Start this exercise in groups of 4-6 students.

Images for part 1:

Part 1:
Split your group into three smaller groups (1-2 students) each of which takes one of the following problems. If two students are in a small group, each does the whole exercise.

In each exercise draw the data structure and, using a series of images and/or clear labelling on the image(s), show in as much detail (small steps) as possible what happens in each operation.

1. Look at the image of a list of characters
    a. What happens when the value ***u*** is appended?
    b. What happens when the value ***g*** is inserted at index **2**?
    c. What happens when the item at index **4** is set to ***h***?
    d. What happens when the value at index **3** is removed?

2. Look at the image of an ordered list (*it should still be ordered after each operation*)
    a. What happens when the value 4 is added?
    b. What happens when the value 1 is added?
    c. What happens when the value 9 is added?
    d. What happens when the value 5 is removed?

3. Look at the image of a list of characters
    a. What happens when the value ***w*** is appended?
    b. What happens when the value ***h*** is inserted at index **3**?
    c. What happens when the item at index **0** is set to ***f***?
    d. What happens when the value at index **1** is removed?

4. Look at the image of an ordered list (*it should still be ordered after each operation*)
    a. What happens when the value 7 is added?
    b. What happens when the value 0 is added?
    c. What happens when the value 8 is removed?
    d. What happens when the value 5 is removed?

Now students show the members of their larger group their solutions and ***explain in exact detail*** the **steps** of each operation. If more than one student did the same exercise they take turns explaining the operations (not both the same ones).
Other students should feel free to add detail if they feel something is being "jumped over".

Problems for parts 2 and 3:

A. *Build a class that holds a collection of objects called pizza*
   - Make an operation that adds a pizza with 1-3 toppings
     - Each pizza should be ***unserved***
     - The operation should return a unique ID for this pizza
   - Make an operation that marks a pizza ***served***
     - Use the unique ID to choose this specific pizza
   - Make an operation that returns a string representing all the pizzaz
   - Make an operation that removes all pizzaz that have been served

B. *Build a class that holds a collection of streets and neighborhoods*
   - Make an operation that adds a new street
     - You must also pass the neighborhood to the operation
   - Make an operation that returns a string representing all streets in a neighborhood
     - Use the neighborhood name to identify the correct streets
   - Make an operation that returns the name of the neighborhood of a street
     - Pass the name of the street and get the correct neighborhood

C. *Build a class that stores log entries*
   - Make an operation that adds to the data structure
     - Add the time of entry and a string with the actual log entry
   - Make an operation that returns a string representing all the items
     - Make it possible to send in a start time and end time
     - Only return the logs within the time interval
   - Make an operation that deletes entries in a time interval
   - Make an operation that returns the newest log entry
     - Make it possible to ask for a certain number of newest entries

<u>Read both part 2 and 3:</u> different students will design tests for a problem and implement it.
You are not expected to fully implement these problems. First design tests.

Part 2:

Split your group into three smaller groups (1-2 students) each of which picks one of the problems. If two students are in a small group, they can cooperate.
It is also OK to only have two smaller groups and only pick two of the assignments.

Each smaller group does the following:

- Decide on the names of classes and public operations
    - Public operation are the ones that will be called from outside the class
    - Private operations are the ones that are only used internally (helpers)
- Set up these classes and operations in python code
    - Use “pass” if operation does nothing
    - If operation should return a value, return a dummy value of the correct type.
- Design tests for the operations and implement the tests in python code
    - Call the actual class and make sure everything works
        - Even though it will not return correct results
    - Make sure the tests evaluate every possibility
        - Think about “edge cases”
    - Know what you expect the tests to return
        - Or that you can easily see from the output what should be returned

Now the big group meets again and each smaller group explains their tests.

- Give detailed reasons for your “edge cases”.
    - What is the possible case
    - Why is it different from the general case
    - How does a particular test evaluate exactly this case

After each explanation the larger group discusses the tests and tries, collectively, to identify cases that these tests do not evaluate well enough. Get into details here!!

- The smaller group now updates/rewrites their tests after the group review.

Part 3:

Again split the group into smaller groups.
Each smaller group picks one of the problems that has a class definition and set of test cases from Part 2, but **not** one that any member of this smaller group worked on in Part 2.
Every student should thus be in a group that starts with implemented test cases that they didn't implement themselves.

Each group does the following:

- Design the operations, variables needed, helper classes, etc.
    - Define all this on a code level, but don't necessarily get the code working perfectly.
    - Write explanations of the operations in very good detail
        - Comments and pseudo-code
        - Diagrams
    - Write this up in a way that it can be understood without personal explanations

Now each group takes another group's design

- Seek to understand it
- Try to point out issues
    - Some case that is forgotten or left out
    - An endless loop that might occur
    - Somewhere no value will be returned
    - A change that could make the whole structure more efficient

The larger group meets again to discuss the review of all designs

- In each solution, discuss possible ways of doing it differently
    - whether or not it's necessarily better.
- Also discuss ways it might have been worse

Groups take back their design, fix it and **implement it**

- *The larger group can also pick* **one** *of the problems and try to implement that together.*
- Now they have actual working code that should run correctly through tests from **part 2**
- Don't expect to fully finish this for all the problems (or even any of them).

Now the groups can take their code back or the whole big group can cooperate on making each solution as good as it can be...