Last name ______________________

First name ______________________

LARSON—MATH 255–CLASSROOM WORKSHEET 16
Files & Interacts.

1. Log in to your Sage/Cocalc account.
    (a) Start the Chrome browser.
    (b) Go to `http://cocalc.com` and sign in.
    (c) You should see an existing Project for our class. Click on that.
    (d) Click "New", call it **c16**, 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`.
    (f) When you are finished with the worksheet, click "make pdf", email me the pdf (at `clarson@vcu.edu`, with a header that says **Math 255 c16 worksheet attached**).
2. Now it is the case on any larger program that you will want to use functions you have previously defined. These are called *tools*. Instead of copying and pasting from your old code. You can save them as *files* and load them as needed.
    (a) Click "New". Type `heads_from_n_flips.sage` and then click "file". (You are making a **.sage** file *not* our usual Sage Worksheet file. These are regular text files that are loaded as Python files plus some *preprocessing*).
    (b) Define the function:

```
def heads_from_n_flips(n):
    heads=0
    for i in [1..n]:
        if random()<.5:
            heads=heads+1
    return heads
```

    (c) Click "Save" and then go back to your **c16** worksheet.
    (d) Type `load("heads_from_n_flips.sage")` and evaluate.
    (e) Now try `heads_from_n_flips(100)` a few times. You never need to write this function again. You have a tool!

Reading in, and working with, data files is an important ability. First we will create a data file. Then we will read it in line-by-line, and then we will work with the data.

An important thing to know/note is that a file is actually a big *string*. You can read the lines of a file with `readline()`. Those lines are also strings (and not numbers - despite how they look). If you want numbers they must be converted to numbers.

3. (a) Go to: `http://projecteuler.net/problem=13`. Copy the one hundred 50-digit numbers there.

   (b) Click "New", type in `one_hundred_numbers.txt` as the name of your file, then click "File".

   (c) Paste in your numbers and "Save".

   (d) Now go back to your **c16** worksheet.

   (e) Type in:

   ```
   data=open("one_hundred_numbers.txt")
   numbers=[]
   number_string=data.readline()
   while(number_string!=""):
       number=Integer(number_string)
       numbers.append(number)
       number_string=data.readline()
   ```

   You have a *list* of numbers. You can use built-in Sage functions to find out statistics about this list. How many numbers are there? What is the biggest number? What is the sum of these numbers? What is the average of these numbers? What is their median?

4. Now we will create a file `testio.txt` in *write* mode (hence the "w"), and write something to it. The *close* command forces the writing to happen and flushes the *buffer*. Now you can't write anything else to the file without reopening it.

   ```
   datafile=open("testio.txt","w")
   datafile.write("hello world!")
   datafile.close()
   ```

5. Go to Files, find `testio.txt` and click on it to see what's in there. Now try:

   ```
   datafile=open("testio.txt","w")
   datafile.write("hello again!")
   datafile.close()
   ```

6. Go back and take a look at `testio.txt`. The old data is gone. It was overwritten. To add data you need to open the file for *appending* (with an "a"). Try:

   ```
   datafile=open("testio.txt","a")
   datafile.write("hello again again!")
   datafile.close()
   ```

7. Go back and take a look at `testio.txt`. The new data got mushed together with the old data. Let's start over and give a new line for each input string.

```
datafile=open("testio.txt","w")
datafile.write("2nd try \n")
datafile.write("hello world! \n")
datafile.close()
```

8. Go back and take a look at `testio.txt`. Now let's open up the file to read its contents—without having any danger of modifying the data (hence the "r") and see what's in there.

```
datafile=open("testio.txt","r")
dline=datafile.readline()
```

   Evaluate `dline` to see what that variable holds. Now repeat the last line of the code and reevaluate `dline`.

Prime Numbers

9. `is_prime(n)` is a built-in Sage function which tests if an integer n us prime. We can use it to write a function for counting the number of primes which are no more than n.

```
def count_primes(n):
    count=0
    for x in [2..n]:
        if is_prime(x)==True:
            count = count + 1
    return count
```

   Find `count_primes(100)`, `count_primes(10000)`, and `count_primes(1000000)`.

10. `prime_pi(n)` is a built-in Sage function which does the same thing as `count_primes(n)`. Compute `prime_pi(100)`, `prime_pi(1000)`, and `prime_pi(1000000)`. Check that you get the same results as before.

11. Now lets compare the speeds of our prime counting function and the built-in prime counting function. Try `timeit("prime_pi(100000)")` and `timeit("count_primes(100000)")`. Which is faster?

12. Try the following Sage Interact which visualizes the Prime Number Theorem (PNT).

```
@interact
def pnt(N=input_box(200)):
    show(plot(prime_pi,0,N,color="red")+plot(x/(log(x)-1),5,N,color="blue"))
```

   Try putting different numbers in the input box.

13. Let's do the same thing in a new way. Try:

```
@interact
def pnt2(N=(100,(2..1000000))):
    show(plot(prime_pi,0,N,color="red")+plot(x/(log(x)-1),5,N,color="blue"))
```