LARSON—MATH 556—SAGE WORKSHEET 02
Cocalc/Sage Basics.

1. (a) Start Firefox or Chrome browser.
   (b) Go to http://cocalc.com
   (c) Click “Sign In”.
   (d) Click project Math 556.
   (e) Click “New”, call it s02, then click “Sage Worksheet”.

2. A very useful arithmetic operator in Sage is the modulo operator (represented by %). 
   \(a \% n\) gives the remainder of dividing \(a\) by \(n\). Evaluate 5\%2. Now evaluate 6\%2. Try 99\%5.

   **Boolean Expressions in Sage**

   A *boolean expression* is one that evaluates to True or False.

3. Evaluate 3==4.

4. Evaluate 3==3.

5. Evaluate 3>3.

6. Evaluate 3>=-3.

7. Evaluate 13\%2==1.

8. Evaluate 13\%2==0.

   While “==” is used as a claim of equality of expressions (the left-hand-side and the 
   right-hand-sides of the “==”) the symbol “!=” is used to express does-not-equal.


10. Evaluate 5!=5.
11. We will assign a value to a variable “a”. Then we will use that variable in a boolean expression. (These two lines can be typed in one cell, or each in its own cell). Type and evaluate:

   a=5
   a>2

Boolean expressions can be combined with boolean operators like “and” and “or”.

12. Evaluate 3==3 and 3==4.

13. Evaluate 3==3 or 3==4.

Lists in Sage

A list is a basic data structure in Python and Sage. They are represented by square brackets with comma separated numbers, strings, etc., between them (like [2, 5, 9] or ["red", "blue"]).

14. Lists can be given names. Evaluate L=[2,5,9]. Then evaluate L.

15. Lists are indexed starting with 0. Evaluate each of L[0], L[1], L[2], and L[3].

16. Lists can be combined with “+”. Evaluate [2,5,9]+[3,4,5].

17. Let M=[3,4,5]. Evaluate L+M.

18. If you want all the integers from x to y you can use the shorthand notation [x..y]. Evaluate [3..7].

19. If you want a list with m n’s you can use the shorthand notation [n]*m. Evaluate [0]*7.

20. You can have a list of lists. Evaluate L=[[0,1],[2,3],[4,5]]. Now evaluate L[1]. Then evaluate L[1][0]. What do you think the value of L[0][1] is?

21. You can use map() to apply a function to each term of a list. Evaluate map(abs, [-1,2,-3]).

22. You can also use list comprehension to get the same behavior as map(). Evaluate [abs(x) for x in [-1,2,-3]].