LARSON—MATH 255–CLASSROOM WORKSHEET 14
Recursion & Random Values.

1. (a) Start the Chrome browser.
   (b) Go to `http://cocalc.com`
   (c) Login using **your VCU email address** .
   (d) Click on our class Project.
   (e) Click "New", then "Worksheets", then call it **c14**.
   (f) For each problem number, label it in the Sage cell where the work is. So for Problem 2, the first line of the cell should be `#Problem 2`.

Recursion

A **recursive** function is a function that calls itself. It must always have a *base case* so that the recursion eventually stops. Our examples last class were the factorial and gcd functions.

2. The Fibonacci sequence F_n is defined as follows $F_0 = 0$, $F_1 = 1$ and $F_n = F_{n-1} + F_{n-2}$ for $n > 1$. Here is a recursive function `fibonacci(n)` that computes the nth Fibonacci number.

```
def fibonacci(n):
    if n==0:
        return 0
    if n==1:
        return 1
    else:
        return fibonacci(n-1)+fibonacci(n-2)
```

3. Try this for small values of n to make sure that it works, then try it for $n = 10, 20, 30, 40, 50$. Does it finish? If not, why not?!?!

4. Now define a function `fibonacci2(n)` that computes the n^{th} Fibonacci number *iteratively* (that is, without using recursion.

   Try this for small values of n to make sure that it works, then try it for $n = 50$. Does it finish?

Random Values

5. `random()` returns a random number in $[0, 1]$. Execute it a few times to see what you get.

6. Define a function `my_mood()` which prints "I'm happy" or "I'm sad" randomly.

```
def my_mood():
    if random()<.5:
        print("I'm happy")
    else:
        print("I'm sad")
```

7. Use `random()` to define a function `coin_flip()` which randomly returns the string "H" (for heads) half the time and returns the string "T" (for tails) half the time. Try it a few times; your results will vary.

8. Make an *interactive* coin flipping program:

```
@interact
def i_coins(n=input_box(5)):
    for x in [1..n]:
        print(coin_flip())
```

   Try different numbers in the box.

9. Run your coin flipping program 100 times and collect data. A random coin flipping program should come up heads about half the time. How many times do you get heads?

10. Now run your coin flipping program 1000 times and collect data. A random coin flipping program should come up heads about half the time. How many times do you get heads?

    It is often useful to generate random integers. It only makes sense to generate random integers from within some range of integers. We do this with `randint()`.

11. Evaluate `randint(5,100)` a few times; your results will vary. This will generate random integers in the range $[5, 100]$, including both endpoints.

12. Now try the following function. Evaluate it a few times; your results will vary!

```
def sybil():
    print("My favorite number is {}!".format(randint(1,50)))
```

13. **Investigate**. Does `randint()` produce a *uniform distribution*? (That is, as you repeat experiments of `randint(a,b)` are the number of produced outcomes of each possible integer roughly the same? Do some experiments!)

Getting your classwork recorded

When you are done, before you leave class...

(a) Click the "Make pdf" (Adobe symbol) icon and make a pdf of this worksheet. (If Cocalc hangs, click the printer icon, then "Open", then print or make a pdf using your browser).

(b) Send me an email with an informative header like "Math 255—c14 worksheet attached" (so that it will be properly recorded).

(c) Remember to attach today's classroom worksheet!