Recursion

Calling a Function from Within Itself

C-START Python PD Workshop

Recursion: What is it?

Recursive functions are functions which rely on themselves to calculate part of the answer. Recursive functions usually have a base case that causes the recursion to end. Here is an example as a story:

A child couldn't sleep, so her mother told a story about a little frog,
  who couldn't sleep, so the frog's mother told a story about a little bear,
    who couldn't sleep, so the bear's mother told a story about a little weasel
      ...who fell asleep.
    ...and the little bear fell asleep;
  ...and the little frog fell asleep;
...and the child fell asleep.

Recursion: What is it?

Base Case

A child couldn't sleep, so her mother told a story about a little frog,
who couldn't sleep, so the frog's mother told a story about a little bear,
who couldn't sleep, so the bear's mother told a story about a little weasel
...who fell asleep.
...and the little bear fell asleep;
...and the little frog fell asleep;
...and the child fell asleep.

Recursion: What is it?

Recursive Part

A child couldn't sleep, so her mother told a story about a little frog,
who couldn't sleep, so the frog's mother told a story about a little bear,
who couldn't sleep, so the bear's mother told a story about a little weasel
...who fell asleep.
...and the little bear fell asleep;
...and the little frog fell asleep;
...and the child fell asleep.

Recursive Functions in Python

Consider the factorial operation.

$$n! = n \times (n-1) \times (n-2) \times \cdots \times 1$$

Recursive Functions in Python

Consider the factorial operation.

$$n! = n \times (n-1) \times (n-2) \times \cdots \times 1$$

We could define this recursively as:

- **Base case:** $0! = 1$
- **Recursive part:** $n! = n(n-1)!$

Recursive Functions in Python

Consider the factorial operation.

$$n! = n \times (n-1) \times (n-2) \times \cdots \times 1$$

We could define this recursively as:

- **Base case:** $0! = 1$
- **Recursive part:** $n! = n(n-1)!$

To code this as a recursive function in Python, we could do:

```
def fact(n):
    if n == 0:          # base case
        return 1
    return n*fact(n-1)  # recursive part
```

Recursion in Practicality: Euclid's GCD

The GCD of a and b is:

- a if $b = 0$
- $\gcd(b, a \mod b)$ otherwise

More info about why this is so can be found at
`https://en.wikipedia.org/wiki/Euclidean_algorithm`

Recursion in Practicality: Euclid's GCD

The GCD of a and b is:

- a if $b = 0$
- $\gcd(b, a \mod b)$ otherwise

More info about why this is so can be found at
https://en.wikipedia.org/wiki/Euclidean_algorithm

Implementation in Python:

```python
def gcd(a, b):
    if b == 0:                  # base case
        return a
    return gcd(b, a % b)        # recursive part
```

Practice: Fibonacci Numbers

The n-th Fibonacci number, $F(n)$, is:

- n if $n = 0$ or $n = 1$
- $F(n-1) + F(n-2)$ otherwise

Try it yourself: Implement a Python function which calculates the n-th Fibonacci number recursively.