>>>UNDERSTANDING THE PROBLEM

a. Can I restate the problem in my own words?

b. what inputs go into the problem

c. what outputs should come from the solution to the problem

d. Can the outputs be determined from the inputs? in other words, do I have enough information to solve the problem?

(You may not be able to answer the question until you set about solving the problem. That's okay.

it is still worth considering the question at this early stage.)

How should I label the important pieces of data that are part of the problem?

>>>EXPLORE EXAMPLES

a. Start with simple examples

b. progress to more complex examples

c. explore examples with empty inputs

d. explore examples with invalid inputs

>>>BREAK IT DOWN

a. write down comments on your process to solve the problem.

>>>SOLVE / SIMPLIFY

Simplify

a. find the core difficulty in what you're trying to do

b. temporarily ignore that difficulty

c. write a simplified solution

d. then incorporate that solution back in

>>>LOOK BACK AND REFRACTOR

Refactoring questions::

-- Can you check the result?

-- Can you derive the result differently

-- can you understand it at a glance?

-- Can you use the result/method for some other problem?

-- Can you improve the performance of your solution?

-- Can you think of other ways to refactor?

-- How have other people solved this same problem?

>>>SOME PROBLEM-SOLVING PATTERNS

-- Frequency encounter

-- Multiple pointers

-- Sliding window

-- Divide and conquer

-- Dynamic programming

-- Greedy algorithms

-- Backtracking etc....