Problem Wk.4.1.5: PolyR on Signals

Read the handout for the software lab. Implement the `polyR` procedure. You can assume the following methods are in the `Signal` class:

```
class Signal:
    def __add__(self, other):
        return SummedSignal(self, other)
    def __rmul__(self,other):
        return ScaledSignal(self, other)
```

And so, you can add `Signal` instances and scale them by a constant using arithmetic operations:

```
s1 = CosineSignal(0.1)
s2 = CosineSignal(0.3)
s3 = s1 + s2
s4 = 2 * s3
```

Note that the number must be the first argument of the multiply.

You can also use `Rn(signal, k)`; it is already defined for you.

Be careful when using the + operator on `Signals`; it will only work on two `Signals`. For example, you cannot do:

```
result = 0
result += CosineSignal(0.1)
```

If you want to use the Python `sum` operator, then you have to be careful about specifying the start value; see the Python documnetation for `sum`.

The first argument to `polyR` is a signal and the second is an instance of the `Polynomial` class. You can get a list of the coefficients from the `coeffs` attribute of a `Polynomial` instance. You can create new instances via `poly.Polynomial(c)` where c is a list of coefficients.

You are welcome to use recursion in your procedure.

6.01SC Introduction to Electrical Engineering and Computer Science

Spring 2011