CIS 122

Recursion Strikes Again

Recursion

- Reducing a problem to a **smaller** version of itself
- Recursive step
  - How do I reduce my problem?
  - To wash dishes, first wash one dish, then **wash the rest**
  - x! = x * **(x-1)!**
- Base Case
  - Where do I stop?
  - When the sink is empty, the dishes are washed
  - 0! = 1

Not-So-Basic Arithmetic

- Python can multiply numbers with the * operator
  - But what if we want to implement it ourselves?
  - Let's break out some recursion!

Not-So-Basic Arithmetic

- Python can multiply numbers with the * operator
  - But what if we want to implement it ourselves?
  - Let's break out some recursion!

$$a * b = \underbrace{a + a + a + a + \ldots + a}_{b}$$

Not-So-Basic Arithmetic

- Python can multiply numbers with the * operator
  - But what if we want to implement it ourselves?
  - Let's break out some recursion!

$$a * b = a + \underbrace{a + a + a + \ldots + a}_{b-1}$$

Not-So-Basic Arithmetic

- Python can multiply numbers with the * operator
  - But what if we want to implement it ourselves?
  - Let's break out some recursion!

$$a * b = a + a * (b-1)$$

Not-So-Basic Arithmetic

- Python can multiply numbers with the * operator
  - But what if we want to implement it ourselves?
  - Let's break out some recursion!

$$a * b = a + a * (b-1)$$

product(a, b) = a + product(a, b-1)

Not-So-Basic Arithmetic

- Base Case
  - product(a, 0) = 0
- Recursive Step
  - product(a, b) = a + product(a, b-1)

Not-So-Basic Arithmetic

- Base Case
  - product(a,0) = 0
- Recursive Step
  - product(a,b) = a + product(a,b-1)

```
def product(a,b):
    if b==0:
        return 0
    else:
        return a + product(a, b-1)
```

Not-So-Basic Arithmetic

- Base Case
  - product(a,0) = 0
- Recursive Step
  - product(a,b) = a + product(a,b-1)

```
def product(a,b):
    if b==0:
        return 0
    else:
        return a + product(a, b-1)
```

- Does it work?
  - Test it!

Not-So-Basic Arithmetic

- Base Case
  - product(a,0) = 0
- Recursive Step
  - product(a,b) = a + product(a,b-1)

```
def product(a,b):
    if b==0:
        return 0
    elif b < 0:
        return -1 * product(a, -b)
    else:
        return a + product(a, b-1)
```

Not-So-Basic Arithmetic Quiz

- Write a recursive power function
  - power(a, b) = a * a * a * ... * a (b times)
  - (don't worry about negative b)

- Steps
  - Define power recursively
  - Come up with a base case
  - Put it into code

Not-So-Basic Arithmetic Quiz

- Write a recursive power function
  - power(a, b) = a * a * a * ... * a (b times)
- Base Case
  - power(a, 0) = 1
- Recursive Definition
  - power(a, b) = a * power(a, b-1)

```
def power(a, b):
    if b == 0:
        return 1
    else:
        return a * power(a, b-1)
```

Turning Things Around

- How would we reverse a string?

Turning Things Around

- How would we reverse a string?

"ABCDEFG"

Turning Things Around

- How would we reverse a string?
  - What if we knew how to reverse part of it?

"A"+"BCDEFG"

Turning Things Around

- How would we reverse a string?
  - What if we knew how to reverse part of it?

"A"+"BCDEFG"

"GFEDCB"+"A"

Turning Things Around

- How would we reverse a string?
  - What if we knew how to reverse part of it?

- Recursive Step
  - Set aside one letter
  - Reverse the rest of the string
  - Add the letter to the end

"A"+"BCDEFG"

"GFEDCB"+"A"

Turning Things Around

- How would we reverse a string?
  - What if we knew how to reverse part of it?
- Recursive Step
  - Set aside one letter
  - Reverse the rest of the string
  - Add the letter to the end
- Base Case
  - The empty string reversed is itself

Turning Things Around

```
def reverse(string):
    """Returns the reverse of the input string"""
    if string == "":
        return ""
    else:
        firstChar = string[0]        # Set aside first char
        rest      = string[1:]       # Set aside rest of string
        return reverse(rest) + firstChar
```

Turning Things Around

- Problem needs to get smaller when you recurse
- factorial
  - The number gets smaller
  - Base case at 0
- product
  - Second number gets smaller
  - Base case at b==0
- reverse
  - Size of string gets smaller
  - Base case at empty string