Learning Objectives

Review fundamentals of recursion

Implement recursive functions to handle a variety of tasks
Recursion

Who Would Win?

- Highly complex recursive calls
- Simple and basic loops
Recursion

The success or failure of this lab (and the time it takes you) depends on your ability to answer the following:

**Base Case:** What is the smallest sub-problem? What is the trivial solution?

**Recursive Step:** How can I reduce my problem to an easier one?

**Combining:** How can I build my solution from recursive pieces?

*Let's work together to brainstorm some of the following functions!*
Each exercise a fun new twist!

Sum of Digits:

Triangle:

Palindrome:

Fibonacci:

Bonus List Partitioning:
Recursion Practice: Sum of Digits

Given a number, return the numerical value of summing each digit.

\[ 2 + 7 + 7 = 16 \]

\[ 1 + 1 + 1 = 3 \]
Recursion Practice: Sum of Digits

Given a number, return the numerical value of summing each digit.

**Base Case:**
Any length 1 digit \( \rightarrow \) return the digit!
\[ \text{length}(\text{str}(n)) = 1 \]

**Recursive Step:** Separate one digit
Call function again \([\text{Sum}(\ ))\]

**Combining:** One digit + the rest \(\frac{\text{one digit}}{1} + \text{Sum( the rest })\)
Recursion Practice: Triangle

Given the height of a triangle, how many total blocks were used to make it?

Base Case:

- $h = 0 \rightarrow 0$ blocks
- $h = 1 \rightarrow 1$ block

Recursive Step:

Triangle of height one smaller

Combination Step:
Recursion Practice: String Palindrome

Given a string, return whether it is a palindrome or not (True or False)

AAA  yes

racecar  cellular

racetrack  X
Recursion Practice: String Palindrome

Given a string, return whether it is a palindrome or not (True or False)

Base Case:
- 0 letters - is a palindrome
- 1 letter - is a palindrome
- 2 letters - may be a palindrome

Recursive Step:
Remove first & last letter
Look at first & last letter. If same continue; if different return False

Combining:
Recursion Practice: Recursive Fibonacci

Given a number $n$, return the $n$th Fibonacci number:

$$Fib(n) = Fib(n - 1) + Fib(n - 2), \quad n > 1$$

Base Case:

$\text{Base Case: } n = 0 \rightarrow 0$

Recursive Step:

$$\text{Recursive Step: } n \in Fib(n-1) \cup Fib(n-2)$$

Combining:

$$\text{Combining: } n^{th} \# = Fib(n-1) + Fib(n-2)$$
Using all elements in a list, can we make two lists which have equal sums?

**Input**

- `[4, 3, 1]` (T)
- `[3, 1]` (T)
- `[1]` (T)
- `[]` (F)
- 

**Check every permutation**

- `[1, 3, 4], []` (T)
- `[1, 4], [3]` (T)
- `[1, 3], [4]` (T)
- `[], [1, 3, 4]` (F)
- `[3, 4], [1]` (T)
- `[4], [1, 3]` (T)
- `[3], [1, 4]` (T)
- `[], [1, 3, 4]` (F)
Recursive List Partitioning

**Base Case:** When my input list is empty, I have tried every permutation

**Recursive Step:** Given list L, pop(L[0]) to left *and* right and recurse on both

**Combination Step:** If either partition recursion is True, return True