Outline for Today

• Functions, continued

• Conditionals (if, if-else)

• Group A will have lab here Friday; B will be in GHC 5207, 5208, 5210 this Friday only
Functions

• A function is a way to group statements together to do one (small/specific) thing.

• Functions will be useful to organize our implementations of algorithms (think of them as similar to paragraphs in an essay)
  – paragraph is to essay as function is to algorithm
  – functions are used to structure your program in a modular fashion

• Top-down Design
  – Top-down design is a way to solve a problem wherein you start with a high-level solution to the problem (an algorithm), break that solution up into smaller steps, and then translate the solution into a program
  – Often, each “small step” will be its own function
  – Each function should be tested to make sure it works!
Built-in functions (from Monday)

• Python has a number of built-in functions:
  – https://docs.python.org/3/library/functions.html
  – Some useful ones are abs(), input(), int()

• Math library
  – A predefined module of mathematical values and functions

• Examples

  import math
  r = 5 + math.sqrt(2)
  radians = degrees * (math.pi/180)
  print(math.factorial(10))
Functions, continued…

• Functions are *called* and can take 0 or more *arguments* that are bound to *parameters* in the function definition.

• Parameters make functions more general:
  – E.g., `helloWorld()` vs. `hello(whatever)`

• Functions return a value, whether you make that explicit or not…

• Printing vs. returning a result from a function:
  – Print prints the result to the console
  – Return returns the result to the calling scope, allowing it to be used in whatever way the caller needs (including printing 😊)
Local variables and Scope rules

- Any variable defined inside the function (either in the parameter list or in a statement) is local to the function
- Access to that variable/value exists only during the duration of that function’s execution
- Variables outside that scope (even with the same name) are unchanged
- Some examples…
Conditionals

• Conditional execution based on a Boolean expression (one that evaluates to True or False)
• Boolean expressions use relational and logical operators
  – Relational operators: <, <=, >, >=, ==, !=
  – Logical operators: not, and, or
• Precedence (highest to lowest):
  – Exponentiation
  – Multiplication, division, remainder
  – Addition, subtraction
  – Relational
  – Logicals (not, then and, then or)
Conditionals…

• if statement
  
  if <condition>:

  statement # executed if <condition> is True

• if-else statement

  if <condition>:

  statement # executed if <condition> is True

  else:

  statement # executed if <condition> is False
Conditionals…

- if-elif…else statement
  
  if <condition1>:
    statement # executed if <condition1> is True
  
  elif <condition2>:
    statement # executed if <condition2> is True
  
  elif <condition3>:
    statement # executed if <condition3> is True
  
  else:
    executed if all of <condition1..n> are False
Conditionals...

• if-elif...else example

    if score >= 90:
        print "Your grade is A!"
    elif score >= 80:
        print "Your grade is B!"
    elif score >= 70:
        print "Your grade is C!"
    elif score >= 60:
        print "Your grade is D!"
    else:  print "You have failed!"
What is "true"?

- Note, not capital-T True, which is a constant
- Easier to consider what is false:
  - False (I hope so!)
  - None (where have we seen that?)
  - Zero for any numeric type
  - An empty string (""") or an empty collection (later)
- All other values are true (that's a lot of truth) in the context of an if expression (unless compared to True)