Lecture #2: Various

Dice Throws

- When throwing n six-sided dice, what is the probability of getting a score of at least k? Let's ignore all rules except Pig Out, at least for now.
- Relevant for the end game—when score is near 100.
- **Problem 1.** Fill in the following:

```
def throws(n):
    """A sequence of length 6N+1 in which element k is the sequence
    of all sequences of N dice that score k points."""
```

Score Probabilities

- We'll define the random variable S_n to be the score from throwing n dice.
- We can divide this into two pieces:

$$P(S_n \geq k) = P(S_n \geq k \mid \text{no 1s}) + P(S_n \geq k \mid \text{at least one 1})$$

- For convenience, define:

$$\begin{aligned} U_k &= P(S_n \geq k \mid \text{at least one 1}) \\ B_k &= P(S_n \geq k \mid \text{no 1s}) \end{aligned}$$

- So what are they?

Gaming

- Consider a game in which two players alternate add 1, 2, or 3 to a total until the total is $\geq N$ for some N.
- The last player (the one who causes the total to get to N or above) loses.
- How would you fill this in?

```
def forced_win(score, N):
    """True if the current player, starting from a score of SCORE, can
    force a win, if N is the limiting score."""
```

Church Numerals

- Alonzo Church was a famous logician and mathematician, responsible for, among other things, the lambda calculus (whence Python's lambda expressions) and the Church-Turing Thesis.
- The Church-Turing thesis is that any function on the natural numbers that is computable by some algorithm is computable using a Turing Machine.
- The lambda calculus is a pure calculus of functions, without numbers, strings, booleans, conditionals, etc.
- Yet it can represent the non-negative integers, using an encoding known as *Church numerals*.

The Representation

- We *define* zero and the successor (+1) operator as follows:

```
zero =      lambda x: x
successor = lambda n: lambda f: lambda x: f(n(f)(x))
```

- So 1 is successor(zero) and 2 is successor(successor(zero)).
- What is 1 (as a function that does not use successor)?
- What is 2 (as a function that does not use successor)?

Operations

- How does one turn a Church numeral into the integer it represents?
- How does one implement addition?

```
def add_church(a, b):

    return ________________________________________________
```

- Multiplication?

```
def mul_church(a, b):

    return ________________________________________________
```