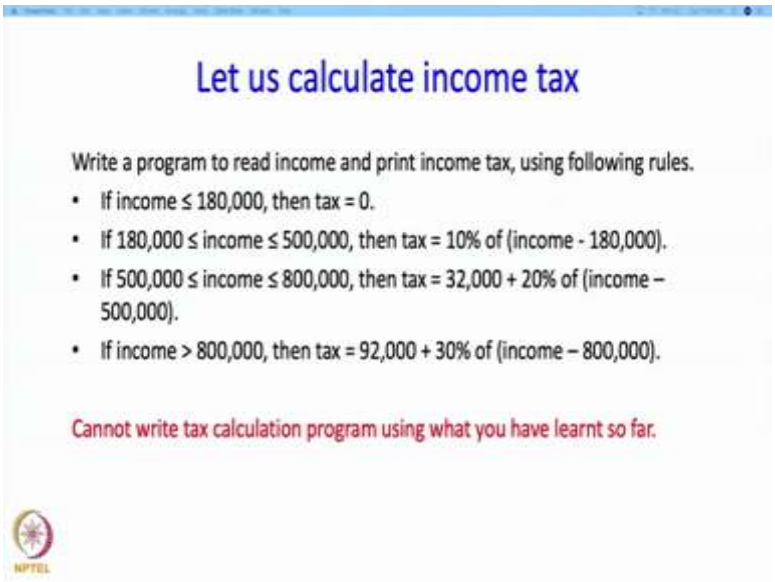
An Introduction to Programming through C ++
Professor Abhiram G . Ranade
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Lecture No. 6 Part - 1
Conditional Execution

Hello welcome to the NPTEL course on an introduction to programming through C++, I am Abhiram Ranade. Today's lecture is about conditional execution and the reading for it is chapter 6 of the book.

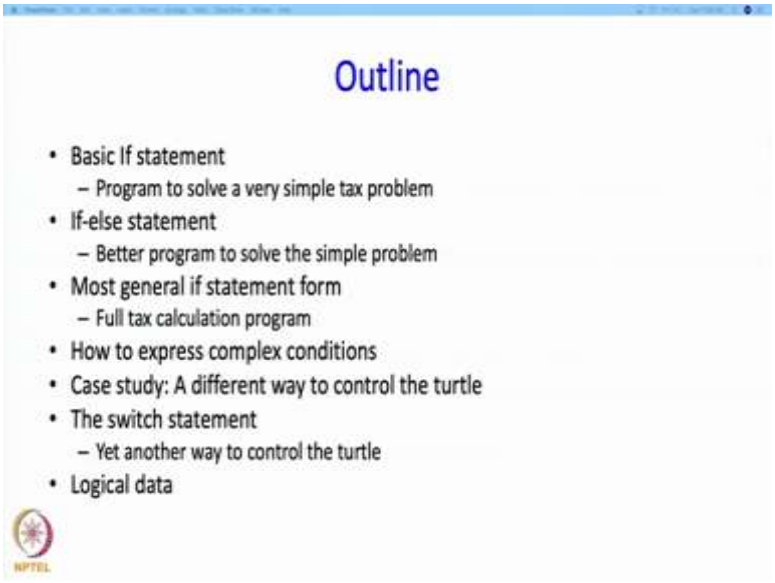
(Refer Slide Time: 0:39)



So let me begin with an example, suppose, we want to calculate income tax. So, let us say we want to write a program which reads the income and prints the income tax and the rules are as follows: if the income is less than 180 thousand or 1 lakh 80 thousand, then tax must be 0. So 0 should be printed, if the income is between 180 thousand and 500 thousand, then the tax should be 10 percent of the income over and above 180 thousand.

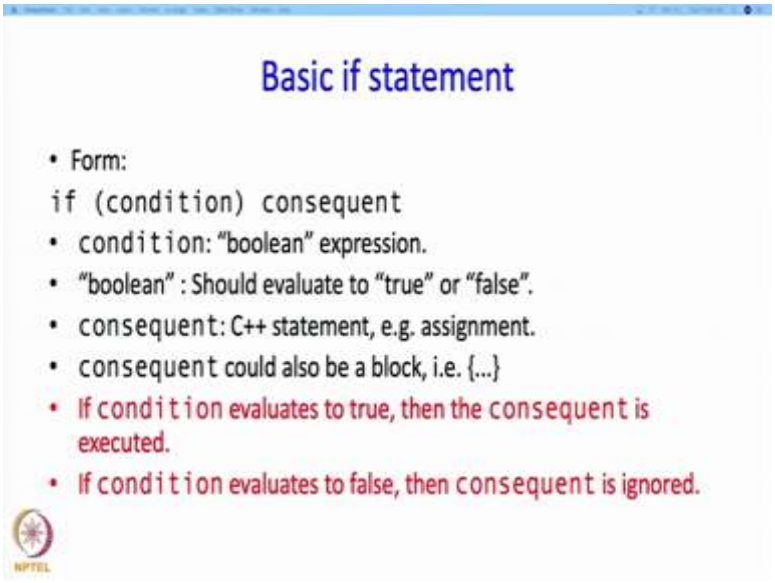
If the income is between 500 thousand and 800 thousand, then the tax should be 32 thousand plus 20 percent of the income above 500 thousand, and if the income is above 800 thousand or 8 lakhs, then the tax should be 92 thousand plus 30 percent of whatever income is there above 8 lakhs. Now, you will realize that you cannot write this program using what you already know so we need to know something more.

(Refer Slide Time: 2:11)



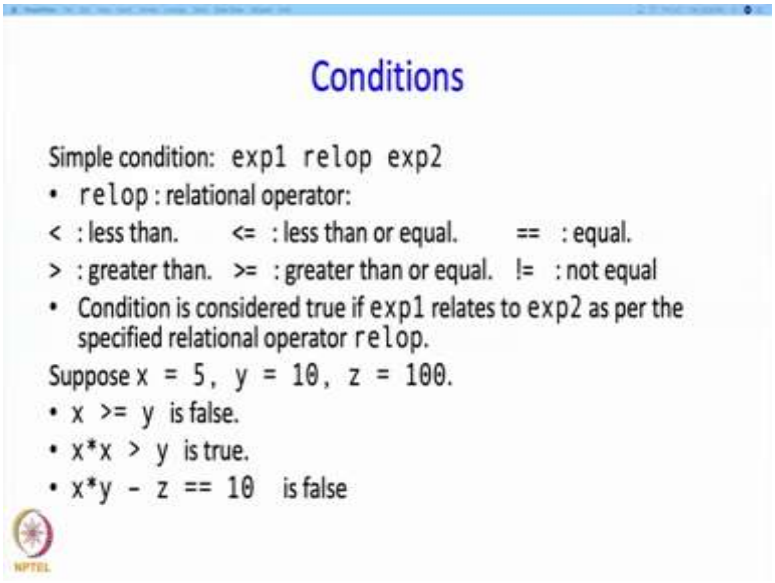
So here is what we are going to do in this lecture, I am going to talk about a new statement called an 'if' statement which will be useful in writing this tax calculation program and several other programs. So we will begin by looking at the basic form of this if statement. We will use that basic form to solve a simplified tax calculation problem well actually we are not going to be calculating tax, we will see this simple problem in a minute. After that we will see second form of the if statement or the so called 'if-else' statement. Using this we will be able to write a better program to solve the simplified problem. Finally, we will study the most general if statement form and using this we will be able to do our full tax calculation program. After that, we will write, we will see how to express compound conditions. So I have written complex over here, but I really mean compound. After that we will do a case study we will have a somewhat larger problem that we will solve and this will be about controlling the turtle in a different manner. Then, we will learn about the switch statement and we will see yet another way of controlling the turtle. Finally, we will study something called logical data.

(Refer Slide Time: 3:55)



Okay, so let us start with the basic if statement. So the form of this is 'if(condition)' and then the consequent. Here, condition is Boolean expression and I will explain what this means in a minute. A Boolean expression is something that evaluates to true or false, we will see this. Consequent is a C++ statement, for example, it could be an assignment statement and consequent could also be a block, again what I mean by a block is a set of statements enclosed inside parentheses. Now, the way this condition executes is as follows. So if, so we evaluate the condition and it if it evaluates to true, then the consequent is executed. If condition evaluates to false, then the consequent is ignored. I have not told you what exactly the condition is and what it means to evaluate it. I will do so next.

(Refer Slide Time: 5:12)



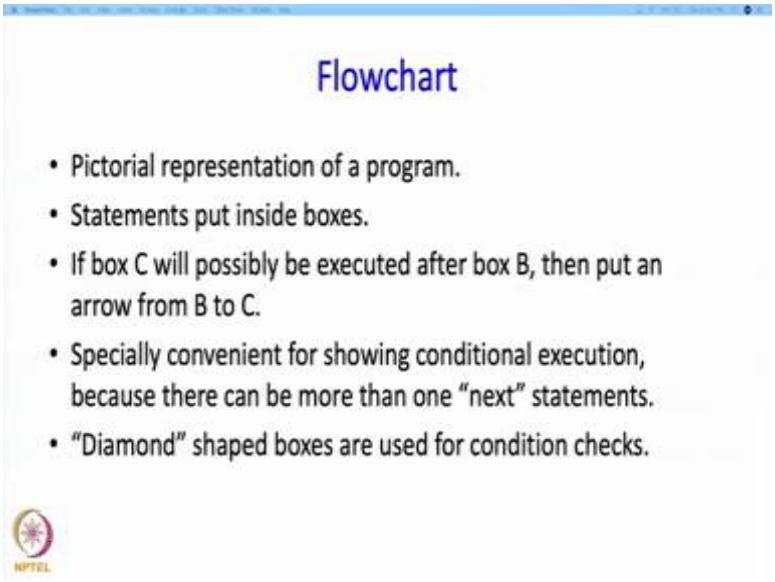
Okay, so a condition is as follows, so I am going to first tell you what a simple condition is. A simple condition looks like some expression followed by something called a relational operator followed by another expression. So a relational operator can be the less than symbol (<), the less than followed by equal to characters which together constitute the less than or equal operator (<=) or two equal to characters (==). So these two equal together constitute the relational operator equal. Notice that, if I just write the equality then in C ++ this means the assignment operator. So since we have already used the equality for the assignment operator now we have to use something different and C ++ designers decided that we will write equal to equal to, to denote the relational operator which compares two expressions. Similarly, we have a greater than (>) as a relational operator, greater than or equal to (>=) as relational operator and not equal (!=), so not equal is the exclamation mark followed by the equality character. So remember that there should not be any spaces between the exclamation mark and the equality and similarly between greater than equal and so on.

So, the condition is considered true if expression 1 relates to expression 2 as per the specified relational operator 'relop', so let me take this through examples. So suppose we have x=5, y=10, z=100. Then if we write x>=y, then that is going to be false because 5 is not greater than or equal to 10. If we could write x*x>y, now x is 5 and x square is 25, 25 is greater than y because y is 10. So therefore, this expression, this condition x square greater than y is going to evaluate to true.

So another example, I could write x*y-z==10, so, what is x times y? x times y is 50 minus z minus 100 the whole thing is minus 50 and minus 50 is not equal to 10 and therefore, this

entire condition is false. So what I told you over here is I have told you what conditions are and what does it mean to evaluate that. So now, I have told you everything that you need to know in order to understand the 'if' statement.

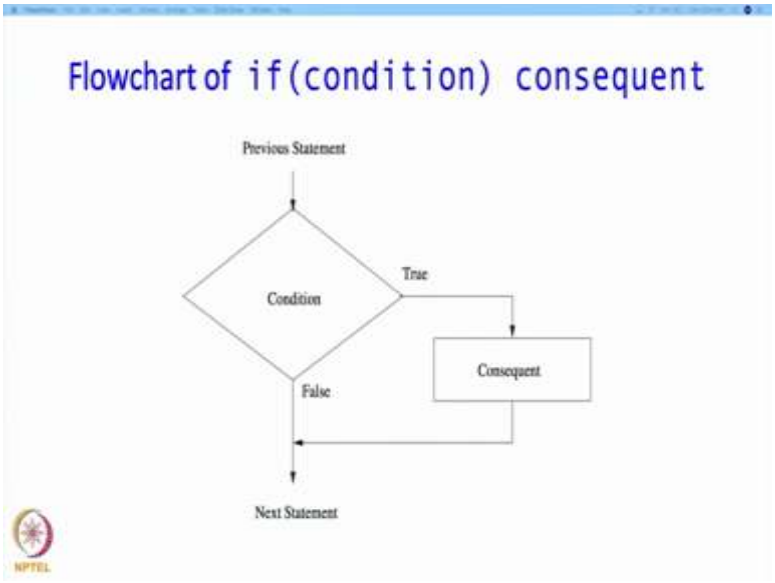
(Refer Slide Time: 9:17)



Okay now, it is customary to describe the if statement as a flow chart and what is a flow chart? It is a pictorial representation of a program or a statement. So in this, statements or even parts of statements are put inside boxes and if box C will possibly be executed after some box B then we put an arrow from B to C. So this allows us to understand how control flows in between the statements so the arrows indicate how the control flows and therefore, such a chart is called a flow chart.

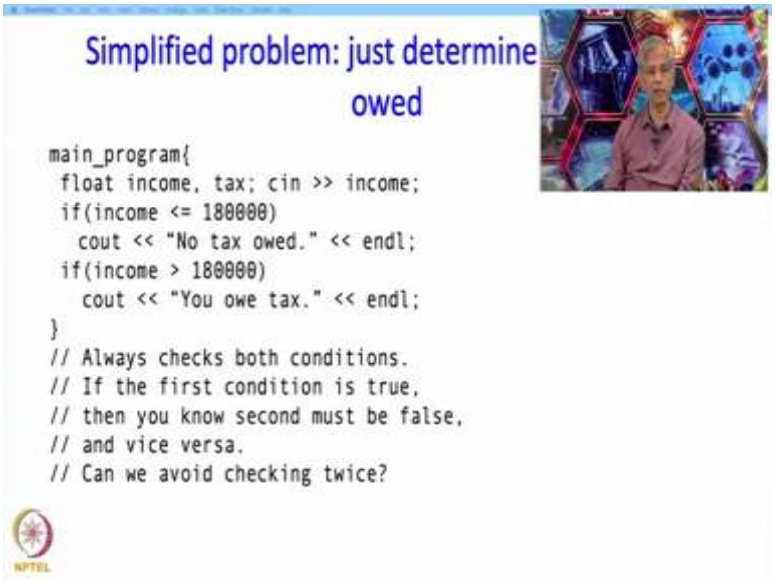
Now, this is specially convenient for showing conditional execution because in conditional execution there can be more than one next statements. So if I have a condition, if something consequent, then I may execute the consequent after checking condition, but I may not, I may directly go on to the next statement. So this is possible to be shown very clearly using a flowchart and assignment statements are usually put in rectangular boxes. Diamond shaped boxes are used for condition checks, so we will see this in a minute.

(Refer Slide Time: 10:38)



So, if I have a statement if condition consequent then its flowchart is as shown in this picture okay so this statement will be preceded in the program by a previous statement, so its flow chart will come over here this program this statement will be succeeded in the program by a next statement and its flow chart will come over here. So the idea is that previous statement will get executed followed by after that it is executed then a the control use this arrow to enter the if statement and now the condition is going to be evaluated. Now, this diamond indicates the condition and if the condition comes out to be true, then the control moves out in this direction where we have put true. So if the condition is true then after that the consequent will get that evaluated, after the consequent is evaluated the control will come out and it will go on to the next statement. If on the other hand the condition was false, then this branch is taken out of the diamond and it will directly go to the next statement. So what I have told you using this flowchart is not really different from what I told you earlier. However, I believe that this is likely to be perhaps clearer or more easy to see, it is probably more easier to see what is going on over here rather than when I wrote down that text where I gave you that text description.

(Refer Slide Time: 12:33)



Okay, so now, we can write the code to just determine if any tax is owed, so this is a simplified problem, our original problem was calculate the tax actually. Now you are saying let us just do something simple first, just determine if any tax is owed. So we are going to read in the income, but having read in the income we are just going to print a message saying yes tax is owed or no tax is owed. Okay, so here is what the program looks like you probably will be able to write it so I am not going to make deal of explaining it and let us just jump into it.

So first I am going to have a variable in which we are going to store the income and maybe we will have a variable tax. In this case variable tax is useless, but it does not matter we have it, then we are going to read in the income, so we are going to read the income form the keyboard, we could have put in the message saying cout give your income, but that is okay. So, let us say that is understood when the user invokes our program the user will type that message anyway without receiving any prompt for it. If, now we check if the income is less than 180000 you remember from the rules that in that case there is no tax, so what do we do? So then we are going to cout “no tax owed”, alright? However, if income is bigger than 180000 then we are going to print out “you owe tax”, okay so we read the income then we compare that with 180000 okay so this is expression 1, this is relational operator and this is expression 2.

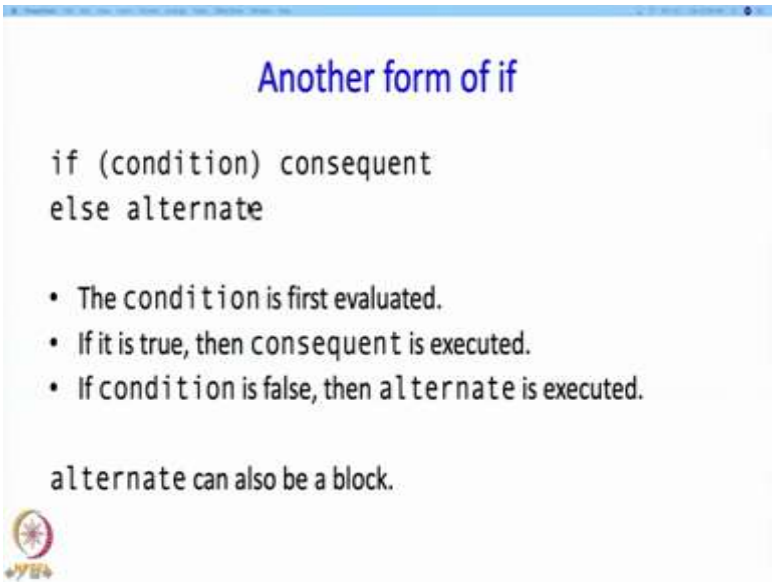
If the income is actually less than 180000, we execute the consequent which is this and the consequent simply requires us to print this message. So, if we execute this and if it comes out to be true, we execute this and we come on to this statement, if this comes out to be false that

is this income is bigger, then we directly come to this statement without printing anything so far.

So in any case after executing the statement we come over here control comes over here and we check is income bigger than 180000, if it is bigger than 180000 than we are going to print a message saying “you owe tax”. Otherwise, we are going to directly equal go to the next statement so the next statement is not there so we terminate the program okay so very simple.

Now, from the description that I just gave you will observe that I am going to check both these conditions on every execution, okay. So I will this condition and maybe I will print out this message or maybe I do not printout this message, but I again so I subsequently check this condition, okay. So the program is correct, okay it does what we want to do, but you may observe and you may be think it surprising that the program needs to check both of these conditions because after all if the first condition is true then we know that the second condition must be false so why should we check it. So, we will see that this can be remedied and we can write a slightly better program for this okay so anyways to summarize this, this is a perfectly fine program it does the job we want except that it checks a condition 2 times and you would like to know if we can avoid checking twice and indeed we can.

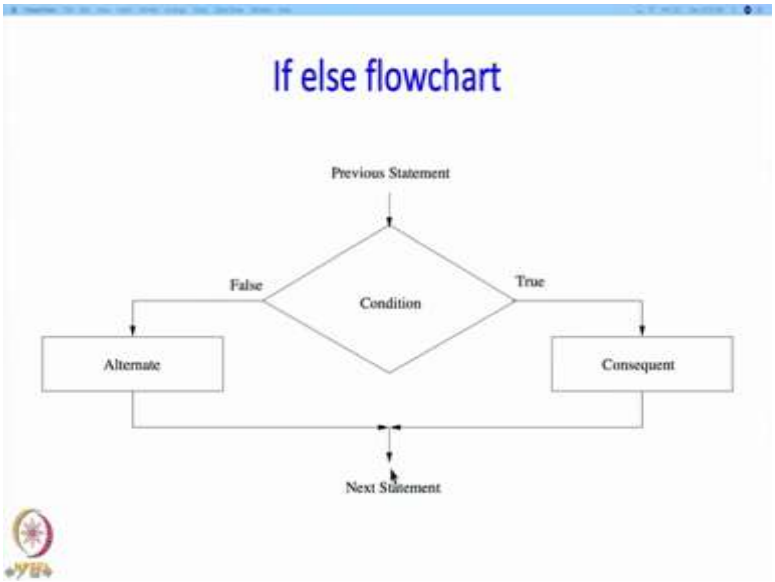
(Refer Slide Time: 16:48)



Okay, so for this we need another form of if statement so this form is ‘if(condition)-consequent, else-alternate’, so in this the condition is first evaluated, if it is true then consequent is executed, if condition is false than alternate is executed and alternate as well as the consequent can both be blocks.

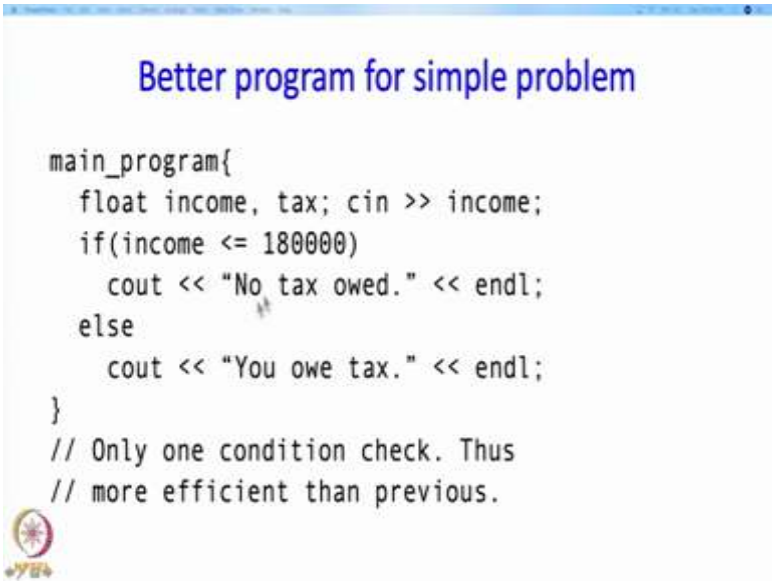
Okay, so what does it mean to have a block over here? So it means that if condition is true all these statements in the block will be executed and likewise if the condition is false all these statements in the block over here can be executed will be executed.

(Refer Slide Time: 17:37)



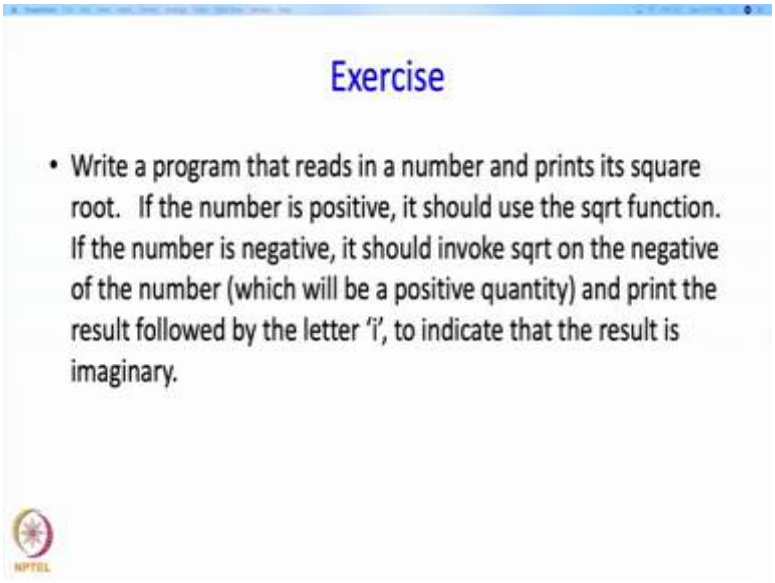
So here is, if else is a flow chart so after executing the previous statement before the if else statement we enter the flow chart of the if else and the action that we perform is we evaluate the condition, if the condition is true we proceed out of this branch and then we execute the consequent after executing the consequent we go on to the next statement in the program. If the condition is false we proceed out of this branch we execute the this branch we execute the alternate and we after that we go to the next statement.

(Refer Slide Time: 18:30)



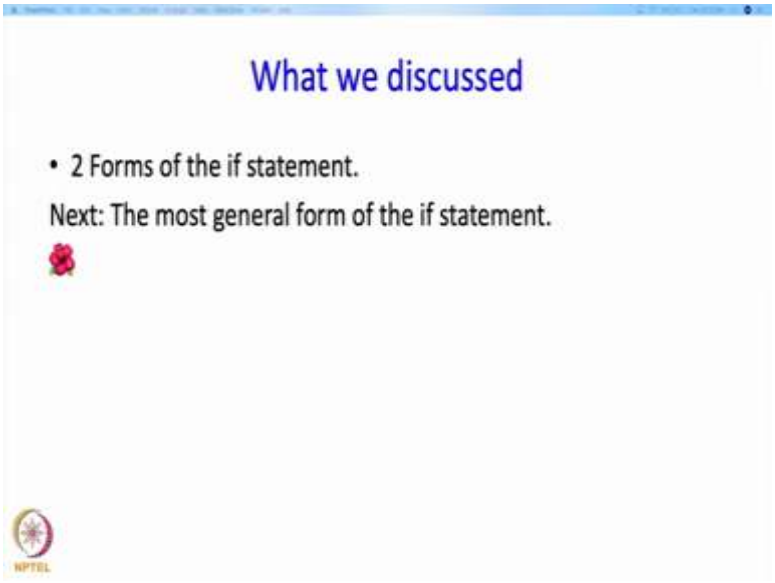
So, you will see that this statement was sort of design to improve our tax calculation program so here it is we are we have the same part so far but now instead of checking whether income is greater than 180000 since we already know we can just say “you owe tax” because the else is going to be executed exactly if income is greater than 180000. So the else part or the alternate part will be executed only if income is bigger than 180000 in which case “you owe tax” will be printed. So in other words what we have do is, we are done is we are checking the condition only once, only one condition is being checked and so in that sense this program is a little bit more efficient than the previous program and also it is a little less verbose. So by writing else you immediately understand what is going on, otherwise if you write the whole condition again then you have to carefully observe that oh that this condition is really the compliment of this condition, by else you know much more clearly that this is going to be executed or this is going to be executed, one of the two is going to be executed.

(Refer Slide Time: 19:56)



Okay, so an exercise write an program that reads in a number and prints its square root. If the number is positive, it should use the square root function, your program should use the square root function. If the number is negative, your program should invoke the square root function on the negative of the number. So that will be a positive quantity therefore, you can invoke square root on it and so you should print the result followed by the letter 'i' to indicate that the result is imaginary.

(Refer Slide Time: 20:36)



So, what have we discussed? We have discussed 2 forms of the if statement. Next, we will discuss the more general form, but before that let us take a quick break.