Problem-first classrooms

In a problem-first classroom, learning begins with a challenging, authentic problem rather than pre-packaged theory. In my Compilers class, the teacher first showed us PLs designed by students in the previous year, and put creating such a PL as the final goal. When designing our own compilers, we proposed solutions, debated ideas, and experimented, while the teacher introduced concepts, symbols, or terminology to the class only when they became necessary to make progress. Each idea was instrumental in resolving doubt, following Dewey's reflective cycle of confusion, hypothesising, testing, and reconstructing understanding. Similarly, in my OS and Computer Networks classes, students were asked to critique naive solutions, identify bottlenecks or edge cases, and iteratively evolve systems, mirroring real-world design. JEE coaching also applies the same principle: concepts are taught just-in-time to solve problems, not as abstract goals, and each worksheet problem is a reflective experience in itself (guessing methods learnt from past problems to solve it and learning from the current trial). In short, in my problem-first classroom;

- The teacher first asks the students the *questions* that are solved the theory/technique they are about to learn in the next few lectures. This creates doubt and confusion in the mind.
- The *students propose/guess* either complete or partial solutions.
- The class and teacher go over different solutions proposed by students.
- The teacher highlights any flaws in the solution or any hidden nuance in the question that went un-noticed. The class has thus completed testing the proposed solution.
- The final solution that came from the discussion is compared to the actual solution given by experts. With a good discussion and a good enough teacher, these two are usually close. Finally the teacher *explains* the actual solution to the class.

This process is fairly easy to achieve for fields such as mathematics, physics, and especially computer science.

For more subjective fields, such as politics or humanities where an in-class discussion is much harder to direct to the correct end-point, the problem-first approach can instead be applied with the help of a personal AI agent for each student which has been told (in its system prompt) to contrast the student's approach with any particular theory and point out how and why the student's approach is to be modified to meet the accepted solution.

While the 5 stages of Deweyan cycle are already there in this approach; we also want to have *instrumentality* of concepts for the less curious, grade oriented students. To do this, the teacher can do a depth-first traversal of the concepts needed to solve a hard last-year question (a goal which many students would want to accomplish). This will go as follows:

- The teacher presents the question to the class and asks the class about what things seem unfamiliar in the question (asking for where to go next) and notes it down.
  Then he starts explaining the first concept in the list he noted down. Once again, he should ask the students what feels unfamiliar. This keeps on going till the students don't find anything unfamiliar, that is, a *primitive concept* has been reached.
- The teacher then pops this concept from his to-teach list/stack and starts teaching the last thing that the students asked questions about. If yet another question arises, the teacher should put that new thing on the to-teach stack.
- After every "major" (depends on context) concept is completely finished, the teacher might put assignments or quizzes *only* on that concept.
- Eventually every pre-requisite for the question is finished, and so the teacher can ask the students to solve it and later *show* the kids how to solve it.

This, top-down; recursive approach to teaching rather than the traditional bottom-up approach helps give the students incentive to study the many seemingly un-related primitive concepts that are needed for the derived concepts and ultimately the last-year question. Instrumentality of the current concept is present at any given moment in this manner. A lot of books are in-fact written in this manner; and experts (at least software engineers) often work in this top-down manner.