Lecture 33: Recursion

Recursion: When you are inside a function, calling that same function

```python
def fun():
    fun()
```

- Can be useful to formulate solutions to problems that would otherwise require lots of code.

When to use recursion?

- When a solution to a problem can be expressed in terms of solutions to simpler versions of the same problem.
Example: Computing the factorial of a number

\[ n! = n \cdot (n-1) \cdot (n-2) \cdot (n-3) \cdots 1 \]

\( n \) needs to be positive int \( \geq 1 \).

| Define | \( 0! = 1 \) |

| \( 4! = 4 \cdot 3 \cdot 2 \cdot 1 = 24 \) |
| \( 3! = 3 \cdot 2 \cdot 1 = 6 \) |
| \( 2! = 2 \cdot 1 = 2 \) |
| \( 1! = 1 \cdot 1 = 1 \) |

\( 0! \) defined as 1 “base case” problem boils down to a fixed number: 1.
Let's code up the function

Factorial Symbol table

print(factorial(3))

return 6

factorial

Name | Value
--- | ---
n | 3

return 3 * factorial(2)

factorial

Name | Value
--- | ---
n | 2

return 2 * factorial(1)

factorial

Name | Value
--- | ---
n | 1

return 1 * factorial(0)

factorial

Name | Value
--- | ---
n | 0

return 1

After return symbol table goes away

n! = n * (n-1)! for n > 0

factorial (1) = 1

factorial (0) = 1

3! = 3 * 2 = 6

2! = 2 * 1 = 2

1! = 1

0! = 1

then symbol table goes away