Last name _____

First name _____

LARSON—MATH 255–CLASSROOM WORKSHEET 04
Strings, Booleans, and More.

1. Log in to your Sage Cloud account.

   (a) Start the Chrome browser.

   (b) Go to `http://cloud.sagemath.com` and sign in.

   (c) You should see an existing Project for our class. Click on that.

   (d) Click "New", call it **c04**, then click "Sage Worksheet".

   (e) For each problem number, label it in the Sage cell where the work is. So for Problem 1, the first line of the cell should be `#Problem 1`.

String formatting.

A *string* is a sequence of *characters* (letters, numerals, symbols, etc). If you put a sequence of characters between quotes, you are telling Sage to treat what's between the quotes as a string (instead of as a *keyword*). Strings can be manipulated, and have places that can be filled in.

2. Type and evaluate `print('This string has { }'.format('17 characters'))`. Now try replacing '17 characters' with any other string.

3. Type and evaluate the following program.

```
def superstring(x):
    print('This string has { }'.format(x))
```

4. Now test your function. Type and evaluate `superstring('black letters')`.

More graphing and calculating basics.

5. Make a point at $(4, 4)$ Evaluate `point((4,4))`.

6. Make it bigger by adjusting the "size" parameter; evaluate `point((4,4),size=200)`. Try other values for `size`.

7. Draw a line from $(-1, 1)$ to $(4, 4)$ by evaluating `line([(-1,1),(4,4)])`. Try drawing a line with 3 points.

8. Make the line thicker by adjusting the "thickness" parameter: evaluate `line([(-1,1),(4,4)],thickness = 4)`. Try other values of `thickness`.

9. Make the line dashed by adjusting the "linestyle" parameter: evaluate `line([(-1,1),(4,4)],linestyle="dotted")`. Try another value for "linestyle" by reading the options from the help command `line2d?`.

10. Now make the line red.

11. Draw a triangle between $(1,1)$, $(1,2)$, and $(2,1)$ using the line command.

12. Now draw a triangle between $(1,1)$, $(1,2)$, and $(2,1)$ using the `polygon` command; find examples of how this command works with `polygon?`. What's the difference?

Boolean Expressions in Sage

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

13. Evaluate `3==4`.

14. Evaluate `3==3`.

15. Evaluate `3>3`.

16. Evaluate `3>=-3`.

17. Evaluate `13%2==1`.

18. 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 in-equality.

19. Evaluate `5!=7`.

20. Evaluate `5!=5`.

21. 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".

22. Evaluate `3==3 and 3==4`.

23. 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"]`). We have already seen lists in our use of both the `solve()` and `line()` commands which used, respectively, a list of equations and a list of points.

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

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

26. Lists can be combined with "+". Evaluate `[2,5,9]+[3,4,5]`.

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