Chapter 9, Miller 3rd ed, Recursion

1. Problem vs. Instance of a problem

   Problem: Sum items in a list, sumList(L)
   Instances: sumList([4, 2, 4]), sumList([9, 4]), sumList([3]), sumList([])

2. Recursion
   Solve an instance of a problem
   → By solving smaller instance(s) of the same problem

3. For the Sum List problem
   Answer = first element + sum of remaining elements

   ```
   sumList([1, 2, 3])   =   1 + sumList([2, 3])
                        =   1 + 5
                        =   6
   ```

   In general, this is **`sumList(L) = L[0] + sumList(L[1:])`**

4. Turn this into a recursive function
   a. Naïve attempt:

   ```
   def sumList(L):
       return L[0] + sumList(L[1:])
   ```

   Gives index out of range error for `sumList([1, 2, 3])`

   ```
   sumList([1, 2, 3])  =  1 + sumList([2, 3])
                       =  1 + 2 + sumList([3])
                       =  1 + 2 + 3 + sumList([])
                       =  error: L = [] → L[0] is an error
   ```

b. Need to say when the computation is finished → called the **BASE CASE**
When L = [] → sumList(L) = 0

```
sumList([1, 2, 3])  =   1 + sumList([2, 3])
                    =   1 + 2 + sumList([3])
                    =   1 + 2 + 3 + sumList([])
                    =   1 + 2 + 3 + 0
                    =   6
```

```
def sumList(L):
    # Check for base case
    if L = []:
        return 0

    return L[0] + sumList(L[1:])
```

See 09-01-sum-list.py

5. When using recursion → Need to answer three questions:
    1. How do I make the smaller instance (What does it look like)?
    2. How can I use the solution of the smaller instance to solve this instance?
    3. What is the base case? What is the solution to the base case?

    a. For the Sum List problem:
        1. **Smaller Instance:** List without first element
        2. **How to use smaller instance:**
           solution = (1st element) + (solution to smaller instance)
        3. **Base case:** Empty list → solution is zero

6. More about the base case
    a. Base case is the smallest possible instance of the problem.
    b. Can be solved trivially, without other instances of the problem.
    c. For sum List: sumList([]) = 0

7. Common errors
    → Not checking for base case
    → Recursive step makes no progress toward base case

8. More Examples
    a. Smallest in List
    Problem: Find smallest element in a list

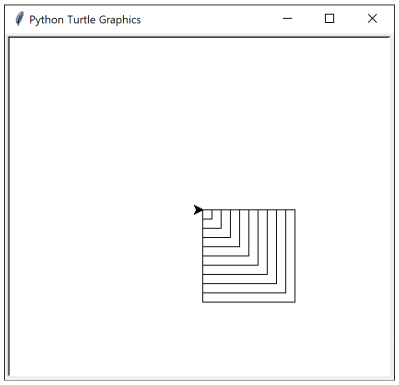
    Function call: `smallest(L)`

    Smaller instance: List without 1st element, [1:]
    How to use smaller instance: smallest = min(first, smallest([1:])
    Base Case: len(L) == 1 → smallest = L[0]

    See 09-02-smallest-in-list.py

    b. Draw Nested Squares
    Given largest side length → nested to side length == 1

    Function call: `nestedSquares(t, sideLen)`



    Smaller instance: Draw nested squares starting with smaller side length
    How to use smaller instance:
    1. Draw this square
    2. Draw nested squares with smaller side length

    Base Case: sideLen ≤ 1 → don't draw anything, just return

    See 09-03-nested-squares.py