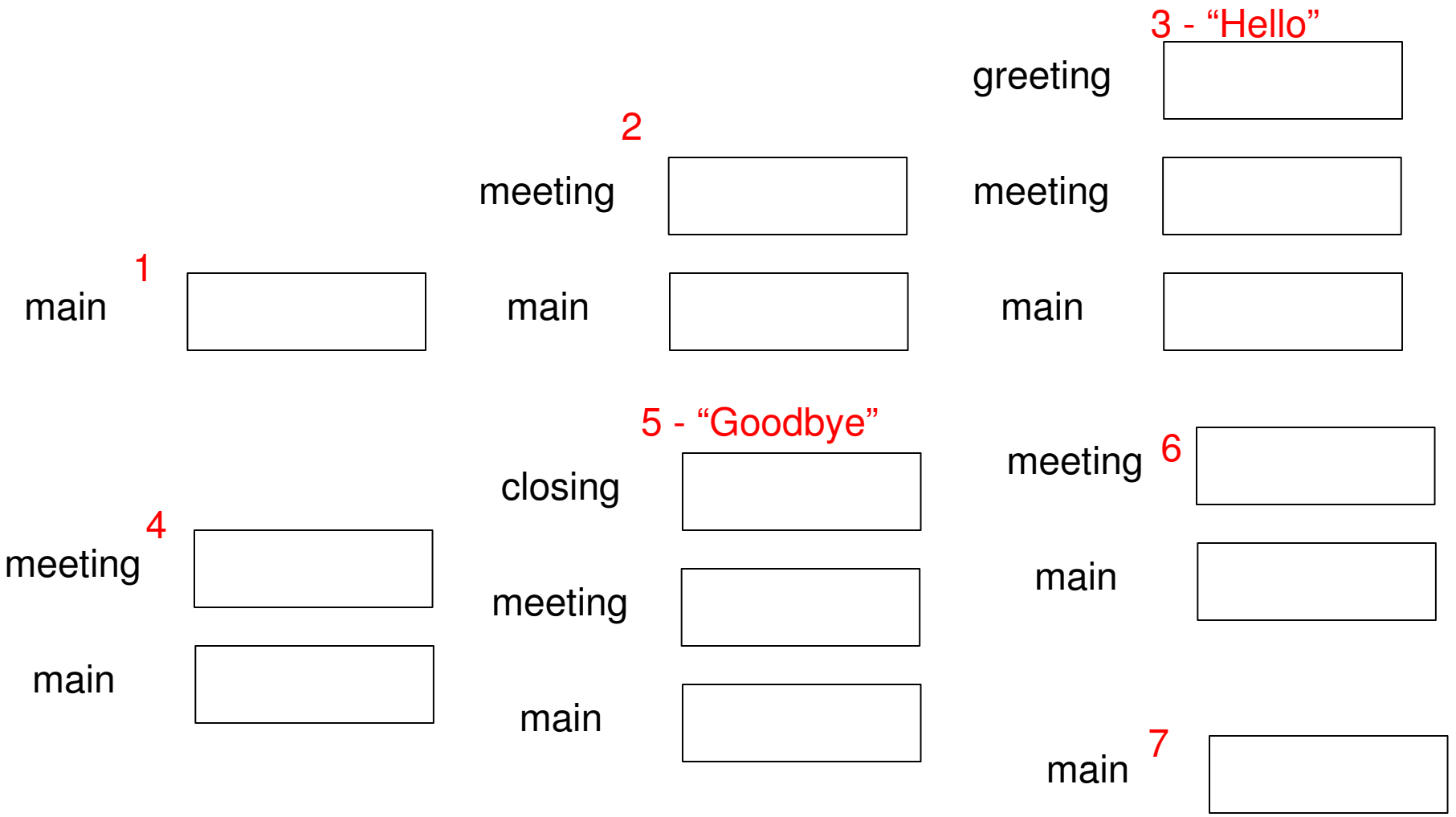
Example - no input

```
#definition of function to print a greeting
#no input, no output, side-effect: greeting is displayed
def greeting():
        print "Hello”
greeting() #call to function greeting

#definition of function to print a closing
#no input, no output, side-effect: closing is displayed
def closing():
        print "Goodbye"
closing() #call to function closing

#definition of function to print a greeting and closing
#no input, no output, side-effect: greeting and closing displayed
def meeting():
        greeting() #example of a function call from within
        closing()  #a function
meeting() #call to function meeting
```

Call to function meeting()



Parameters/Arguments

- Input for functions
- Specify variable names in parameter list

```
def add(number1, number2):
    sum = number1 + number2
    print “Sum: “, sum
```

- When function *add* is called, two numbers must be passed as input

```
add(3, 4)
```

- Variable number1 gets the value 3 and variable number2 gets the value 4

Parameters/Arguments

- Values are assigned in order
  - the first value passed in the function call is assigned to the first parameter in the function definition

```
>>> def taketwo(mynum, mystring):
...     print "mynum ", mynum
...     print "mystring ", mystring
...
>>> taketwo("hello", 7)
mynum  hello
mystring  7
```

Parameters/Arguments

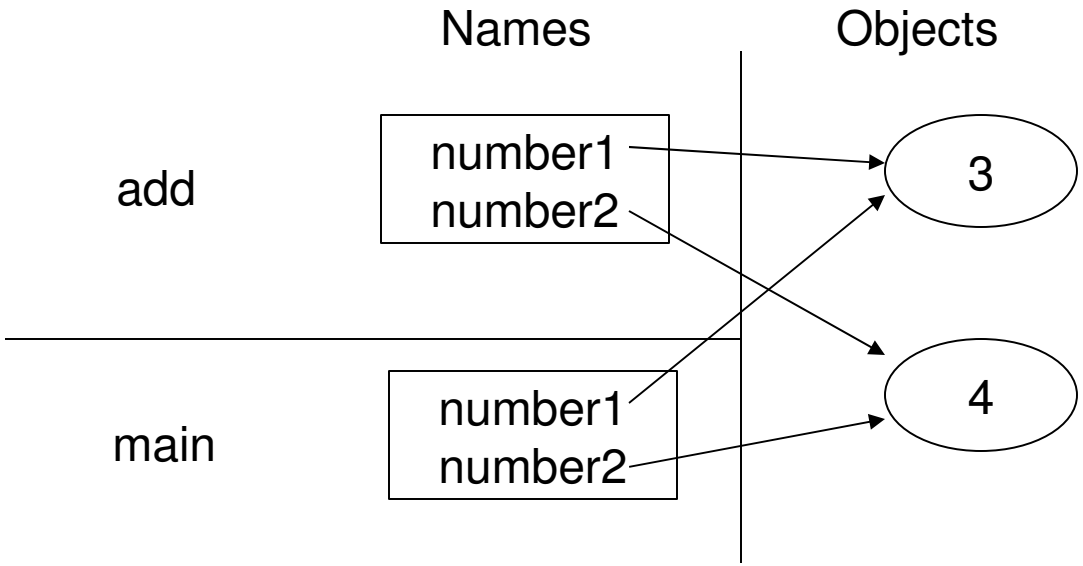
- Variables can be passed as parameters

```
number1 = input("Enter first number: ")
number2 = input("Enter second number: ")
add(number1, number2)
bob = input("Enter first number: ")
alice = input("Enter second number: ")
add(bob, alice)
```

Parameters/Arguments

- Pass by assignment

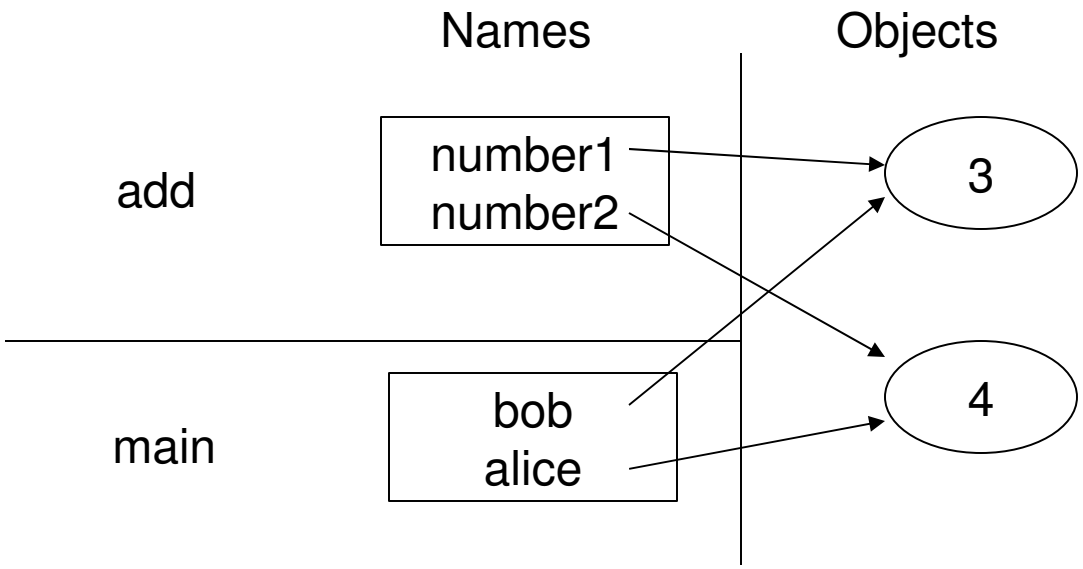
```
number1 = input("Enter first number: ")
number2 = input("Enter second number: ")
add(number1, number2)
```



Parameters/Arguments

- Pass by assignment

```
bob = input("Enter first number: ")
alice = input("Enter second number: ")
add(bob, alice)
```



Scope

- Parameters and variables defined inside a function can only be accessed in that function

```
def greeting(word):
        sentence = "The greeting is " + word + "."

print sentence

Traceback (most recent call last):
  File "test.py", line 4, in ?
    print sentence
NameError: name 'sentence' is not defined
```

Scope

- Parameters and variables defined inside a function can only be accessed in that function

```
def greeting(word):
        sentence = "The greeting is " + word + "."

print word

Traceback (most recent call last):
  File "test.py", line 4, in ?
    print sentence
NameError: name 'word' is not defined
```

Another Example

```
def greeting(word):
        sentence = "The greeting is " + word + "."
        print sentence

sentence = "This is not the greeting."
print sentence
greeting("hello")
print sentence
```

```
This is not the greeting.
The greeting is hello.
This is not the greeting.
```

Return Values

- Functions may return a value to the caller
- Results should be saved in a variable
  - *the function call should appear on the right side of an =*

```
#a function to get input
def getprice():
        price = input("Enter purchase price: ")
        return price
price = getprice()
```

```
TAX_RATE = .0825

def getcost():
        cost = input("Enter item cost: ")
        return cost

def calctax(cost):
        tax = cost*TAX_RATE
        return tax

def calctotal(cost, tax):
        total = cost+tax
        return total

def printresult(cost, tax, total):
        print "Cost: ", cost
        print "Tax : ", tax
        print "Total: ", total

cost = getcost()
tax = calctax(cost)
total = calctotal(cost, tax)
printresult(cost, tax, total)
```