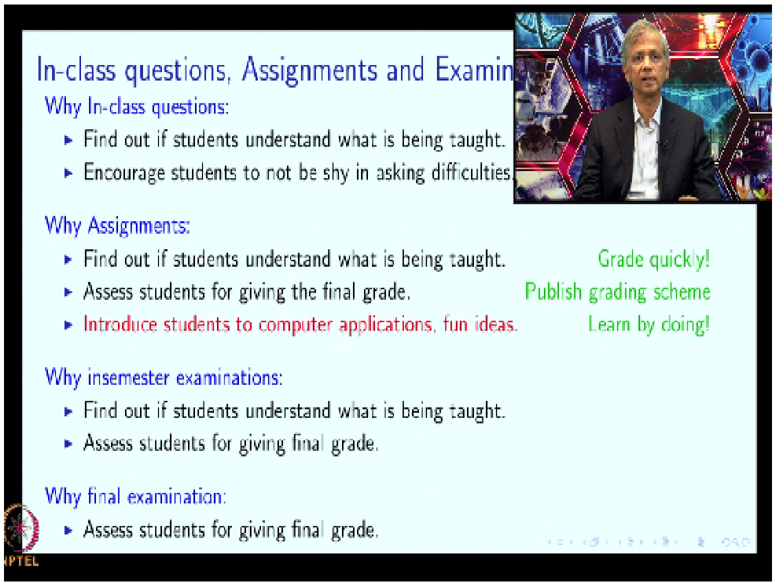
Design and Pedagogy of The Introductory Programming Course

Prof. Abhiram G. Ranade
Department of Compute Science and Engineering
Indian Institute of Technology - Bombay

Lecture – 20
In class questions, Assignments, Examination. 0: In class questions and lab assignments

Hello. Welcome to this final sequence of lectures in the course on Design and Pedagogy of the Introductory Programming Course. The topic for these lectures is In class Questions, Assignments and Examinations.

(Refer Slide Time: 00:37)



In-class questions, Assignments and Examin

Why In-class questions:
- Find out if students understand what is being taught.
- Encourage students to not be shy in asking difficulties

Why Assignments:
- Find out if students understand what is being taught. Grade quickly!
- Assess students for giving the final grade. Publish grading scheme
- Introduce students to computer applications, fun ideas. Learn by doing!

Why insemester examinations:
- Find out if students understand what is being taught.
- Assess students for giving final grade.

Why final examination:
- Assess students for giving final grade.

So let us first go over why In-class questions are necessary. The most obvious reason is that we want to find out if students understand what is being taught. This is to be considered as the feedback to the students that look you are perhaps not understanding what I am teaching as well as feedback to the teachers, look students are not understanding. So what can you do to help them understand.

So it is something necessary for the students as well as the teacher. Now in-class questions are necessary so that the classroom atmosphere is a little light so that students are encouraged to ask questions if they are having difficulties. If the students are not accustomed to hearing voices of other students and if the students are not accustomed to seeing other students ask questions or even just answer questions, they may have difficulty in raising their hands, they may be shy to raise their hands and ask their own difficulties.

Why should you have assignments? So again assignments are there to find out if students understand what is being taught and since this is meant as feedback to students, you should grade assignments reasonably quickly and return papers or return the marks on the assignments so that students know that they are not understanding or they are understanding certain topics. On the assignments, you can tell them what they need to improve upon.

Of course, this is all fairly obvious but I think it is worth going over it. And of course, assignments are used for accessing students as well. So we have to decide whether a student should get a high grade or a low grade and for that purpose, we need to have assignments. Assignments form an important part of the insemester evaluation. So insemester evaluation is sort of study evaluation which happens over the entire semester so that presumably the stress is less.

And if you are giving assignments again for the comfort of the students and general fairness, you should publish a grading scheme. Then really important reason for giving assignments is actually to teach something new. So in class, you cannot expect students to sort of assimilate all details or you cannot expect to teach a lot of things. For example, you cannot expect to have time to teach applications of computers.

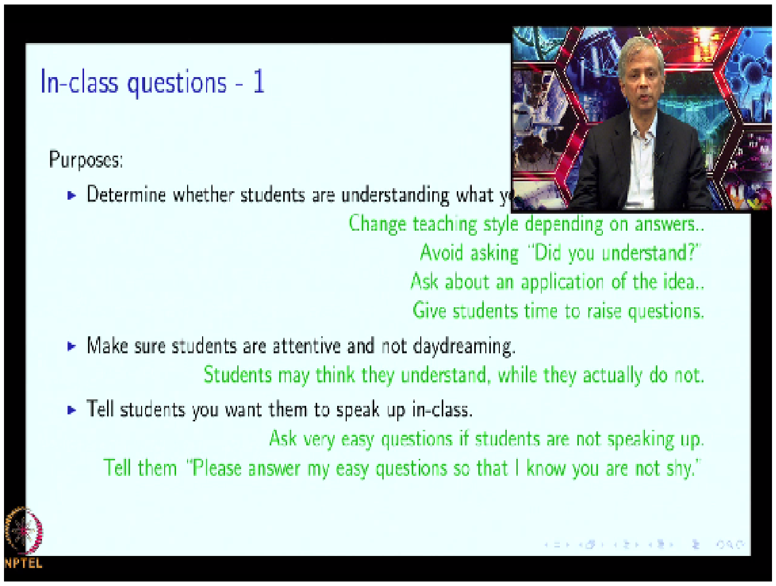
So when you give an assignment on the other hand, you can describe an interesting problem for which they have to write a program and that can be given at some level of detail. So students can read it, students can think about it. So you can sort of give slightly bigger problems and problems which might have some interesting ideas as well and in fact, you can introduce students to some of the major ideas in computer science through assignments. So this is sort of learning by doing.

So assignments, I guess, the major purpose really is learning by doing. So feedback is important and the assessment is important but really, we have assignments so that students can learn by doing and in the practical course like programming, leaning by doing is absolutely essential. Why should you have insemester examinations? So again, it is so that you find out whether the students are, are understanding what is being taught. So it is like assignments.

So again, it is important that you grade the examinations quickly and return the answer books so that the students have the opportunity to act on the feedback. And again the insemester exams are presumably a more relaxed way of giving the final, making the decision about the final grade but even here again, just like assignments, you should publish a grading scheme. You should allow students to complain if, if they feel they have less marks because remember that when they complain, they are seriously thinking through the topics that you are teaching and therefore, they are learning.

So encourage that and for that, publish the grading scheme and let them argue it out. The final examination usually serves just one purpose, assess the students for giving the final grade. Even here many universities allow students to look at the final examination papers and report complaints about the grading. So I think in the interest of fairness, I think we should strive for doing this.

(Refer Slide Time: 05:59)



In-class questions - 1

Purposes:

- Determine whether students are understanding what y
  - Change teaching style depending on answers..
  - Avoid asking "Did you understand?"
  - Ask about an application of the idea..
  - Give students time to raise questions.
- Make sure students are attentive and not daydreaming.
  - Students may think they understand, while they actually do not.
- Tell students you want them to speak up in-class.
  - Ask very easy questions if students are not speaking up.
  - Tell them "Please answer my easy questions so that I know you are not shy."

Let me says a little bit more about in-class questions. So purposes, we said that we should ask in-class questions to determine whether students are understanding what you just taught. So you should teach and ask questions and depending upon how students are answering, you should change your teaching style. You should go slow may be. Now how you decide whether students are understanding what you just taught? You could ask, did you understand? Okay.

But did you understand is, it does not elicit answers very well. Students have a temptation to say yes I understood, especially the shy students and sometimes even if they do not understand, they may not know exactly how to explain what they did not understand. A much better way of doing this is to ask about an application of what you just taught. So it could be something small, it could be something crisp and then, but it should be something in which the student has to apply what you just taught.

So over here, if you do not get the correct answer, then you know that what is being taught is not being understood. Now different people have different styles regarding asking questions. You could ask, you could throw a question to the class at, at large or you could point to specific students and ask questions of them. So this depends on your temperament and this depends upon how you see your students.

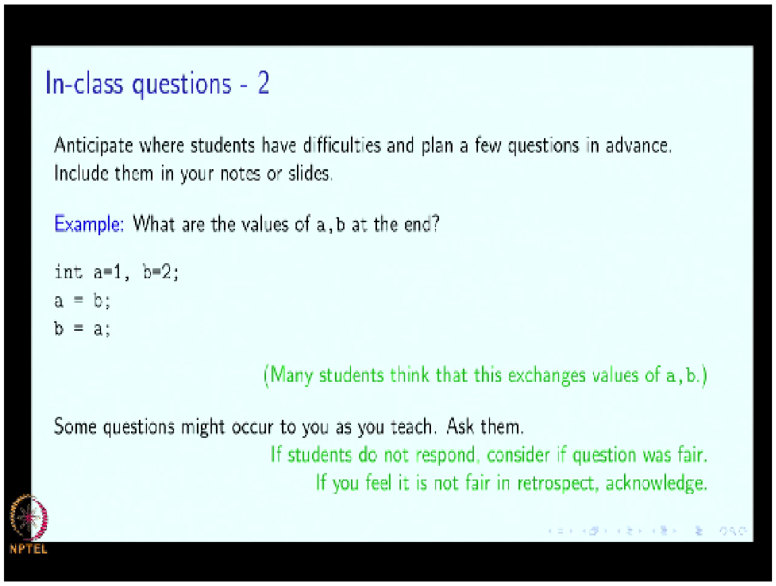
If you are seeing that some students are continuously looking perplexed, then you should direct questions at those students but often students dislike being put on a spot and so some tradeoff you have to decide depending upon your own temperament I guess. And this is important. So just as you ask questions, student will want to ask questions. So students might have some difficulty. So leave some pauses, leave some silence.

Do not, do not be, do not be upset or do not be perturbed if nobody speaks anything for a few seconds. So sometimes silence make people uncomfortable but if that happens, that is okay. So that is an opportunity for students to ask questions. So you should wait. The other reason for asking questions is to make sure that the students are attentive and they are not daydreaming, okay.

So they are paying attention to you. And here, you should note that even if students are actually paying attention and they may feel that they understand, okay, so they may think that they are paying attention, they may feel that they understand but they actually are not getting some subtle features. So if you ask questions, you will encourage them to listen more carefully. And when you ask a question and you wait, you are telling them that look you should speak up in class.

You should participate, that is important. For this, you could even ask extremely easy questions. So if you feel that students are not speaking up, ask very easy questions. So you can tell them, please answer my easy questions so that I know that you are not shy. I know that you are willing to ask questions and I know that you are not asking questions because you are understanding everything. So get them to talk.

(Refer Slide Time: 09:47)



In-class questions - 2

Anticipate where students have difficulties and plan a few questions in advance. Include them in your notes or slides.

Example: What are the values of a,b at the end?

```
int a=1, b=2;
a = b;
b = a;
```

(Many students think that this exchanges values of a,b.)

Some questions might occur to you as you teach. Ask them.

If students do not respond, consider if question was fair.

If you feel it is not fair in retrospect, acknowledge.

You have to anticipate where students have difficulties and plan a few questions in advance. You can include them in your notes or slides however you teach. Here is an example. Here is a piece of code in which you set first a=1, b=2, a=b, b=a. You may think that this is a very simple questions asking them what is the value of a and b at the end. However, educational research show that students do not immediately understand the assignment statement.

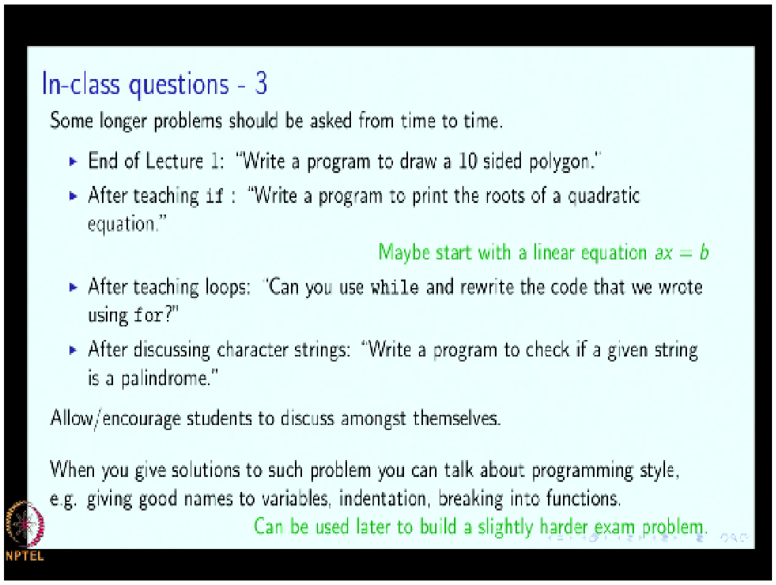
They sort of gloss over the fact that assignments are executed one assignment at a time top to bottom. So they may think that the values of a and b are going to be exchanged which is not the case at all. So by looking at the educational literature or by knowing your own experience, you know where students might have difficulties and where they are likely to make mistakes, so you put those questions in your slides or in your notes and ask them upfront so that their, their misunderstandings will be corrected very early on.

Now of course, everything does not have to be planned, okay. As you teach, some questions might occur to you. So just go ahead and ask them, okay. So all questions that you ask, do not have to be great questions, okay. So it might so happen that you ask a question, you think that the answer should be obvious, you think that somebody, if somebody answers the question, will not it be great because that would indicate that somebody in the class is really thinking.

However, it might so happen that nobody responds. So in that case, especially if it is a question that you have asked spontaneously, you should think about whether that question was fair or not. Perhaps it might so happen that you ask the question and then you realize that no, no, no, that was a very difficult question. The students did not have enough, enough training to answer that question.

So if you think that is the case, and you may think so in retrospect after asking the question, you should just acknowledge that. You should say, oh, sorry about that. I thought, I, now I think that this is perhaps not a fair question, okay. So maybe then go on to the next topic just or, or explain to them what the answer was, of course.

(Refer Slide Time: 12:36)



In-class questions - 3

Some longer problems should be asked from time to time.

- End of Lecture 1: "Write a program to draw a 10 sided polygon."
- After teaching `if` : "Write a program to print the roots of a quadratic equation."

  Maybe start with a linear equation $ax = b$
- After teaching loops: "Can you use `while` and rewrite the code that we wrote using `for`?"
- After discussing character strings: "Write a program to check if a given string is a palindrome."

Allow/encourage students to discuss amongst themselves.

When you give solutions to such problem you can talk about programming style, e.g. giving good names to variables, indentation, breaking into functions.

Can be used later to build a slightly harder exam problem.

Now in class, you should ask some longer problems from time to time, not just lecture. So ask some longer problems and give students time to solve. So end of lecture 1, where we talked about turtle graphics, where we talked about drawing the square, you could ask write a program to draw a 10-sided polygon and just give them 5-10 minutes where they try to write the program and then show them the answer, okay.

So for example, when you talked about the if statement, you could say now write a program to print the roots of a quadratic equation. Or you could start off with something simpler. So you could say that look, so print the roots of a linear equation so that they have to read a and b and print x=b/a. So this is real trivial but there is a catch. So the catch is that if a is 0 then you should not be dividing by a.

So wait for them to write that down and explain to them that they should really be testing. Whether a is 0 and if so, say that this has no solution; otherwise, print the solution. After teaching loops, you could ask, can you use while and rewrite the code that we just wrote using for? So this is a good test of not only the while, understanding of the while statement but also of the understanding of the for statement.

After discussing character strings, you could say, okay, write a program to check if a given string is a palindrome, okay. Now when you ask such questions, you should encourage students to discuss amongst themselves. So in high school or in primary school especially teachers are worried if the class becomes noisy. Here if the class starts talking with each other, if they become animated, then you know that they are, they are sort of participating, they know that they are understanding the material.

They know that they are taking a stand and they are trying to explain to their friend and they are doing your own work. They are doing the work of teaching. So, so encourage them. So even when they explain, they are learning. Even if they are hearing the explanation from their friends, maybe they are learning better because the friend sometimes know exactly what to say, what to say in the language that their friends can understand.

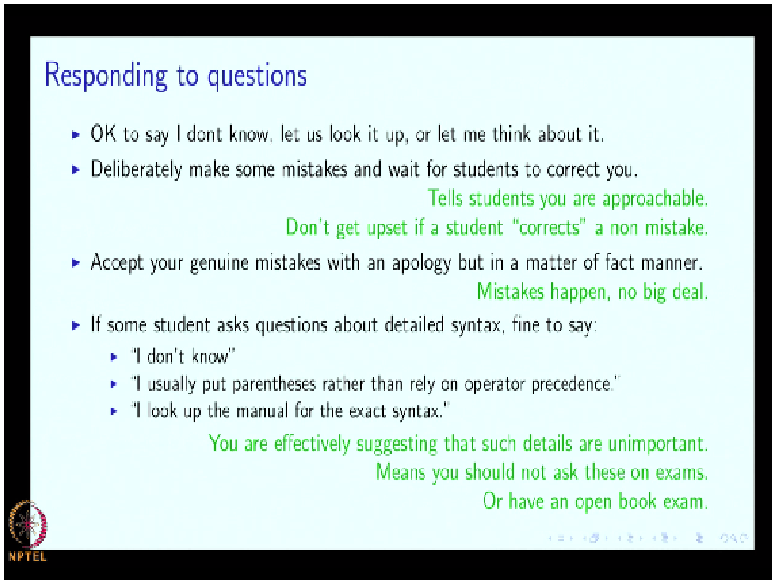
So friends can be better teachers. So let them explain, let them do your work. Then at the end of such, such a question, you have to give the solution and at that point, you can talk about programming style. So you can say that look when you write this code, you should give good names to variables, you should indent, maybe you should even break things into a function, okay.

So all of these things which are sort of style issues, could be brought in after these longer problems are asked. And it is important to ask questions because once you ask questions, once you have a discussion, then you know that the students understanding at least the basic, the basics or you have, you have, you are telling them the solution to some basic problems. So then you can, then you can ask slightly more difficult problems in the examination.

If you had asked that same problem in the examination without having this discussion before, then perhaps that problem might have been unfair. So if you have already discussed sort of an easy problem and then you ask them a harder problem, then the difficulty of the harder problem is reduced and therefore, it becomes, possibly becomes a fairer question to ask in an examination.

So therefore, it is important that you ask questions in exams as well as in class and then use those questions to design your examination at the end. This of course assumes that you have the freedom and responsibility of designing an examination. Now I do want to say something regarding how to respond to questions. Again, some of this maybe obvious but I would like to tell you my own, my own feeling and my own experience regarding responding to questions.

(Refer Slide Time: 17:11)



Responding to questions

- OK to say I dont know, let us look it up, or let me think about it.
- Deliberately make some mistakes and wait for students to correct you.

  Tells students you are approachable.

  Don't get upset if a student "corrects" a non mistake.
- Accept your genuine mistakes with an apology but in a matter of fact manner.

  Mistakes happen, no big deal.
- If some student asks questions about detailed syntax, fine to say:
  - "I don't know"
  - "I usually put parentheses rather than rely on operator precedence."
  - "I look up the manual for the exact syntax."

  You are effectively suggesting that such details are unimportant.

  Means you should not ask these on exams.

  Or have an open book exam.

So I think it is perfectly okay to say to a question that I do not know. I do not know the answer to this or it is okay to say that look I do not know the answer to this, let us look it up. It is also okay to say I do not know the answer to this, let me think about it. I will tell you in the next lecture or something like that, that is perfectly fine. Students do not think less of you if you answer like this a few times.

Well if you do this all the time, of course that is a bad thing. But if you do this, once or twice or thrice in a semester, students are not worrying. Yes, of course, students are not worried if on the whole they feel that you are, you have their best interest at heart and you are putting in enough effort and they know that from time to time, everybody slips up, everybody makes a mistake and so long as they regard you as a friend, they are completely happy to, happy to forgive your mistake.

In fact, I would go further and say that we should deliberately make some mistakes in the class and having made those mistakes, you should wait for students to correct you. So this is to tell the students that it is okay to make a mistake, tell the students that you may make a mistake, tell the students it is that they should be alert and if they spot a mistake, you are not going to get angry but you are actually going to commend them for their alertness.

So I think, so do that, see what, how students respond to it. And it tells the students that you are approachable, that if you, if you are contradicted, you are not going to get angry. Once in a while, the students, the students may make, may point out something and say that look you have made a mistake. Do not get angry about it. Again, you want, you want to, you want to become, you want to establish some kind of an equality relationship.

Because an equality relationship, an equal relationship is good if students want to learn. If they are very respectful, then they are not thinking in a free manner. So let them be a little more relaxed. So once in a while it may see that they are becoming, they are becoming too smart, ignore it, okay. So do not respond to it, just move on. Just tell them that no, no, no, this is, this is actually correct and if you feel that it is not correct, let us discuss it after the, after the lecture.

But accept your genuine mistakes. Say you make a mistake and it has, it has been pointed out whether it is a planted mistake or whether it is a real mistake. Accept it with an apology and again as I said, do not get worry, talk about it. It is okay, people make mistakes. So you should accept it in a matter of fact manner and just continue. Sometimes students may ask questions about detailed syntax, okay. So should I put a semicolon after this or should I, should I parenthesize this or not and you may not know the answer or you may not care to even know the answer.

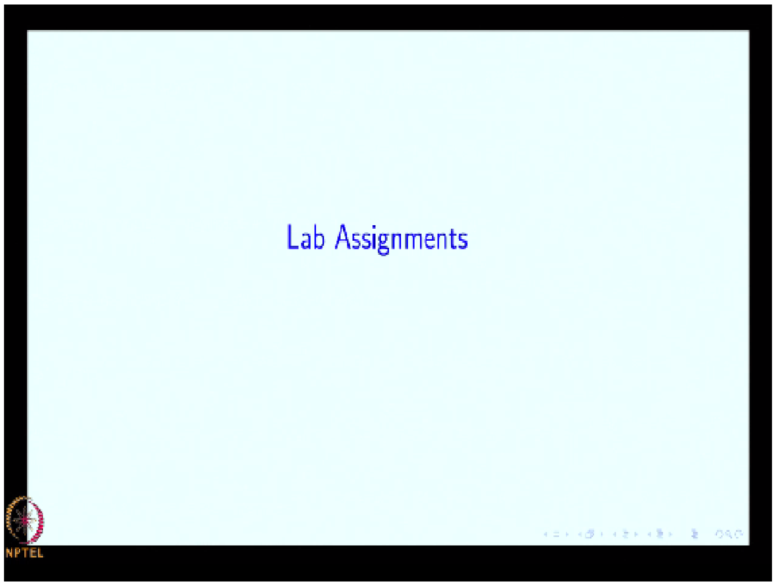
So first of all, you should say I do not know but you might say that look I usually put parenthesis rather than rely on operator precedence. So I do not really worry about this, does times operator have more precedence than the and operator, I do not worry about it. I just put parenthesis so that it becomes easier to read, so that nobody who is reading my code, has to really think too carefully, okay.

So I want to make the job of reading my code easy and therefore, I am going to put parenthesis for that purpose rather than myself trying to remember all the operating, operator precedence rules or expect others to remember the operator precedence rules. And if it, if such code comes to me, I will look up the manual for the exact syntax. So I am not worried about making syntax mistakes, okay.

So that is perfectly fine to say and it sort of says to the students that look syntax mistakes are important but not that important. There are bigger things then syntax mistakes that, that we should be really worried, talk about. So these details are unimportant but then it also puts a responsibility on you. Then that means you really should not be asking syntax-related questions in your exams as well.

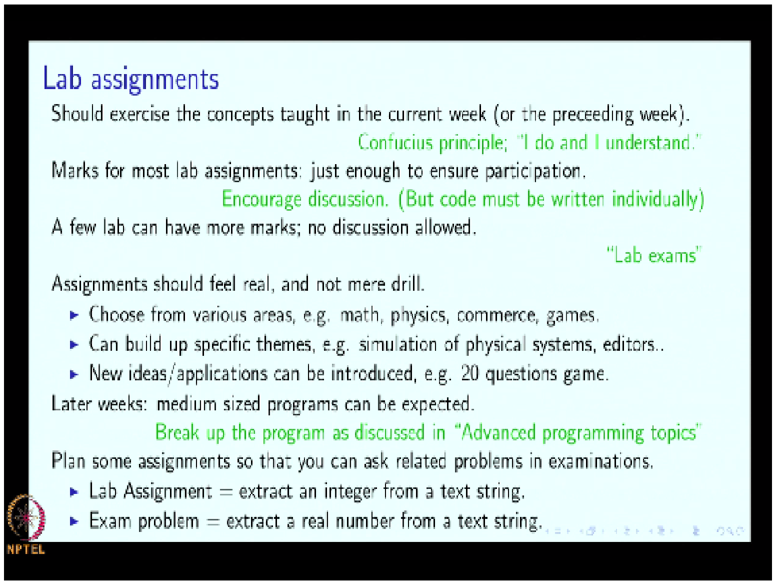
Because if you do ask questions, then students say, oh, then that means they are important. So instead of that, do not ask such questions. Ask questions more about, about interesting, more interesting points. Well if you have an open book exam on the other hand, then asking syntax questions is fine because students should know how to look up and where to look up so that they answer every possible question. The next topic I want to talk about is Lab Assignments.

(Refer Slide Time: 22:41)



Lab Assignments

(Refer Slide Time: 22:45)



Lab assignments

Should exercise the concepts taught in the current week (or the preceeding week).

Confucius principle; "I do and I understand."

Marks for most lab assignments: just enough to ensure participation.

Encourage discussion. (But code must be written individually)

A few lab can have more marks; no discussion allowed.

"Lab exams"

Assignments should feel real, and not mere drill.

- Choose from various areas, e.g. math, physics, commerce, games.
- Can build up specific themes, e.g. simulation of physical systems, editors..
- New ideas/applications can be introduced, e.g. 20 questions game.

Later weeks: medium sized programs can be expected.

Break up the program as discussed in "Advanced programming topics"

Plan some assignments so that you can ask related problems in examinations.

- Lab Assignment = extract an integer from a text string.
- Exam problem = extract a real number from a text string.

So for lab assignments, there are many reasons we said and the most important reason perhaps, well not the most important reason but the most natural reason is that they should exercise the concepts taught in the current week or the preceding week. So it is that Confucius principle, I just do not teach something but I force the students to do something with it and then the students will understand and also if I had just taught something, that idea is fresh in the minds of the students, so I am going to ask them to use it sort of strike while the iron is hot.

For lab assignments or I should say for most lab assignments, do not give too many marks. If you give enough marks so that participation is ensured, okay. And furthermore, I would say that you should even encourage discussion. Let the students go back to their hostels or their homes or whereever and talk to each other, okay. How do I do this problem? How did you do it, okay? Now they should discuss but they should not write the code together.

They can say that, oh, I am going to do this, okay. I, I do this, I do that, whatever it is, okay. But tell them that they should not discuss. They should not say that I am going to have a variable called this, okay. I am going to have a variable which, which I am going to first set this, this way, I am going to do it, do next with it and so on. So that is as good as writing the code together. So encourage discussion but tell them that the code must be written individually.

And in fact, there are tools available which will check whether the assignments that people have submitted and when I say lab assignments, they have to submit them and they have to, they have to be run in order be tested whether they are right or wrong. So you can have tools which will check whether the assignments are copied from each other. So that reporting will happen. So use those.

Use those to see whether students are copying assignments or not. But as I said the important point over here is encourage students to discuss, to participate but having participate, having discussed, they should write the code on their own. So create conditions so that students do not feel stressed, they do not feel completely lost but yet they feel the need to work and they feel the possibility that if they do work, they will get marks.

A few labs should have more marks and discussion should not be allowed, okay. So these are labs where the students will be given problems, the moment they come to the lab and they have to finish, by the time, when the lab time is up. So this is more like a lab exam. After all programming is a practice based subject and so you have to test whether students understand, the students can practice.

So it is a practical subject. So some lab exam like assignments must be there. Now if you want students to work enthusiastically on assignments, assignments should feel real. They should be fun. They should not be mere drill. So you should choose problems from different areas say math, physics, commerce, games. You can build up specific themes as you go through lab assignments.

They are, these themes might be slightly different from what you are teaching in the class as well. So for example, one theme which is very important to scientist and engineers so if your students are science or engineering majors, a simulation of physical systems. So when we showed the demo, you saw that how to do the simulation of a bouncing ball was one of the problems.

So such, such simulation of physical systems or simulation of logistic systems, so say maybe simulation of a, of a train or of an airport, so those are good assignments. Students will get involved in them. You can also ask editors to be designed as a part of lab assignments. Even as early as the third or fourth, I think, sorry the fifth lecture, when you teach the if statement, you can build an, an elementary editor to draw on the screen.

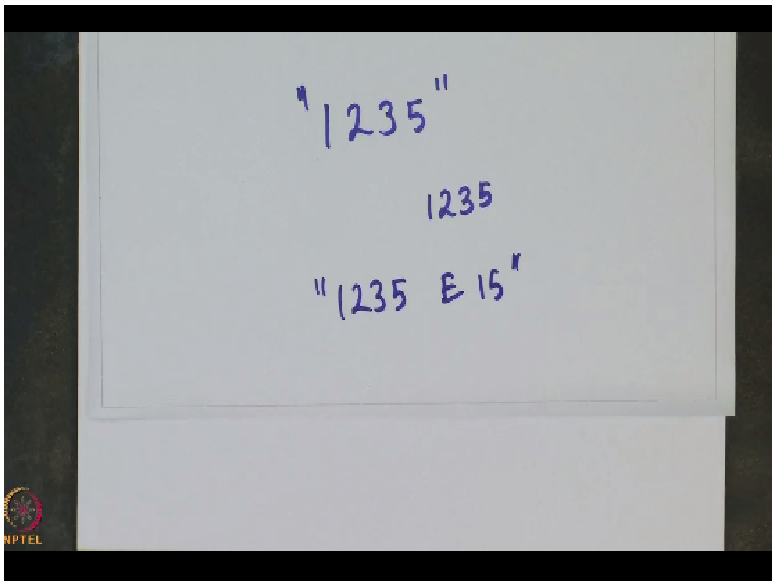
So these are problems which look big but students can actually write it in 15-20 lines and that makes the students sort of feel very accomplished. Now there are many ideas that you may not be able to introduce in class because you may not have the time. So some of them can be introduced in, in the lab and if they are game like, some students might be familiar and this 20 questions game which I will talk about in a minute is one such idea. Initial weeks, the programs will have to be small.

Later weeks, medium-sized programs can be expected. So in that case, you should break up the program as we discussed earlier when we talked about how to design medium-sized programs in the lecture sequence on advanced programming topics. So help them out. So if you want them to write say maybe a 100-line program, help them in designing it. So give them a main program, expect them to write the rest, something like that.

We have also said this regarding in-class questions but it applies even more so to lab assignments. That give some lab assignments so that you can ask related problems in examinations. So you build up the background and now you can ask problems which are perhaps slightly harder but since they have done some groundwork, the difficulty level is reduced effectively.

So for example, the lab assignment could be, I am giving you a text string and from that extract the integers.

(Refer Slide Time: 29:23)



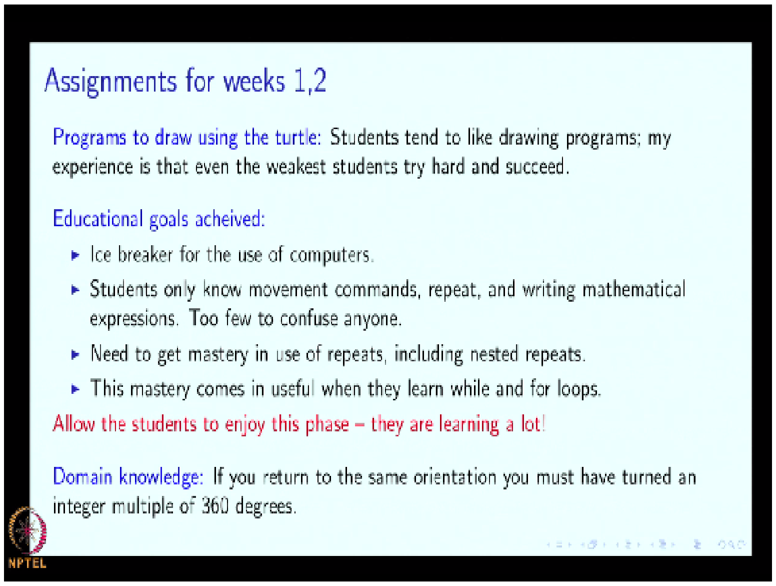
So for example, I may have 1 2 3 5 in a text string. So from this, so this is of either string class or this is a null terminated character string and from this, they are supposed to extract the value 1 2 3 5. How do they do that? Well, they have to read 1, the ASCII character, subtract the ASCII character 0 from it so that they get the place value of that 1, then maybe multiply by 1000, whatever, okay.

So then get in 2, get in 3, get in 5 and that way we can construct the number. Now this is actually an interesting assignment because a computer program has to do this. So this assignment is not just a programming exercise but it tells the students, oh, when I type 1 2 3 5, the computer is actually doing what this lab is asking you to do, okay. So that is one reason for it. But there is another reason.

So if you ask this in a lab, then in your exam, you could ask them to write, ask them to take 1 2 3 5 E 15 so which means 1 2 3 5 *10 raise to 15 and generate a floating point number, store that value in a floating point number. So this could be an exam problem. But if you had given the exam problem directly, then students would have had some additional difficulty, okay. Students would, would not get this idea that to extract the digit value, I have to subtract just the character 0 from whatever character I read.

So once your, once that principle is through in the lab assignment, assignment, then this is not, this is sort of a fair question for an examination.

(Refer Slide Time: 31:24)



Assignments for weeks 1,2

Programs to draw using the turtle: Students tend to like drawing programs; my experience is that even the weakest students try hard and succeed.

Educational goals acheived:

- Ice breaker for the use of computers.
- Students only know movement commands, repeat, and writing mathematical expressions. Too few to confuse anyone.
- Need to get mastery in use of repeats, including nested repeats.
- This mastery comes in useful when they learn while and for loops.

Allow the students to enjoy this phase – they are learning a lot!

Domain knowledge: If you return to the same orientation you must have turned an integer multiple of 360 degrees.

Now, when you teach this course, the assignments for the first couple of weeks are a little bit difficult to design, design, okay. But what you can do here is you can ask them to write programs using the turtle. So students actually like these programs, these drawing programs quite a bit and my experience is that even the weakest students try to work very hard on this and they succeed. So they are, even the weaker students are making an effort and they are learning.

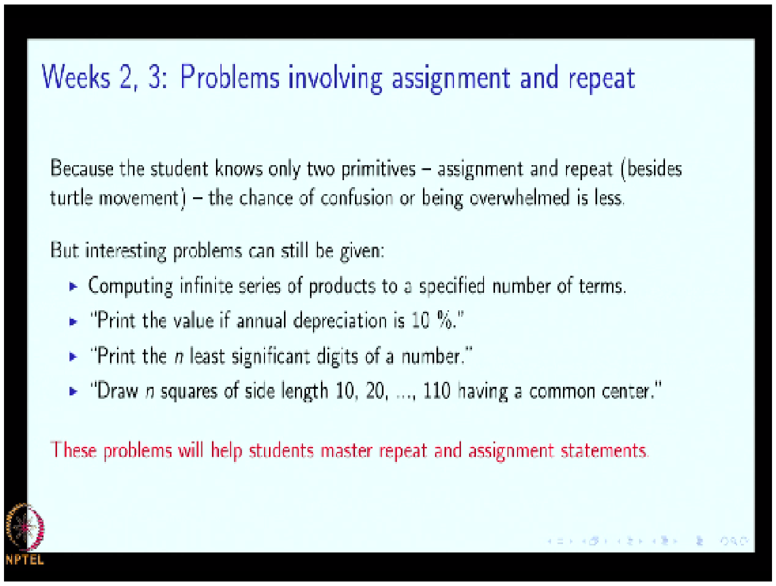
So that is, that is very good news. And they actually achieve educational, serious educational goals. So far example, in the first 2 weeks, students are, maybe some students are afraid of computers, okay. So this is a simple way by which they are persuading themselves, oh, computers are actually fun, they are easy to use and things like that. And now students only know the movement commands and the repeat command and maybe writing mathematical expressions.

So they only know, so 2 or 3 things. So if you give them a problem, they only have 2 or 3 tools at their disposal. So they are not going to get confused because they have 10 tools at their disposal and they do not know which one to use. So all the more reason to tell them how to use those tools in the best possible manner. So it is important that you give assignments at this point. Because the students will not be confused.

The students will have limited choice but they will learn to exercise that limited choice first and only later, will they face the problem of having a big wide choice. Now repeats look simple but repeats can be nested and iteration is a very fundamental idea and even these picture drawing problems students will get the mastery of repeats or basic iteration and in fact, even nested iteration. I would say let the students enjoy the phase, okay.

They are having fun but they are also learning a lot. So do not, do not hurry, okay. Now the, along with this, you should give them some domain knowledge, okay. So for example, you should tell them that if the turtle is going to come back and point in the same direction, you know that it is going to point in the same direction, then the total turning that you have done in this entire period must be a multiple of 360. So if you tell them this principle, their writing programs will become much faster.

(Refer Slide Time: 34:11)



Weeks 2, 3: Problems involving assignment and repeat

Because the student knows only two primitives – assignment and repeat (besides turtle movement) – the chance of confusion or being overwhelmed is less.

But interesting problems can still be given:

- Computing infinite series of products to a specified number of terms.
- "Print the value if annual depreciation is 10 %."
- "Print the n least significant digits of a number."
- "Draw n squares of side length 10, 20, ..., 110 having a common center."

These problems will help students master repeat and assignment statements.

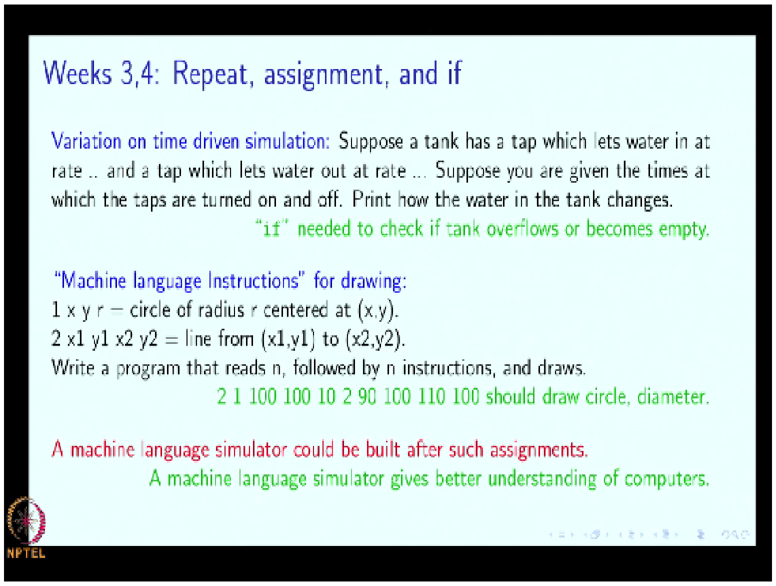
So week 2, 3, again, there is a similar issue. They have not learnt enough. So you cannot in the normal course of teaching, you cannot give them interesting assignments but if you have taught repeat as I hope you would have, then interesting assignments are still possible. So again, the idea is that you should make use of this occasion, they know the assignment statements and they know the repeats besides turtle movements.

So again you should give assignments because they are going to be less confused because they only know these 2-3 things and they will not be overwhelmed. So you should give assignments. So interesting problems can still be given. So computing infinite series of products to a specified number of terms can be done. Printing the value of say, if the annual depreciation is 10, print the value, okay.

That kind of problems can be given, okay. Print the n least significant digits of a number, problems like this can be given, okay. Draw n squares of side length 10, 20, 30, 40, whatever 110 having a common center. This can be given because this can be done simply by using the repeat and assignment, assignment a change, change the length of the square to increase the length of the square.

So these problems will help students master repeats as well as assignment statements. So say in the square drawing program, the student will need to have a variable whose length they are going to keep on changing. So that would be a good, good assignment for mastering the assignment statement.

(Refer Slide Time: 36:00)



Weeks 3,4: Repeat, assignment, and if

Variation on time driven simulation: Suppose a tank has a tap which lets water in at rate .. and a tap which lets water out at rate ... Suppose you are given the times at which the taps are turned on and off. Print how the water in the tank changes.

"if" needed to check if tank overflows or becomes empty.

"Machine language Instructions" for drawing:
1 x y r = circle of radius r centered at (x,y).
2 x1 y1 x2 y2 = line from (x1,y1) to (x2,y2).
Write a program that reads n, followed by n instructions, and draws.

2 1 100 100 10 2 90 100 110 100 should draw circle, diameter.

A machine language simulator could be built after such assignments.

A machine language simulator gives better understanding of computers.

Week 3, 4, they have repeat assignment. They have learnt repeat, they have learnt assignment statement and they have learnt if. So at this point, you can actually give many interesting things. Again, knowing the repeat statement is the key. If they do not know the repeat statement, then you cannot. But since they know the repeat statement, you can. So you can give problems like this. Suppose a tank has a tap which lets water in at some rate and the tap which lets water out at some rate.

Suppose you are given the times at which the taps are turned on and off, print how the water in the tank changes, okay. So if they know how many times, the water is turned on and off, then they just have to check whether the tank was empty, whether the tank was overflowing and that is about it. So this can be done just using repeat assignment and if, okay. Now you can also do things like, you can design machine language instructions for drawing.

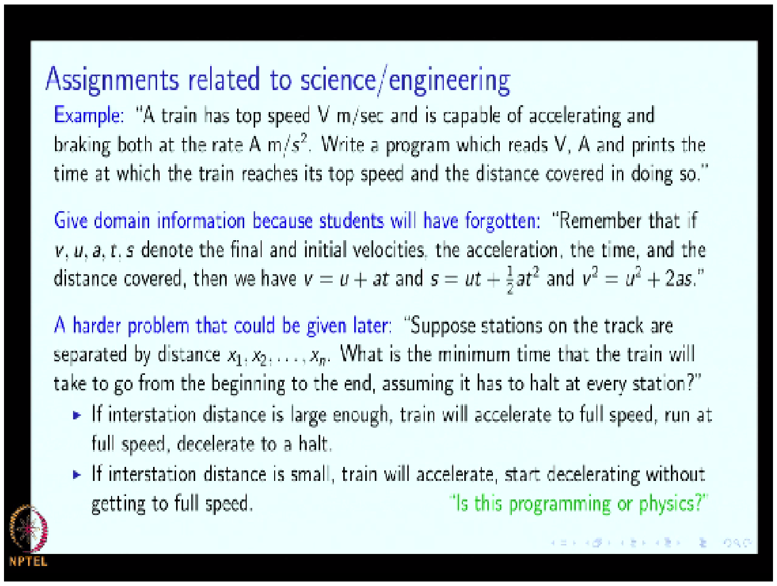
So for example, you can say that let us play a game. Let us have some codes. So if I say 1 x y r, say maybe I say 1 10 20 30, what does it mean? It means I want you to draw a circle of radius r centered at the coordinates x y. So that is my code, that is my secret code if you want for suggesting a circle. Maybe if I say 2 x1 y1 x2 y2, so 2 followed by 4 numbers then I am telling you in a game, in a secret game if you like that I want a line drawn from x1 y1 to x2 r2, okay.

So why do not you write a program that reads n followed by n such instructions and draws whatever is needed. So this is kind of a machine language instructions for drawing. So for example, I might give this sequence of numbers. So what is that sequence? So first, the first 2 says how many instructions are there. So there are 2 instructions. What is the first instruction? 1 100 100 10.

So it is saying I want a circle of radius 10 at centered at x y, or at 100 100. The next says I want a line from 90 100 to 110 100. So effectively what I am drawing, what I am asking to be drawn is the circle and its horizontal diameter. So the student program should be able to take such numbers and draw the required picture. So this is, this is a fun program, also an instructive program because the notion of using numbers as codes is a very key part of programming and they are getting it over here.

Subsequently, you can build a machine language simulator also, okay. A machine language simulator, okay, which has been discussed or say using the machine language discussed in chapter 2 of the book is useful because it tells the student more about computers. It is sort of reinforces the idea that computer has this machine language and that the instructions in the machine language program are executed one after another.

(Refer Slide Time: 39:19)



Assignments related to science/engineering

Example: "A train has top speed V m/sec and is capable of accelerating and braking both at the rate A m/s^2. Write a program which reads V, A and prints the time at which the train reaches its top speed and the distance covered in doing so."

Give domain information because students will have forgotten: "Remember that if v, u, a, t, s denote the final and initial velocities, the acceleration, the time, and the distance covered, then we have $v = u + at$ and $s = ut + \frac{1}{2}at^2$ and $v^2 = u^2 + 2as$."

A harder problem that could be given later: "Suppose stations on the track are separated by distance $x_1, x_2, \ldots, x_n$. What is the minimum time that the train will take to go from the beginning to the end, assuming it has to halt at every station?"

- If interstation distance is large enough, train will accelerate to full speed, run at full speed, decelerate to a halt.
- If interstation distance is small, train will accelerate, start decelerating without getting to full speed.

"Is this programming or physics?"

We can also give assignments related to science and engineering. So for example, if a train, a train has top speed V m/sec and is capable of accelerating and braking both at the rate of A m/sec, write a program which reads V, A and prints the time at which the train reaches its top speed and prints the distance covered in doing so. So this does not require loops but this does require the if statement.

So this is a good time to give such a program after you have taught the if statement. Now here you may need to give the domain informations, information because the students might have forgotten and you are, you are not trying to test their physics knowledge. You are trying to test their programming knowledge. So give them the physics knowledge. So what they should be, should be able to do is to use their physics knowledge in their program.

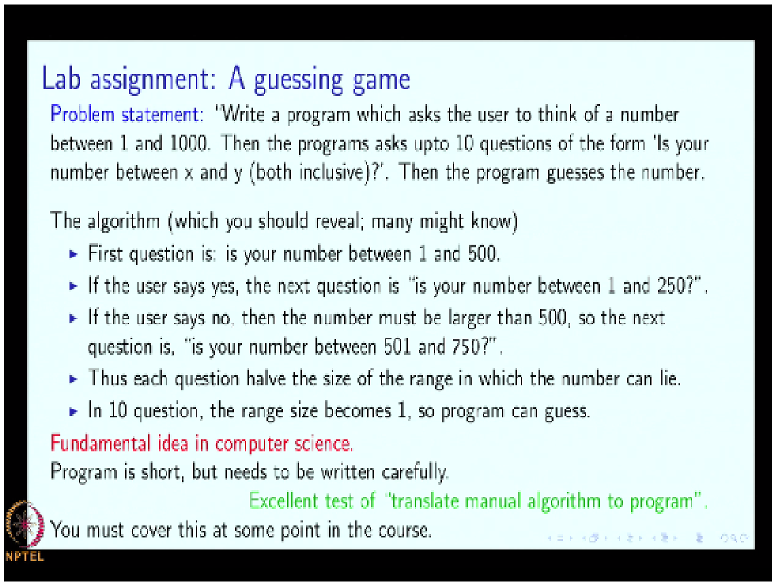
So tell them that the kinematics rules are v=u+at s=ut+1/2at square and v squared=u squared+2as. You can give them all the equations even if all of them might, might not be needed in the program. Because picking which equations are needed is also an important skill. A harder problem that could be given later is that suppose I have stations on the track and they are separated by some specific, specified distances.

What is the minimum time that the train will take to go from the beginning to the end assuming it has to halt at every station? If the interstation distance is large enough, then the train will accelerate to full speed, run at full speed, decelerate to a halt, okay. If the interstation distance is small, the train will accelerate, start decelerating without getting to a full speed and so they have to distinguish the 2 cases.

Actually, in the first example, I guess the if statement is really not needed because we are just saying when does the time train reach the top speed and you have equations exactly for predicting that. But for this harder problem, the if statement will be needed, okay. So is this programming or physics? Well, programming is always about a domain and in this case, it is about physics. So that is okay.

The point is that in the minds of the students, programming must get integrated with other subjects and therefore, such examples are necessary. You can also give a guessing game.

(Refer Slide Time: 41:53)



Lab assignment: A guessing game

Problem statement: "Write a program which asks the user to think of a number between 1 and 1000. Then the programs asks upto 10 questions of the form 'Is your number between x and y (both inclusive)?'. Then the program guesses the number.

The algorithm (which you should reveal; many might know)

- First question is: is your number between 1 and 500.
- If the user says yes, the next question is "is your number between 1 and 250?".
- If the user says no, then the number must be larger than 500, so the next question is, "is your number between 501 and 750?".
- Thus each question halve the size of the range in which the number can lie.
- In 10 question, the range size becomes 1, so program can guess.

Fundamental idea in computer science.
Program is short, but needs to be written carefully.
Excellent test of "translate manual algorithm to program".
You must cover this at some point in the course.

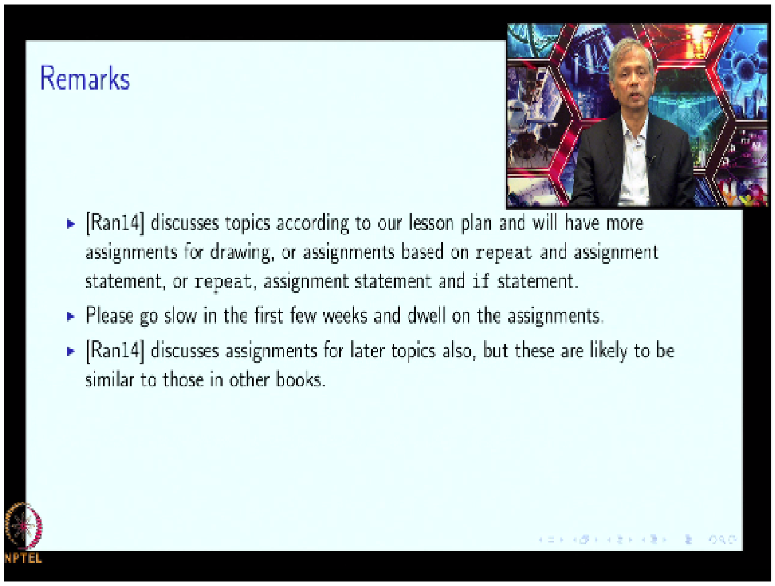
So for example, the problem statement over here could be, write a program which asks the user to think of a number between 1 and 1000. Then the program asks up to 10 questions of the form, is your number between x and y, both inclusive and after asking 10 questions, the program guesses the number. So the algorithm which you should reveal and many might know the algorithm already is that the first question should be, is your number between 1 and 500.

After that if the user says yes, the next question should be, is your number between 1 and 250, okay. If the user says no, then the next number must be larger than 500, that is what you get from the answer and so the next question is, is your number between 501 and 750. So here it should be 750. So essentially each question is going to halve the size of the range, range in which the numbers can lie.

So in 10 questions, you will reduce the size, range size becomes 1, so that the program can guess and this, this binary search, this is nothing but binary search. People know it as a game or even if they do not know, they always want to guess and so this is a good way of introducing them to an important principle, okay.

Program is short but it needs to be written carefully. And it is an, it is an excellent test of translate manual algorithm to a program because they will be able to get the manual algorithm. My feeling is that this is a nice, really nice problem, really important problem, really fun problem and you should try to cover it at some point in the course.

(Refer Slide Time: 43:38)



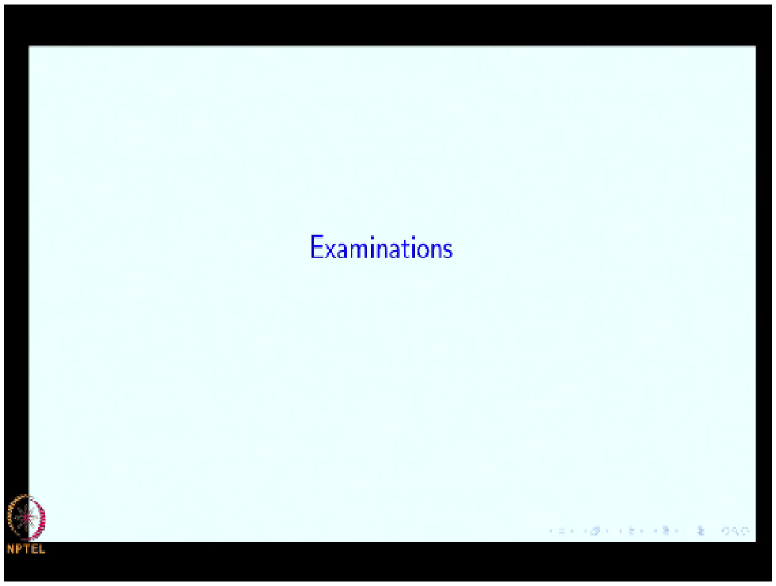
Remarks

- [Ran14] discusses topics according to our lesson plan and will have more assignments for drawing, or assignments based on repeat and assignment statement, or repeat, assignment statement and if statement.
- Please go slow in the first few weeks and dwell on the assignments.
- [Ran14] discusses assignments for later topics also, but these are likely to be similar to those in other books.

So some remarks, the book discusses topics according to our lesson plan and will have more assignments for each, each week if you follow the plan, okay. So earlier you will only have assignment repeat and assignment or repeat assignment and if but appropriate for that amount of learning, the book has more assignments that can be given. I suggest you should go slow in the first few weeks and dwell on the assignments.

Let students work on writing programs when they only have a small number of things to choose from, okay. So that will increase their confidence. Now the book of course, discusses assignments for later topics also but these are likely to be similar to other books. So I think the initial weeks are critical and for that you should look at our book for deciding what to ask.

(Refer Slide Time: 44:41)



Examinations

So this concludes the first part of this lecture. I will stop here and then in the next lecture; I will come to examinations. Thank you.