

CSC111 Prep 3: Recursively-Defined Functions

Definition 1 (formula)

Domain \mathbb{N} , codomain \mathbb{R} .

$$f(n) = \begin{cases} 4.0 & \text{if } n = 0 \\ f(n-1) + \frac{4 \cdot (-1)^n}{2n+1} & \text{if } n > 0 \end{cases}$$

Definition 2 (formula_multiple_args)

Domain $\mathbb{N} \times \mathbb{R} \times \mathbb{R}$, codomain \mathbb{R} .

$$f(n, a, b) = \begin{cases} 0.0 & \text{if } n = 0 \\ a \cdot f(n-1, a, b) + b & \text{if } n > 0 \end{cases}$$

This illustrates a recursive definition for a function with multiple arguments, where the recursion occurs on one of the arguments.

Definition 3 (double_recursion)

Domain $\mathbb{N} \times \mathbb{N}$, codomain \mathbb{N} .

$$f(n, m) = \begin{cases} m & \text{if } n = 0 \\ 2 \cdot f(n-1, n) & \text{if } n > 0 \text{ and } m = 0 \\ 3 + f(n, m-1) & \text{if } n > 0 \text{ and } m > 0 \end{cases}$$

This illustrates a recursive definition for a function with two arguments, where the recursion occurs on *both* arguments.

Definition 4 (create_list1)

Domain \mathbb{N} , codomain: the set of all possible lists of integers.

$$f(n) = \begin{cases} [0] & \text{if } n = 0 \\ 2 \cdot f(n-1) + [n] & \text{if } n > 0 \end{cases}$$

This illustrates how we can have a recursive function that returns a *list* rather than a number. Note that here we're using the same definition of “list multiplied by natural number” and “list + (concatenation)” as in Python.

Definition 5 (create_list2)

Domain $\mathbb{N} \times \mathbb{N}$, codomain: the set of all possible lists.

$$f(n, m) = \begin{cases} [m] & \text{if } n = 0 \\ [n] & \text{if } m = 0 \\ [f(n-1, m), f(n, m-1)] & \text{if } n > 0 \text{ and } m > 0 \end{cases}$$

This illustrates two new ideas: a recursively-defined function with more than one base case, and how a recursively-defined function can even return a *nested list*, i.e., a list that contains other lists.