Diagnostic 3a Solutions

Modeling and Simulation
Fall 2017

This diagnostic covers material from Section 3.1 to 3.6 of *Modeling and Simulation in Python*. It is open book, open notes, open computer. You can take as long as you need. If you have not done the reading yet, do it now. You should make a reasonable effort to complete the diagnostic on your own, but you can ask classmates, NINJAs, and instructors for help.

When you are confident that your answers are correct, please meet with a NINJA or instructor to get checked off. Most of the time we expect this to happen during class, but occasionally you might have to schedule a meeting.

Vocabulary

Fill in the blanks in the following sentences with the words that make the most sense.

Before you can use the functions defined in a library, you have to _____import_____ the library.

The result from `read_html` is a sequence of _____DataFrame_____ objects.

The _____TimeSeries_____ object, which is defined in the modsim library, represents a sequence of times or dates and a sequence of associated values.

Putting data values in the text of a program is called _____hard coding_____, and it is usually considered a bad idea.

Bonus question: What is the Green Revolution?

Code

Getting back to the penny example from the previous diagnostics, suppose you have two `System` objects with variables `heads` and `tails`.

Write a function called `add_pennies` that takes two `System` objects as parameters and makes a new `System` object, where the new object contains the total number of heads and tails from the two inputs.

```
def add_pennies(s1, s2):
    heads = s1.heads + s2.heads
    tails = s1.tails + s2.tails
    return System(heads=heads, tails=tails)
```

(The following also works, but I don't expect anyone to do it.)

```
def add_pennies(s1, s2):
    return System(s1 + s2)
```

From the previous diagnostic, assume we have a function called `run_simulation` that takes `p` as a parameter, creates a `System` object with variables `heads` and `tails`, calls `flip_until_tails`, and returns the `System` object as a return value.

Write a few lines of code to call `run_simulation` twice, assign the resulting `System` objects to two variables, then use `add_pennies` to add the `System` objects, and print the total number of heads.

```
s1 = run_simulation(0.5)
s2 = run_simulation(0.5)
s3 = add_pennies(s1, s2)
print(s3.heads)
```

Bonus question: Write a function called `run_simulations` that takes parameters `n` and `p`. It should call `run_simulation` `n` times with probability `p` and return a `System` object that contains the total number of heads and tails after `n` simulations.

```
def run_simulations(n, p):
    total = System(heads=0, tails=0)
    for i in range(n):
        system = run_simulation(p)
        total = add_pennies(total, system)
    return total
```