6.009: Fundamentals of Programming

Lecture 4: Recursive Patterns

With special guests:

- Doctests
- Generators

Adam Hartz

`hz@mit.edu`

2 March 2020

Recursion

In a general sense, *recursion* occurs when a thing is defined in terms of itself.

Example: For nonnegative integer n,

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n \times (n-1)! & \text{otherwise} \end{cases}$$

To solve a problem recursively, we typically identify:

- One or more **base cases** (a terminating scenario that does not use recursion to produce an answer), and
- One or more **recursive cases** (a set of rules that reduce all other cases toward the base case).

Example: Factorial

```
def factorial(n):
    if n == 0:
        return 1
    return n * factorial(n - 1)
```

Recursion vs Iteration?

Factorials can also be computed iteratively.

```
def factorial(n):
    if n == 0:
        return 1
    return n * factorial(n - 1)

def factorial(n):
    out = 1
    for i in range(1, n+1):
        out *= i
    return out
```

Which would you choose? Why?

Recursion vs Iteration?

Factorials can also be computed iteratively.

```
def factorial(n):
    if n == 0:
        return 1
    return n * factorial(n - 1)

def factorial(n):
    out = 1
    for i in range(1, n+1):
        out *= i
    return out
```

Which would you choose? Why?

Do we even need recursion?

Some Processes are Naturally Recursive

Examples:

- JSON Encoding
- Finding Files
- Flood Fill
- Other Examples

The rest of today: more live programming examples.