6.009: Fundamentals of Programming

Week 3 Lecture: Recursive Patterns

With special guests: testing and debugging

adam hartz
hz@mit.edu

2sday, 22/2/22

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).

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?

Recursion in the Wild

```
def can_log(x):
    """
    Checks whether a given value can be a log entry.

    Valid log entries are strings/bytestrings, ints, floats, complex numbers,
    None, or Booleans; _or_ lists, tuples, sets, frozensets, dicts, or
    OrderedDicts containing only valid log entries.
    """
```

Recursion in the Wild

```
def can_log(x):
    """
    Checks whether a given value can be a log entry.

    Valid log entries are strings/bytestrings, ints, floats, complex numbers,
    None, or Booleans; _or_ lists, tuples, sets, frozensets, dicts, or
    OrderedDicts containing only valid log entries.
    """
    if isinstance(x, (str, bytes, int, float, complex, NoneType, bool)):
        return True
```

Recursion in the Wild

```
def can_log(x):
    """
    Checks whether a given value can be a log entry.

    Valid log entries are strings/bytestrings, ints, floats, complex numbers,
    None, or Booleans; _or_ lists, tuples, sets, frozensets, dicts, or
    OrderedDicts containing only valid log entries.
    """
    if isinstance(x, (str, bytes, int, float, complex, NoneType, bool)):
        return True
    elif isinstance(x, (list, tuple, set, frozenset)):
        return all(can_log(i) for i in x)
```

Recursion in the Wild

```
def can_log(x):
    """
    Checks whether a given value can be a log entry.

    Valid log entries are strings/bytestrings, ints, floats, complex numbers,
    None, or Booleans; _or_ lists, tuples, sets, frozensets, dicts, or
    OrderedDicts containing only valid log entries.
    """
    if isinstance(x, (str, bytes, int, float, complex, NoneType, bool)):
        return True
    elif isinstance(x, (list, tuple, set, frozenset)):
        return all(can_log(i) for i in x)
    elif isinstance(x, (dict, OrderedDict)):
        return all((can_log(k) and can_log(v)) for k,v in x.items())
```

Recursion in the Wild

```
def can_log(x):
    """
    Checks whether a given value can be a log entry.

    Valid log entries are strings/bytestrings, ints, floats, complex numbers,
    None, or Booleans; _or_ lists, tuples, sets, frozensets, dicts, or
    OrderedDicts containing only valid log entries.
    """
    if isinstance(x, (str, bytes, int, float, complex, NoneType, bool)):
        return True
    elif isinstance(x, (list, tuple, set, frozenset)):
        return all(can_log(i) for i in x)
    elif isinstance(x, (dict, OrderedDict)):
        return all((can_log(k) and can_log(v)) for k,v in x.items())
    return False
```

Recursion in the Wild

- Searching for files within a directory.
- Web browser (redirects, inline images, etc)
- String Parsing
- Deep-copying arbitrary objects
- Gradient Descent
- Feedback and IIR filters
- Solving some kinds of puzzles (more on that next week)

More Examples

The rest of today: more live programming examples.