Introduction to Computer Programming
CSCI-UA 2

Functions
Encapsulating Logic

⌘
⌘
c
⌘
⌘
p
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

\texttt{type(3.7)}

<table>
<thead>
<tr>
<th>Function name</th>
<th>Arguments passed into function</th>
<th>Return Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

( ) 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

Functions
Encapsulating Logic

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`
- After `return`, Python jumps out of the function and back to the program
Function Syntax

```
def name(parameters):
    statements

    return None

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
```
<table>
<thead>
<tr>
<th>Introduction to Computer Programming</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSCI-UA 2</td>
<td>Encapsulating Logic</td>
</tr>
</tbody>
</table>