Introduction to Computer Programming
CSCI-UA 2

Functions
Encapsulating Logic
Functions

Groups of statements that can be run more than once in a program

Reusable chunks of code

Take input, provide output

Can be reused in a variety of contexts

Maximize code reuse, minimize code redundancy

Encapsulate logic, splitting complex systems into manageable parts
Calling Functions

`type(3.7)`

Function name
Arguments passed into function
Return Value

( ) tells Python to execute the function

Even if a function takes no input, the brackets are still required

Some functions do not return a value
Defining Functions

Just like writing a Python program but with some extra syntax

Function header:

• Begins with `def`
• Followed by name of function
• Function parameter list
• Ends with a `:`
Documentation String

“Doc String”

Optionally follows function header
Explains what function will do
May include example(s)
Almost always good to include for clarity and as a reminder
Function Body

All the code that follows the header
Simply an indented block of code with necessary statements
This code can use the variables from the function header
Function should return a value with the keyword return
Function Syntax

def name(parameters):
    statements

A return value is not required

return None

Anything besides return are called “side effects”

A print() statement is an example of a side effect
Functions
Local and Global Variables
Variables
Local and Global

Variable scope is something we must pay attention to when using functions.

When and where is a variable accessible to your program?

Local variables are only usable within the function they are local to.

Global variables are usable by any function or code within your program.
Local Variables

def greet():
    greeting = 'Hello!'
    return greeting

In this example, greeting is a local variable, accessible only to the
`greet()` function in the program.

When this function ends, greeting is automatically deleted because it is a
local variable.
Global Variables

```python
name = 'Guido'
def greet():
    greeting = 'Hello!'
    return greeting

In this example, name is a global variable, accessible to all functions in the program.
```
Main Function

main()

It is both common and a good idea to use a main function in your programs. This is usually the starting point of a program and is run by typing: `main()`.

This simplifies rerunning programs and as well as passing input values.
Function Parameters

Used to pass input into a function

Python passes values by reference

Default values can be set for parameters

def name(parameter = 'default')
Modules

A module is a group of related functions

Different from a regular Python program in that it acts like a toolbox

A module usually does not have a `main()` function

To use a module, simply import it
Importing Modules and Calling Functions

Calling function after importing module:

```python
import module_name
module_name.function()
```

Calling function after importing module along with all of its functions:

```python
from module_name import *
function()
```
Module Namespaces

Functions within a module are available when you import them

Modules form namespaces

Different modules with the same function name will not clash in the same program

module1.function(parameters)

module2.function(parameters)

The only time functions may conflict is when you import all using *
Introduction to Computer Programming
CSCI-UA 2

Functions
Encapsulating Logic