PEDAC - **A problem solving approach**

PEDAC

Created by Launch Code

Documented in a blog post `Solving Coding Problems With PEDAC`

Breaks down problem solving into five phases.

- `P` - Problem
- `E` - Examples
- `D` - Data Structure
- `A` - Algorithm
- `C` - Code

A problem to solve with PEDAC

Given a list of numbers, perhaps very long, find the largest number.

Restrictions

- We can only look at one number from the list at a time.
- We can only keep track of one additional number.

P Understanding the Problems

Identify the key ideas

Given a list of numbers, perhaps very long, find the largest number.

- List
- Number
- Largest
- Long

P - Restate the problem in more specific terms.

Go through the list, one number at a time, and keep track of the largest number we see.

- Getting more specific.
- How specific can you make the general statement?
- Look for special cases or conditions.
- The more we can refine the problem statement the better our work will be in other steps.

E - Examples

Generate example inputs and see if you can work out the answer.

- Look for edge cases.
- `0`, `1`, `infinity`

E - Examples

- How many examples do I need?
- As many as you feel give you a better understanding of the problem statement.

E - Examples

Input	Description	Output
Empty List	Empty	0?
42	One Number	42
5, 4, 3, 2, 1	Decreasing Numbers	5
1, 2, 3, 4, 5	Increasing Numbers	5
6, 12, 12, 12, 4	Repeating Numbers	12

D - Data Structure

What kinds of variables will we be dealing with?

- What is the type of the input?
- What is the type of the output?
- What kind of "local" information will we need to keep track of?

D - Data Structure

- If the input isn't in a format that helps us, what could we convert it to?
- If we receive data that is of kind X, but is easier to deal with if it was Y, can we transform X into Y?

D - Data Structure

- Input: A list or array of numbers
- Output: A single number
- Local information: A number to keep track of a "largest"

A - Algorithm

This is the main part of the work. This is where we earn our paycheck.

An algorithm is a formal set of instructions for solving a problem.

You've used them all your life:
- Directions to your friends house.
- Favorite recipe for cookies.
- Long division.
- Swinging a bat and hitting a ball.

A - Algorithm

- Head north on Main Street
- Turn left onto highway 42
- Take the highway for three exits
- Exit on Parker Street
- Turn right on Palm Ave
- Look for the fourth house on the right
- Stop.

A - Algorithm

This step might take the longest.

Might have to restart and rethink.

Might have to go back to step P and think again.

Will need to try out your algorithm steps on your example data.

1. If the list is empty, the answer is 0, stop.
2. Start at the first element of the list
3. That number is your largest so far
4. Move to the next number
5. If we are at the end of the list stop and return our answer
6. If the current number in the list is larger than `largest so far`, replace `largest so far` with this number.
7. Move to the next number in the list
8. Go back to step 5

Let's try it out

How do we get better?

Practice, practice, practice.

Make practice deliberate.

It will be painful.

But it will be worth it.

Suggestions

- Start to look at *everything* as a PEDAC: Homework, things from your daily life, real world problems.
  - Example: watch this video about making PB&J sandwiches (https://www.youtube.com/watch?v=cDA3_5982h8) and see if you could do better.
- When working on code challenges don't take shortcuts with google searches.
- Give yourself permission to **FAIL**.

Challenge

- Given a list of numbers, perhaps very long, sort the number in increasing order.
- You can only look at two numbers from the list at a time.
- You can ony swap two numbers from the list at any time.