COMP 110

More on Recursion… Reverse Engineering

Goal for today

- Start with a recursive python program and find out the standard function representation that it is describing.

Recap

Last class, we started with standard function definition and followed steps to define it as a recursive python program.

Steps:

- Standard function
- Sequence representation
- Recursive definition
- Recursive Python function

Recap

Last class, we started with standard function definition and followed steps to define it as a recursive python program.

Steps:

- Standard function
- Sequence representation
- Recursive definition
- Recursive Python function

$$f(n) = n$$

Recap

Last class, we started with standard function definition and followed steps to define it as a recursive python program.

Steps:

- Standard function
- Sequence representation
- Recursive definition
- Recursive Python function

Input n	0	1	2	3	…	n
Output f(n)	0	1	2	3	…	n

Recap

Last class, we started with standard function definition and followed steps to define it as a recursive python program.

Steps:

- Standard function
- Sequence representation
- Recursive definition
- Recursive Python function

- Base case:
  - for n = 0: f(n) = 0

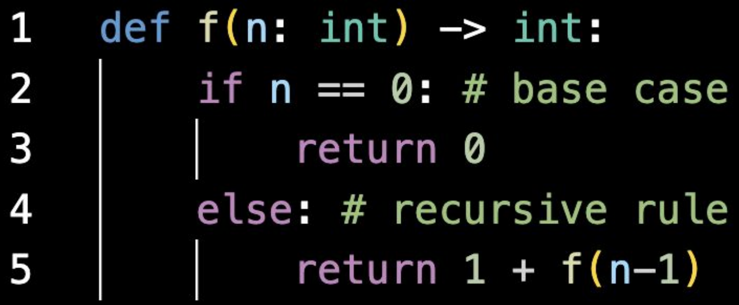
- Recursive rule:
  - for n > 0, f(n) = f(n-1) + 1

Recap

Last class, we started with standard function definition and followed steps to define it as a recursive python program.

Steps:

- Standard function
- Sequence representation
- Recursive definition
- Recursive Python function



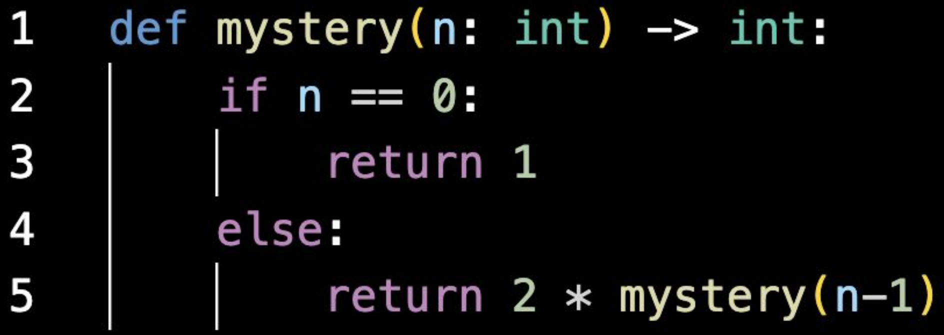
```python
def f(n: int) -> int:
    if n == 0: # base case
        return 0
    else: # recursive rule
        return 1 + f(n-1)
```

Today: Do it in Reverse

- Start with recursive Python function
- From that, get the recursive definition
- From that, get the sequence representation
- From that, get the standard definition

Today: Do it in Reverse

- Start with recursive Python function
- From that, get the recursive definition
- From that, get the sequence representation
- From that, get the standard definition



```
def mystery(n: int) -> int:
    if n == 0:
        return 1
    else:
        return 2 * mystery(n-1)
```