CIS 231

Python – Ch. 6

Fruitful functions

6.1 Return values

- Functions can return values back to whatever function called them
- Functions that don’t explicitly return a value have a designated return value of `None`
- The `return` statement designates the value to return, then leaves the function immediately.

(more)

Return values cont.

- You can have multiple return statements triggered by conditionals
  - Each leads to leaving the function
- While it’s a good idea to have a return for all possible outcomes, it’s not required in Python
  - If doesn’t hit a `return` statement, returns `None`

6.2 Incremental development

- It’s highly advisable, especially when writing a fairly involved program, to code and test it one piece at a time
  - Easier to detect which problem(s) was just introduced
- After each new phase, make sure that you test the incumbent code as well to make sure nothing was broken
  - A/k/a *Regression Testing*

6.3 Composition

- Using other functions to construct a function
  - Done by calling the other function from within the function you’re building
  - Builds on work already done
  - Easier to maintain

6.4 Boolean functions

- Functions can return boolean (true/false) values
- Good for evaluating conditions to be used by….conditional statements

6.5 More recursion

- Recursive functions will tend to return values
- Will typically use values from a recursive call to compute and return their own value
- Review Figure 6.1 Stack diagram, p. 57

6.6 Leap of faith

- With built-in and recursive functions, the assumption is that they work correctly
- Verify via aggressive testing!

6.7 One more example

- Recursive Fibonacci sequence
  - Can be difficult to follow the flow when each recursive call calls the function more than once to compute its value
- Be aware of possible head explosions
- In practice, doing Fibonacci recursively is a rather bad idea

6.8 Checking types

- You can find out what type of value a variable is during the program (a/k/a its *runtime type*) via the `isinstance()` function
- Returns a Boolean
- Pass it the variable you’re asking about, and the type you want to verify that it might be

```
# is n an integer?
if isinstance(n, int):
```

Up Next

- Ch. 7 – Iteration
  - Multiple assignment
  - Updating variables
  - The `while` statement
  - `break`
  - Square roots
  - Algorithms