Outline for January 25

Reading: *text*, §3.2, 4.14, 12 **Assignments:** Homework 2, due on February 1 at 11:55pm

1. Conditions
   a. Resolves to boolean value
   b. Literal booleans: `True` (1), `False` (0)
   c. Relational operators
      i. Use two arithmetic expressions connected with relational operatorsto create a boolean
      ii. Relational operators: `>`, `>=`, `<`, `<=`, `==`, `!=`
      iii. Precedence: resolved after arithmetic operators
      iv. Connectives: `and`, `or`, `not`
      v. `6 > 2 + 3`; `"UCD" == "Sac State"`
2. Indefinite loops: execute until a general condition is false (`while`)
   a. `while` [*while.py*]
   b. Contrast with `for`
   c. `break` causes program to fall out of loop (works with `for` too) [*loop1.py*]
   d. `continue` causes program to start loop over immediately (works with `for` too) [*loop1.py*]
3. Definite loops: execute a specific (definite) number of times (`for`)
   a. General form: `for i in` *`iterator`*
   b. *Iterator* is either list or something that generates a list
   c. Very common form: `for i in range(1, 10)`
4. `range()` in detail [*for.py*]
   a. `range(10)` gives 0 1 2 3 4 5 6 7 8 9
   b. `range(3, 10)` gives 3 4 5 6 7 8 9
   c. `range(2, 10, 3)` gives 2 5 8
   d. `range(10, 2, -3)` gives 10 7 4
5. Handling exceptions
   a. `except` [*except0.py*]
   b. `except error` [*except1.py*]
   c. `else` [*except2.py*]
   d. `except error as msgvar` [*except3.py*]
   e. `finally` [*except4.py*]
   f. Exceptions in a function: who handles them? [*except5.py, except6.py*]