CSSE 120 – Introduction to Software Development

Concept: *Calling Functions*

Defining functions

A *function* is *a chunk of code that has a name*. Here (to the right) is a portion of an example of the notation for *defining* a function. (It shows how the builtins function called *input* might have been defined in that module.)

```python
def input(prompt):
    print(prompt, end='')
    ...
    ...
    return <what the user typed, as a string>
```

The *name* of the function follows the keyword `def`. The variables in the parentheses after the name of the function are called *parameters*. We'll talk lots more about *parameters* in a subsequent session.

Why have functions?
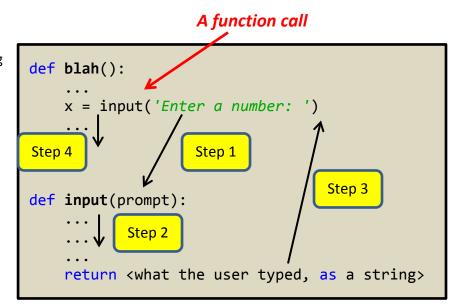
Functions are powerful for 2 reasons:

- They help *organize a program into logical chunks*. That makes it easier to:
    - Test the program (by testing the chunks, called *unit testing*).
    - Modify the program (by focusing your interest on the chunks of interest).
    - Write correct code (by understanding the organization of the program).

- You can *re-use functions*. That is, you can call them over and over again, with different values for the parameters to achieve different results. We'll talk lost more about that soon.

Calling functions

You *call* (aka *invoke*) a function by writing its *name followed by parentheses*, with the *actual arguments* placed inside the parentheses.

When you call a function:

1. The actual *arguments* of the function call (the values in the parentheses) are sent to the formal *parameters* of the function definition.

2. *Execution continues* at the beginning of the definition of the called function.

A function call

```python
def blah():
    ...
    x = input('Enter a number: ')
    ...
```

Step 4   Step 1

Step 3

```python
def input(prompt):
    ...
    ...      Step 2
    ...
    return <what the user typed, as a string>
```

3. When the function's *return* statement is executed, the returned value is sent back to the calling function. Or, if the end of the function is reached without a return statement, the special value *None* is sent back to the calling function.

4. *Execution continues* from the place where the function call appeared, with the returned value replacing the function call.